[Team LiB]

NEXT



- Table of Contents
- Index
- Examples

HP-UX 11i System Administration Handbook and Toolkit, Second Edition

By Marty Poniatowski

START READING

Publisher: Prentice Hall PTR

Pub Date: February 10, 2003

ISBN: 0-13-101883-3

Pages: 1440

Your complete single source for HP-UX system administration!

Now, there's a single source for all the information HP-UX administrators needto succeed! Marty Poniatowski has updated the world's #1 HP-UX systemadministration guide to cover all the latest HP-UX 11i enhancements, plus all theUNIX commands HP-UX sysadmins need. This edition also contains comprehensive newcoverage of the latest options and tools for HP-UX partitioning!

HP-UX in depth...

- System boot: PDC, ISL, hpux, GSP, Secure Web Console, and Virtual Partition Booting
- Installing HP-UX, Software Distributor, and Operating Environments
- HP-UX kernel—including kernel update procedure
- Device files
- Users/groups, Ignite-UX backup/restore, and startup/shutdown scripts
- System Administration Manager (SAM)
- Logical Volume Manager with extensive examples—plus Veritas Volume Manager
- Performance management: iostat, vmstat, netstat, top, sar, and more
- Networking: Ethernet, TCP/IP, DNS, NFS, CIFS/9000, Samba, and more

- CDE and GNOME

UNIX commands and shells...

- File Types
- File Commands: cd, cp, file, ls, mv, mkdir, pwd, rm, rmdir, and more
- File Viewing: cat, head, tail, and more
- Advanced Commands: grep, sed, awk, find, cut, paste, split, sort, cmp, diff, comm, and more
- Bash, Korn, and C shells and shell programming
- Manual pages for many commonly used commands are included in the appendices

New HP-UX partitioning coverage...

- vPars: preparation, creation, configuration, and commands
- nPartitions: creation, modification, cells, and commands
- HP Process Resource Manager (PRM)
- MC/ServiceGuard: background and examples

[Team LiB]

NEXT



- Table of Contents
- Index
- Examples

HP-UX 11i System Administration Handbook and Toolkit, Second Edition

By Marty Poniatowski

START READING

Publisher: Prentice Hall PTR
Pub Date: February 10, 2003
ISBN: 0-13-101883-3
Pages: 1440

Copyright
Hewlett-Packard® Professional Books
Preface
About The Book
Relevant URLs
Manual Pages Supplied with This Book
Acknowledgments
Conventions Used in this Book
Section 1. HP-UX System Administration
Chapter 1. Booting (PDC, ISL, hpux), Secure Web Console, GSP Configuration, vPars Booting
Background
Boot Process Overview
Boot Console Handler (BCH) & Processor Dependent Code (PDC)
Initial System Loader
HPUX Secondary System Loader (hpux)
Consoles
Configuring the Guardian Service Processor (GSP)
Virtual Partition Boot Process Overview
Extensible Firmware Interface (EFI)
Manual Pages for Commands Used in Chapter 1
Chapter 2. Installing HP-UX
Topics Covered in this Chapter
Background on HP-UX 11i Operating Environments

- Installing the HP-UX 11i Operating Environment
- Setting the System Parameters after Booting
- Software Distributor Example
- Loading Patches
- Software Distributor Background
- Manual Pages for Commands Used in Chapter 2

Chapter 3. Logical Volume Manager and Veritas Volume Manager

- Introduction
- Veritas Volume Manager
- Logical Volume Manager Background
- Commonly Used LVM Procedures
- Adding Disks
- Reconfiguring Some Disks - An Example of Using Some Logical Volume Commands
- Optional Procedure to Exchange Dump Priorities
- HP VERITAS Volume Manager
- Some Additional File System-Related Commands

Chapter 4. Building an HP-UX Kernel

- Introduction
- kcweb

Chapter 5. Devices

- Background on Device Files and Peripherals
- Device Files in HP-UX 11i
- An Example of Adding a Peripheral
- Device Files in Cell-Based Systems

Chapter 6. Backup

- Built-In Backup Programs
- Backup of Virtual Partitions

Chapter 7. System Startup and Shutdown Scripts

- Introduction
- System Startup and Shutdown Mechanism
- System Shutdown
- System Startup and Shutdown Scripts in Virtual Partitions
- Application Startup in Virtual Partitions
- Virtual Partition Startup
- Manual Pages for Commands Used in Chapter 7

Chapter 8. Users and Groups

- Set Up Users and Groups
- Assigning Users to Groups
- Virtual Partitions and Users
- NIS for Managing Users

Chapter 9. Ignite-UX

- Ignite-UX Overview
- Set up Server with Depots and Install Clients
- Run Ignite-UX GUI
- System Recovery with Ignite-UX
- System Recovery to Local Tape Drive with make_tape_recovery
- Creating Bootable Recovery Archives with make_net_recovery
- Using Ignite-UX GUI to Create a vPar Archive
- Manual Pages for Commands Used in Chapter 9

Chapter 10. System Administration Manager (SAM) and ServiceControl Manager (SCM)
- SAM and SCM
- SAM Overview
- Running and Using SAM as Superuser
- Author's Disclaimer - SAM Is a Moving Target
- Running Restricted SAM Builder
- Accounts for Users and Groups
- Auditing and Security
- Backup and Recovery
- Clusters
- Disks and File Systems
- Kernel Configuration
- Networking and Communications
- Partition Manager
- Performance Monitors
- Peripheral Devices
- Printers and Plotters
- Process Management
- Routine Tasks
- Run SAM on Remote Systems
- Software Management
- Time
- NFS Diskless Concepts
- ServiceControl Manager (SCM) Overview
- Manual Pages for Commands Used in Chapter 10

Chapter 11. Introduction to UNIX Performance Tools - Security
- Introduction
- Standard UNIX Commands
- Killing a Process
- Signals
- Show System Swap
- sar: The System Activity Reporter
- timex to Analyze a Command
- More Advanced and Graphical Performance Tools
- HP GlancePlus/UX
- HP OpenView Performance Agent and HP OpenView Performance Analyzer/UX
- Manual Pages for Commands Used in Chapter 11

Chapter 12. Networking Background
- An Overview of IEEE802.3, TCP/IP
- Internet Protocol (IP) Addressing and Subnet Mask

Chapter 13. Configuring and Using Networking
- Configuring Networking and LAN Adapters
- Commonly Used Networking Commands
- Using Networking
- **Internet Services ARPA** (Communication among Systems w/ Different OS)
- **Internet Services Berkeley** (Communication between UNIX Systems)
- Host Name Mapping
- Set Up DNS Servers Using BIND 8.1.2 and 9.2 on HP-UX 11i
- Network File System (NFS)

- Manual Pages for Commands Used in Chapter 13
- Chapter 14. Common Desktop Environment (CDE) and GNOME
  - Introduction
  - Why a Graphical User Interface (GUI)?
  - CDE Basics
  - Customizing CDE
  - CDE - Advanced Topics
  - X, Motif, and CDE Configuration Files
  - How Configuration Files Play Together
  - Specifying Appearance and Behavior
  - The Sequence of Events When CDE Starts
  - CDE and Performance
  - Font Paths
  - CDE and Virtual Partitions
  - Custom Greeting for Different Virtual Partitions
  - GNOME
- Chapter 15. CIFS/9000 and Samba
  - CIFS/9000 Overview
  - Samba Overview
  - Additional Samba Topics

Section 2. HP-UX Partitions

- Chapter 16. Virtual Partitions (vPars)
  - Introduction
  - About Virtual Partitions
  - Virtual Partitions Background
  - Uses of Virtual Partitions
  - Preparing to Create Virtual Partitions
  - Loading the Software Required for Virtual Partitions
  - Virtual Partitions Command Summary
  - Steps to Create Virtual Partitions
  - Virtual Partition Dump Files
  - Summary
  - vparmgr GUI
  - Manual Pages for Commands Used in Chapter 16
- Chapter 17. nPartitions Overview
  - Introduction to nPartitions
  - Cells and nPartitions
  - Ways of Managing nPartitions
  - HP-UX nPartition Commands
  - Modifying and Creating an nPartition
  - Partition Manager (parmgr)
  - Management Processor nPartition Commands
  - The Boot Process Boot Console Handler (BCH) Commands
- Chapter 18. HP Process Resource Manager (PRM)
  - PRM and WLM Overview
  - PRM Software
  - Working with PRM
  - PRM and Oracle Setup
- Chapter 19. MC/ServiceGuard

- About MC/ServiceGuard
- MC/ServiceGuard Background
- A Simple Example

Section 3. UNIX Background

- Chapter 20. UNIX File System Introduction - File System Layout, file and ls Commands
  - The Basis of UNIX
  - File Types
  - The file Command
  - The ls Command
  - File System Layout
  - Linux File System Layout
- Chapter 21. Working with Files and Directories - Permissions, Commands, File Name Expansion, and Wild Cards
  - Introduction
  - Permissions
  - Absolute and Relative Path Names
  - File Name Expansion and Wild Cards
  - pwd and cd
  - chmod
  - cp
  - mv
  - mkdir
  - rm
  - rmdir
  - Using Commands
- Chapter 22. Viewing Files - Redirection, cat, more, pg, head, and tail Commands
  - First Things First
  - Redirection
  - Viewing Files with cat, more, pg, head, and tail
  - Manual Pages for Commands Used in Chapter 22
- Chapter 23. UNIX Tools - split, wc, sort, cmp, diff, comm, dircmp, cut, paste, join, and tr
  - Not All Commands on All UNIX Variants
  - Manual Pages for Commands Used in Chapter 23
- Chapter 24. Advanced UNIX Tools - Regular Expressions, sed, awk, and grep
  - Three Commands
  - Regular Expression Words-of-Caution
  - Expressions Are Strings and Wildcards
  - sed
  - awk
  - grep
  - Manual Pages for Commands Used in Chapter 24
- Chapter 25. find Command
  - find Overview
  - Finding Files of a Specific Type
  - Find Empty Files and Directories
  - Finding Files By Name, Size, and Both Name and Size
  - Finding Files By Owner, Type, and Permissions
  - Finding Long Unaccessed Files and Running Commands on Found Files

Find Summary
Manual Pages for Commands Used in Chapter 25
Chapter 26. The vi Editor
The vi Editor
Regular Expression Words-of-Caution
Expressions Are Strings and Wildcards
Manual Pages for Commands Used in Chapter 26
Chapter 27. Introduction to the KornShell, C Shell, and Bash
Different Shells
Introduction to KornShell
Introduction to the C Shell
Introduction to Bash
Chapter 28. Introduction to Shell Programming
Shell Programming
Steps to Create Shell Programs
ksh Programming
An HP-UX Logical Volume Manager Shell Program
C Shell Programming
Manual Pages for Commands Used in Chapter 28
Appendix A. Select HP-UX Online Manual Pages
Manual Pages for Commands
awk
boot
crontab
find
grep
hpux
ifconfig
ignite
inittab
iostat
isl
netstat
pdc
ps
sed
shutdown
"sw" command summaries
top
vmstat
Appendix B. Virtual Partitions Online Manual Pages
Man Pages
vparboot
vparcreate
vparmodify
vparremove
vparreset
vparresources

vparstatus
vpartition
vparutil
LICENSE AGREEMENT AND LIMITED WARRANTY
ACKNOWLEDGMENT
About the CD-ROM
Technical Support
Index

Copyright

Library of Congress Cataloging-in-Publication Data

Poniatowski, Marty.

HP-UX 11i system administration handbook and toolkit / Marty Poniatowski.--2nd ed.

p. cm.

Includes index.

ISBN 0-13-101883-3 (pbk.)

1. HP-UX. 2. Operating systems (Computers) I. Title.

QA76.76.O63P647 2003

005.4'469--dc21 2002193143

Editorial/production supervision: *Patti Guerrieri*

Cover design director: *Jerry Votta*

Cover design: *Nina Scuderi*

Manufacturing buyer: *Maura Zaldivar*

Acquisitions editor: *Jill Harry*

Editorial assistant: *Brenda Mulligan*

Marketing manager: *Dan DePasquale*

Publisher, Hewlett-Packard Books: *Patricia Pekary*

© 2003 by Hewlett-Packard Company

Published by Pearson Education

Publishing as Prentice Hall Professional Technical Reference

Upper Saddle River, New Jersey 07458

Prentice Hall books are widely used by corporations and government agencies for training, marketing, and resale.

For information regarding corporate and government bulk discounts, please contact: Corporate and Government Sales (800) 382-3419 or corpsales@pearsontechgroup.com.

All product or company names mentioned herein are the trademarks or registered trademarks of their respective owners.

All rights reserved. No part of this book may be reproduced, in any form or by any means, without permission in writing from the publisher.

Printed in the United States of America

10 9 8 7 6 5 4 3 2 1

Pearson Education LTD.

Pearson Education Australia PTY, Limited

Pearson Education Singapore, Pte. Ltd.

Pearson Education North Asia Ltd.

Pearson Education Canada, Ltd.

Pearson Educación de Mexico, S.A. de C.V.

Pearson Education — Japan

Pearson Education Malaysia, Pte. Ltd.

[Team LiB]

PREVIOUS NEXT

Hewlett-Packard® Professional Books

STRATEGIC SOLUTION

Brans	Mobilize Your Enterprise: Achieving Competitive Advantage Through Wireless Technology
Caldwell	The Fast Track to Profit: An Insider's Guide to Exploiting the World's Best Internet Technologies
Carlson/Huck	Itanium Rising: Breaking Through Moore's Second Law of Computing Power
Fink	The Business and Economics of Linux and Open Source
Hayes-Roth/Amor	Radical Simplicity: Transforming Computers into Me-centric Appliances
Kincaid	Customer Relationship Management: Getting It Right!

HP-UX

Fernandez	Configuring CDE: The Common Desktop Environment
Madell	Disk and File Management Tasks on HP-UX
Olker	Optimizing NFS Performance: Tuning and Troubleshooting NFS on HP-UX Systems
Poniatowski	HP-UX 11i Virtual Partitions
Poniatowski	HP-UX 11i System Administration Handbook and Toolkit, Second Edition
Poniatowski	The HP-UX 11.x System Administration Handbook and Toolkit
Poniatowski	HP-UX 11.x System Administration "How To" Book
Poniatowski	HP-UX 10.x System Administration "How To" Book
Poniatowski	HP-UX System Administration Handbook and Toolkit
Poniatowski	Learning the HP-UX Operating System
Rehman	HP Certified: HP-UX System Administration
Sauers/Weygant	HP-UX Tuning and Performance: Concepts, Tools, and Methods
Weygant	Clusters for High Availability: A Primer of HP-UX Solutions, Second Edition
Wong	HP-UX 11i Security

UNIX, LINUX, WINDOWS, AND MPE I/X

Diercks	MPE i/X System Administration Handbook
Mosberger	IA-64 Linux Kernel
Poniatowski	UNIX User's Handbook, Second Edition
Roberts	UNIX and Windows 2000 Interoperability Guide
Stone/Symons	UNIX Fault Management

COMPUTER ARCHITECTURE

Kane	PA-RISC 2.0 Architecture
Markstein	IA-64 and Elementary Functions

NETWORKING/COMMUNICATIONS

Blommers	Architecting Enterprise Solutions with UNIX Networking
Blommers	OpenView Network Node Manager
Blommers	Practical Planning for Network Growth
Brans	Mobilize Your Enterprise: Achieving Competitive Advantage Through Wireless Technology
Cook	Building Enterprise Information Architecture: Reengineering Information Systems
Lucke	Designing and Implementing Computer Workgroups
Lund	Integrating UNIX and PC Network Operating Systems

SECURITY

Bruce	Security in Distributed Computing: Did You Lock the Door?
Pearson et al.	Trusted Computing Platforms: TCPA Technology in Context
Pipkin	Halting the Hacker, Second Edition
Pipkin	Information Security

WEB/INTERNET CONCEPTS AND PROGRAMMING

Amor	E-business (R)evolution, Second Edition
Apte/Mehta	UDDI: Building Registry-based Web Services Solutions

Apte/Mehta	Web Services: A Java Developer's Guide Using e-Speak
Greenberg	A Methodology for Developing and Deploying Internet and Intranet Solutions
Mowbrey/Werry	Online Communities: Commerce, Community Action, and the Virtual University
Tapadiya	.NET Programming

OTHER PROGRAMMING

Blinn	Portable Shell Programming: An Extensive Collection of Bourne Shell Examples
Caruso	Power Programming in HP OpenView: Developing CMIS Applications
Chaudhri	Object Databases in Practice
Chew	The Java/C++ Cross Reference Handbook
Grady	Practical Software Metrics for Project Management and Process Improvement
Grady	Software Metrics: Establishing a Company-wide Program
Grady	Successful Software Process Improvement
Lewis	The Art and Science of Smalltalk
Lichtenbelt	Introduction to Volume Rendering
Mellquist	SNMP++: An Object-Oriented Approach to Developing Network Management Applications
Mikkelsen	Practical Software Configuration Management: The Latenight Developer's Handbook
Norton	Thread Time: The Multi-Threaded Programming Guide
Tapadiya	COM+ Programming
Yuan	Windows 2000 GDI Programming

STORAGE

Thornburgh	Fibre Channel for Mass Storage
Thornburgh/Schoenborn	Storage Area Networks
Todman	Designing Data Warehouses: Supporting Customer Relationship Management

IT/IS

Missbach/Hoffman	SAP Hardware Solutions: Servers, Storage, and Networks for mySAP.com

IMAGE PROCESSING

Crane	A Simplified Approach to Image Processing: Classical and Modern Techniques in C
Day	The Color Scanning Handbook: Your Guide to Hewlett-Packard Scanjet Color Scanners
Gann	Desktop Scanners: Image Quality Evaluation

MORE BOOKS FROM MARTY PONIATOWSKI

HP-UX Virtual Partitions

UNIX User's Handbook, Second Edition

HP-UX 11.x System Administration Handbook and Toolkit

HP-UX 11.x System Administration "How To" Book, Second Edition

HP-UX System Administration Handbook and Toolkit

HP-UX 10.x System Administration "How To" Book

Learning the HP-UX Operating System

Preface

About The Book

Relevant URLs

Manual Pages Supplied with This Book

Acknowledgments

Conventions Used in this Book

About The Book

This update of *HP-UX 11i System Administration Handbook and Toolkit* includes a lot of new HP-UX 11i information. There are many new functions in 11i that have been introduced since the previous revision of this book was introduced such as: Virtual Partitions (vPars), hard partitions (nPartitions), the GNOME interface running on HP-UX, the Web-based **kcweb** kernel tuning tool, and many other advancements that are covered in this revision. At the same time; however, there are even more new 11i functions and architecture updates being introduced in the near future. Before this book is even on the shelf in bookstores I am preparing the next revision. HP-UX 11i and the hardware are advancing very quickly so there are a lot of important improvements for me to document that all HP-UX system administrators want to know about.

The HP-UX 11i System Administration Handbook and Toolkit is composed of following three sections:

- Section 1 - HP-UX System Administration
- Section 2 - HP-UX Partitions
- Section 3 - Introduction to UNIX® and HP-UX

The first section covers almost every aspect of HP-UX system administration for your day-to-day work. The Partitions section covers some of the latest and most important topics related to HP-UX partitions. Since partitions, either virtual, hard, or both, are available on every new HP-UX system I treat these important topics in a separate section. I have included Section 3 due to the many requests I have received to provide UNIX background for those readers with limited exposure to UNIX. There are new HP-UX system administrators and users every day who can use this introduction.

You'll be seeing additional updates to this book on a regular basis. There are many enhancements taking place such as HP systems based on the Intel® Itanium® architecture. I've just started to include such information in this book. Chapter 1, for instance, introduces the Extensible Firmware Interface (EFI). EFI is used on Itanium-based systems. It sits between the firmware for the IPF system and the operating system loader. There will be many more enhancements to come, so I'll be busy updating.

I hope you enjoy reading the book and learning the material as much as I did writing it.

Marty Poniatowski

marty.poniatowski@hp.com

Relevant URLs

There are many Web sites that can assist you in your HP-UX system administration endeavors. I have listed some of the more prominent HP-UX-related Web sites below as they existed at the time of this writing:

HP Developer and Solution Partner Portal (DSPP). This is a web site that has a lot of good software developer information and tools such as the Software Transition Kit (STK). You can download the STK to test your application when moving from one revision of HP-UX to another.

http://www.hp.com/dspp

The Extended Firmware Interface (EFI). This is firmware on Itanium-based systems that sits between the operating system and platform firmware:

http://www.intel.com/technology/efi/index.htm

IT Resource Center (ITRC) has valuable hardware, software, network, and other kinds of information for HP system administrators:

http://www.itrc.com

Technical documentation, including most all HP-UX documents:

http://www.docs.hp.com

Software depot home page for HP:

http://www.software.hp.com

HP Education Web Site

http://www.education.hp.com

vPar product information:

http://www.hp.com/go/servicecontrol

HP-UX documentation:

http://docs.hp.com/hpux/

Instant Capacity on Demand (iCOD):

http://www.hp.com/go/icod

The International Association of HP Computing Professionals:

http://www.interex.org

Information on Intel's Itanium Processor:

http://www.hp.com/go/itanium

Register name servers at:

http://www.icann.org/registrars/accredited-list.html

Excellent unsupported system administration scripts at:

ftp://contrib:9unsupp8@hprc.external.hp.com/sysadmin/

Ximian GNOME on HP-UX:

http://www.hp.com/workstations/support/software/hpux/gnome/index.html

Open Source and Other Sites of Interest

Information on Perl, including sites to download Perl:

http://www.perl.com

The Perl Journal:

http://www.tpj.com

Information about the GNOME desktop environment:

http://www.gnome.org

Public-domain software that has been ported to HP-UX:

http://hpux.connect.org.uk

Site devoted to managing and promoting open source:

http://www.opensource.org

Linux documentation site:

http://www.linuxdoc.org

Manual Pages Supplied with This Book

I am most grateful to Hewlett-Packard Company for having allowed me to include selected HP-UX manual pages in this book. I have received a great deal of positive feedback on the inclusion of manual pages in my previous books. Many readers find it helpful to have the manual pages in the book to refer to when reading when no system is available to check on a command (apparently a lot of people are reading my books while not at the office).

Only a limited number of manual pages are included in the book in Appendixes A and B. Most of the manual pages are related to either Virtual Partitions (vPars,) the boot process such as **isl**, or pages that I often refer to such as **awk** and **grep**

When a command is used for which there is an online manual page included in the book, the following information appears in the margin:



This is a "man page" block, which includes the man page icon, the command name, in this case **isl**, and the appendix in which the online manual page appears, in this case Appendix A.

Acknowledgments

There were too many people involved in helping me with this book to list each and every one. I have, therefore, decided to formally thank those who wrote sections of the book and those who took time to review it. I'm still not sure whether it takes more time to write something or review something that has been written to ensure that it is correct.

Duane Zitzner

Duane Zitzner is Executive Vice President of Personal Systems Group at Hewlett-Packard Company. Duane acted as the executive champion and sponsor of this book. His support was invaluable in helping get the resources necessary to complete this book.

The Author: Marty Poniatowski

Marty has been a Solution Architect with Hewlett-Packard Company for fifteen years in the New York area. He has worked with hundreds of Hewlett Packard customers in many industries, including Internet startups, financial services, sand manufacturing.

Marty has been widely published in computer industry trade publications. He has published over 50 articles on various computer-related topics. In addition to this book, he is the author of eleven other Prentice Hall books: Marty holds an M.S. in Information Systems from Polytechnic University (Brooklyn, NY), an M.S. in Management Engineering from the University of Bridgeport (Bridgeport, CT), and a B.S. in Electrical Engineering from Roger Williams University (Bristol, RI).

Greg Powell

Greg is a technical consultant with Hewlett-Packard. He provides education and consulting services in several technical fields, with specialization in the UNIX operating system, high availability computing, and network programming. Greg holds a Master's degree in Computer and Information Science from the University of Pennsylvania and Master's and Ph.D. degrees in Mathematics from Lehigh University (Bethlehem, PA).

Greg reviewed all of the ServiceGuard and Logical Volume Manager material in the book and was instrumental in ensuring the accuracy and thoroughness of those chapters.

Donna Kelly

Donna wrote the "Common Desktop Environment (CDE)" chapter of this book and reviewed almost every chapter in the book. Donna and I have collaborated on a number of projects. Donna has painstakingly reviewed many of my books for both technical accuracy and readability. Donna is both a technical expert in many operating systems and an excellent evaluator of the usefulness of a topic and the way it is covered. She not only ensures that the material is technically accurate, but she also makes certain that each topic is covered in a useful manner and that it is easy to read and comprehend.

Donna has been responsible for a number of computing environments of Hewlett Packard in Roseville, CA. She has experience with several operating systems, including HP-UX, MPE, and AS/400. Donna is also a Microsoft Certified Systems Engineer (MCSE).

Elizabeth Zinkann

Elizabeth reviewed all of the chapters in the UNIX section of the book and helped in many other areas of the development of this book.

Elizabeth is a logical Linux catalyst, a freelance technical editor, and an independent computer consultant. She was a Contributing Editor and Review Columnist for *Sys Admin Magazine* for 10 years. (Her most recent reviews have taken refuge at http://www.equillink.com.) Her articles have also appeared in *Performance Computing*, *Linux Magazine*, and *Network Administrator* magazines. As an independent computer consultant, she has built Linux servers, maintained Linux, Solaris, Macintosh, and Windows computers, programmed databases, and taught Linux, UNIX, computer hardware basics, and Internet essentials. In a former life, she also programmed communications features, including ISDN at AT&T Network Systems.

Brian Hackley

Brian supplied material on DNS and BIND (along with Corey Dow), NFS Performance Assessment (along with Pat Kilfoyle), and NIS background used in various sections throughout the book. Brian is a member of the Hewlett Packard North American Response Center UNIX Network Team (NETUX.)

Brian came to Hewlett Packard as a result of the HP-Apollo merger in 1989. Until 1993, he was an Offline Technical Marketing Support Engineer, responsible for NPI (New Product Introduction) for Apollo Domain network products, and later part of the HP-UX 9.0x and 10.0 project teams. Brian moved into the Chelmsford Response Center in 1993, where he worked first as a NETUX-Foreground, and then an NETUX-Background, Engineer. During this time, Brian began telecommuting full time for HP.

Brian worked for Sun Microsystems in 1995 and 1996, and returned to HP and re-joined the HP Response Center NETUX Team in early 1997. Assignments since his return include HP-UX Network Response Center Engineer for High Availability and Strategic Enterprise (HASE) customers, HP-UX Network Product Competency Center RCE, and most recently, a Network Business Recovery Specialist for Mission Critical Support.

Brian lives and works in Lexington Massachusetts, is married to Wendy Carter a Registered Pharmacist. They have a son Steve who is a Programmer at Merchant Internet Group in Michigan.

Corey Dow

Corey supplied material on DNS and BIND, collaborating with Brian Hackley. Corey is a Senior Engineer working for the North American Response Center UNIX Network Team (NETUX) in Roseville, California.

Corey began his career with HP in 1998, starting as a UNIX Networking Engineer in Atlanta, responding to customer calls regarding Internet Services and Lan Link protocols, before moving to the Roseville facility in early 1999. He has provided DNS consulting services to Fortune 500 companies such as Motorola, assisting in the migration of their nameservers to newer releases, and has attended classes taught by Cricket Lui, former HP hostmaster and author of several DNS books.

Corey enjoys spending free time with his wife, Valerie, living each day to the fullest.

Pat Kilfoyle

Pat (along with Brain Hackley) supplied material on DNS and BIND, NFS Performance Assessment, and NIS background used in various sections throughout the book.

Pat received a BSEE from the University of Alaska in 1983. He came to work for HP as a Customer Engineer in the Bellevue, WA office immediately afterward.

Pat then moved into a network support position, where he became involved in all aspects of network support including: NFS; X.25; interconnect devices such as bridges, routers, LANs, and WAN; and worked as a project manager on cabling and ifrastructure projects.

In 1991, Pat joined the HP Escalation Center supporting the full range of networking products, including hardware and software for the HP 9000, HP 3000, and Intel servers.

In 1994 Pat became involved in the support of networking software and support tools development. He specialized in socket programming, NFS, kernel transport, LAN driver code areas, and kernel performance support.

Pat is currently a Senior Systems Support Engineer on HP's Crisis Management Team. He is assigned to specific customers with HP's Business Continuity Support (BCS.)

Steven M. Wolff

Steven supplied the tear-out card for this book.

Steven came to HP as a result of HP's acquisition of Convex. He has been supporting High End Servers since 1989. He is currently in the Remote/E-Delivery Organization working for the Superdome Support Team supporting North American Superdome Customers.

Norbert H. Vicente

Norbert supplied the trial performance software on the CD-ROM in this and my other books.

Norbert is a Program Manager with Hewlett Packard's OpenView Software Division in Roseville, California. He began his career with HP in 1977, starting on the production assembly line building the HP 5061A Cesium frequency clocks used a calibration device of government contracts. Norbert transferred to the Roseville facility in 1985 and changed his career to MPE systems management. In 1991, there was a great need for Unix administration services across the site and a team of administrators were deployed to various labs on the Roseville facility to assist the networking and HP-UX administration requests.

Norbert influenced the site managers to begin utilization of NIS (Network Information Services) to increase the ratio of systems to administrators and to centralize HP-UX administration services for various labs across the site.

Every spare moment and weekend during the last two years have been spent in building a home for his wife Silvia and his two daughters, Aleika and Magali.

Permissions

There is a wealth of information inside HP in various forms that I was generously given permission to include in this book. In addition to the afore mentioned man pages and other items described under the biographies that appeared earlier, I would like to thank the following HP representatives for the permission they granted for material to be used in the book: Scott Eldridge for the technical "cheat sheets" included on the CD-ROM; and Alan Hymes for his white paper that is included on the CD-ROM. Although much of this information is available either on various HP Web sites or on your system itself, it is valuable to have all of this information as part of this book.

Reviewers

In all there were about 25 reviewers of this book. I'm not sure what makes someone agree to review a book. You don't get the glory of a contributing author, but it is just as much work. I would like to thank the many people who devoted a substantial amount of time to reviewing this book to ensure that I included topics important to new system administrators and covered those topics accurately.



Conventions Used in this Book

I don't use a lot of complex notations in the book. Here are a few simple conventions I've used to make the examples clear and the text easy to follow:

$ and #	The HP-UX command prompt. Every command issued in the book is preceded by a command prompt. Either one of these two will be used or a system name are usually used as prompts.
italics	Italics are used for variable values and when referring to functional areas and menu picks in the System Administration Manager (SAM).
bold and " "	Bold text is the information you would type, such as the command you issue after a prompt or the information you type when running a script. Sometimes information you would type is also referred to in the text explaining it, and the typed information may appear in quotes.
<----	When selections have to be made, this convention indicates the one chosen for the purposes of the example.
[ ]	Brackets indicate optional items and command descriptions.
{ }	Curly braces indicate a list from which you must choose.
|	A vertical bar separates items in a list of choices.
<Enter>	Indicates the "Enter" key has been pressed on the keyboard. Sometimes <Return> is used to indicate the return key has been pressed.

One additional convention is that used for command formats. I don't use command formats more than I have to because I could never do as thorough a job of describing commands as the HP-UX manual pages. The manual pages go into detail on all HP-UX commands. Here is the format I use when I cover commands:

```
form 1   command [option(s)] [arg(s)]
form 2   command [option(s)] [arg(s)]
form n   command [option(s)] [arg(s)]
```

I try not to get carried away with detail when covering a command, but there are sometimes many components that must be covered in order to understand a command. Here is a brief description of the components listed above:

form # - There are sometimes many forms of a command. If there is more than one form of a command that requires explanation, then I will show more than one form.

command - The name of the executable.

option(s) - Several options may appear across a command line.

cmd_arg(s) - Command arguments such as path name.



Section 1: HP-UX System Administration

Chapter 1. Booting (PDC, ISL, hpux), Secure Web Console, GSP Configuration, vPars Booting

Chapter 2. Installing HP-UX

Chapter 3. Logical Volume Manager and Veritas Volume Manager

Chapter 4. Building an HP-UX Kernel

Chapter 5. Devices

Chapter 6. Backup

Chapter 7. System Startup and Shutdown Scripts

Chapter 8. Users and Groups

Chapter 9. Ignite-UX

Chapter 10. System Administration Manager (SAM) and ServiceControl Manager (SCM)

Chapter 11. Introduction to UNIX Performance Tools - Security

Chapter 12. Networking Background

Chapter 13. Configuring and Using Networking

Chapter 14. Common Desktop Environment (CDE) and GNOME

Chapter 15. CIFS/9000 and Samba



Chapter 1. Booting (PDC, ISL, hpux), Secure Web Console, GSP Configuration, vPars Booting

Background

Boot Process Overview

Boot Console Handler (BCH) & Processor Dependent Code (PDC)

Initial System Loader

HPUX Secondary System Loader (hpux)

Consoles

Configuring the Guardian Service Processor (GSP)

Virtual Partition Boot Process Overview

Extensible Firmware Interface (EFI)

Manual Pages for Commands Used in Chapter 1



Background

The following is a bullet list of topics I'll cover in this chapter:

- An overview of the boot process.
- Boot Control Handler (BCH) and Processor Dependent Code (PDC.)
- Initial System Load (ISL).
- Secondary System Load (hpux.)
- Consoles (Local, Remote, LAN, and Secure Web Console.)
- Secure Web Console.
- Guardian Service Processor (GSP.)
- Booting and boot states of Virtual Partitions (vPars.)
- Introduction to the Extensible Firmware Interface (EFI) on Itanium-based systems.
- There is a tear-out card included with the book with a lot of useful boot and firmware-related information.
- There are some man pages related to boot in Appendix A.

As you can see from the list there is a lot to cover related to booting in HP-UX systems. In the first few sections of this chapter, I want to give you an overview of the low-level boot and configuration of an HP system, setting up HP Secure Web Console, and then configuring the Guardian Service Processor (GSP). You might perform these in the opposite order in which I present them. I chose to cover boot first, however, because it is an area of much confusion for new system administrators. Not everyone has a Secure Web Console, so I decided to cover this second. Securing your Guardian Service Processor can be done after the system boots, so I cover this third. The next topic covered is booting Virtual Partitions (vPars). Since partitions in general, both vPars and hard partitions (nPartitions), have become a more important part of an HP-UX 11i environment, It is importanto cover booting partitions.

There is an excellent tear-out card that you can use as a quick reference for many boot-related commands. There is a lot to know related to boot and working with partitions, covered later in this book, that the quick reference card summarizes nicely.

The system used In the examples in this chapter is an rp54xx system. I intentionally used a simple system so we could concentrate on the boot process with a minimum amount of hardware-related messages. Reviewing the boot process in this chapter won't make you an HP Customer Engineer (CE) or take the place of any formal training; however, knowing a little of what is going on "pre-operating system" may be helpful. In addition, if you are an experienced UNIX system administrator but haven't before used an HP system, these sections will serve as an introduction to booting a system. Since booting varies from system to system, even within the HP system family, this is not a comprehensive study, but rather an overview.

There is a section at the end of this chapter introducing Extensible Firmware Interface (EFI). EFI is used on Itanium-based systems. It sits between the firmware for the IPF system and the operating system loader. The introduction to EFI is worth taking a look at even if you are using only Precision Architecture-based systems at this time. EFI is an addition to the boot process on IPF systems that you'll get a feel for in the introduction at the end of this chapter.

Boot Process Overview



The boot process on an HP-UX system can be reduced in its simplist form to three steps. I'll provide a quick description of the three steps and then we'll take a look at some example boot processes so that we can see these three steps in action. The following is a description of the three steps:



PDC

HP-UX systems come with firmware installed called Processor Dependent Code (PDC - man page pdc in Appendix A). After the system is powered on or the processor is RESET, the PDC runs self-test operations and initializes the processor. PDC also identifies the console path so it can provide messages and accept input. PDC would then begin the "autoboot" process unless you were to interrupt it during the 10-second interval that is supplied. If you interrupt the "autoboot" process, you can issue a variety of commands. The interface to PDC commands is called the Boot Console Handler (BCH). This is sometimes a point of confusion; that is, are we issuing PDC commands or BCH commands? The commands are normally described as PDC commands and the interface through which you execute them is the BCH.



ISL

The Initial System Loader (ISL - man page isl in Appendix A). is run after PDC. You would normally just run an "autoboot" sequence from ISL; however, there are a number of commands that you could run from the ISL prompt.



hpux

The hpux utility manages loading the HP-UX kernel and gives control to the kernel. ISL can have hpux run an "autoexecute" file, or commands can be given interactively. In most situations, you would just want to automatically boot the system; however, we'll take a look at some of the hpux commands you can execute. This is sometimes called the Secondary System Loader (SSL).

I find that there is a lot of confusion related to the boot process for new system administrators. To begin with, there is not much documentation that comes with new systems related to HP system boot. Second, without any background on the boot process, it is difficult to determine at which phase of the boot your system is at any given time. Table 1-1 shows some system states and the corresponding prompts that you can expect for each state in roughly the order you might see them at the time of system boot:

Table 1-1. System Boot Prompts

Boot State of System	Prompt
Boot Console Handler (BCH) Seen when you discontinue boot within 10 seconds. Used to perform PDC-related work.	`Main Menu: Enter command or menu >`
Initial System Loader (ISL). Seen after PDC-related work.	`ISL> ?`
hpux Prompt varies. You usually issue **hpux** command at ISL prompt to boot. This	Varies depending on the state of the system.

is sometimes called the Secondary System Loader (SSL).	
Guardian Service Processor (GSP) Seen when you type ^b (control b) to get access to GSP.	GSP>



There is only one way to describe the boot process and that is through example. The boot of a system with minimal hardware will be covered in the upcoming sections. I choose the boot of a system with minimal hardware so as not to get bogged down in hardware-related details, but to instead focus on the boot process. The messages supplied as a result of booting this modest system will allow us to focus on the boot process rather than on the many hardware components. The boot process consists of mostly the same steps for any HP system, so you can apply this information to your system. It may be, however, that you have a much larger system with more components that will produce more lengthy boot messages.

[Team LiB]

PREVIOUS NEXT

Boot Console Handler (BCH) & Processor Dependent Code (PDC)



As mentioned earlier, HP-UX systems come with firmware installed called Processor Dependent Code (PDC). After the system is powered on or the processor is RESET, the PDC runs self test operations and initializes the processor. PDC also identifies the console path so that it can provide messages and accept input. PDC would then begin the "autoboot" process, unless you were to interrupt it during the 10-second interval that is supplied. If you interrupt the "autoboot" process, you can issue a variety of commands.

The following example shows booting an rp54xx system and the PDC-related messages we receive on the system. The first messages you see are a variety of self-test-related messages. The processor, memory, I/O and other components are run through a variety of tests. I abbreviated the list of messages in the example, wherever you see three dots, because the complete list of test results was too long to include in this book:

```
Value of TERM has been set to "vt100".
WARNING:  YOU ARE SUPERUSER !!1.00] (see /etc/issue)
Console Login: root
# reboot -h...checking for disk quotas
Shutdown at 17:41 (in 0 minutes) ckard Co.,  All Rights Reserved.
(c)Copyright 1979, 1980, 1983, 1985-1993 The Regents of the Univ. of California
(c)C*** FINAL System shutdown message from root@hp.serviceengine.com ***
(c)Copyright 1986-1992 Sun Microsystems, Inc.
System going down IMMEDIATELY Massachusetts Institute of Technology
(c)Copyright 1989-1993  The Open Software Foundation, Inc.
System shutdown time has arrivedent Corp.
(c)Copyright 1990 Motorola, Inc.
(c)Copyright 1990, 1991, 1992 Cornell University
(c)Copyright 1989-1991 The University of Maryland
Console reset done.arnegie Mellon University
(c)Copyright 1991-1997 Mentat, Inc.
Boot device reset done.ng Star Technologies, Inc.
(c)Copyright 1996 Progressive Systems, Inc.
(c)Copyright 1997 Isogon Corporation

System has halted
OK to turn off power or reset systemD RIGHTS LEGEND
UNLESS "WAIT for UPS to turn off power" message was printed above to
restrictions as set forth in sub-paragraph (c)(1)(ii) of the Rights in
********** VIRTUAL FRONT PANEL **********se in DFARS 252.227-7013.
System Boot detected
****************************************
LEDs:  RUN      ATTENTION     FAULT     REMOTE     POWER
       ON       FLASH         OFF       OFF        ON
                          Palo Alto, CA 94304 U.S.A.
platform                 config                    626F
processor                slave rendezvous          1C17ies are as set
processor                test                      1142
processor                test                      1100
processor                test                      1100
processor                test                      1100
processor                test                      1100
processor                test                      1100
processor                test                      1100
processor                test                      1100
processor                test                      1100
processor                test                      1100
PDH                      config                    322F
PDH                      test                      3149
PDH                      test                      3160
platform                 test                      616A
processor                test                      1146
processor                INIT                      1701
processor                INIT                      1701
processor                test                      1110
-
```

```
-
-
processor                 test                      111C
processor                 test                      111D
processor                 test                      111D
processor cache           test                      2111
processor cache           test                      2111
processor cache           test                      2112
processor cache           test                      2112
processor cache           test                      2113
processor cache           test                      2113
processor cache           test                      2121
processor cache           test                      2121
-
-
-
memory                    test                      71A4
memory                    test                      71A5
memory                    test                      71A5
memory                    test                      71A6
memory                    test                      71A6
memory                    config                    7210
I/O                       INIT                      8701
I/O                       test                      8118
I/O                       test                      8118
I/O                       INIT                      8701
I/O                       INIT                      8701
I/O                       INIT                      8701
I/O                       INIT                      8701
I/O                       INIT                      8701
I/O                       INIT                      8701
I/O                       INIT                      8701
I/O                       INIT                      8701
memory                    config                    7240
memory                    INIT                      7702
memory                    config                    7241
memory                    config                    7243
memory                    config                    72A0
memory                    test                      71A1
memory                    test                      71A2
-
-
-
processor                 test                      1120
processor                 slave rendezvous          1C40
processor                 test                      1142
processor                 test                      113B
platform                  test                      612A
I/O                       config                    8238

****************************************

************ EARLY BOOT VFP *************
End of early boot detected
****************************************

Firmware Version  39.46

Duplex Console IO Dependent Code (IODC) revision 1

------------------------------------------------------------------------------
   (c) Copyright 1995-1998, Hewlett-Packard Company, All rights reserved
------------------------------------------------------------------------------

  Processor   Speed            State           CoProcessor State  Cache Size
  Number                                       State              Inst    Data
  ---------  --------   --------------------  -----------------  ------------
      0      440  MHz   Active                Functional         512 KB   1 MB
      3      440  MHz   Idle                  Functional         512 KB   1 MB

  Central Bus Speed (in MHz)  :         82
  Available Memory            :    2097152  KB
  Good Memory Required        :      11468  KB
```

```
   Primary boot path:    0/0/1/1.2
   Alternate boot path:  0/0/2/0.2
   Console path:         0/0/4/0.0
   Keyboard path:        0/0/4/0.0


Processor is booting from first available device.

To discontinue, press any key within 10 seconds.
```



After the "early" boot is complete, we get a brief system summary, including the firmware revision on our system, and are then given the option to automatically boot off the primary path or press any key to stop the boot process. Under normal system operation, you would autoboot; however, in our case, we'll interrupt the boot process to see what commands are available in the PDC. When we interrupt the boot process, the following menu appears:

```
Boot terminated.


---- Main Menu -------------------------------------------------------------

     Command                          Description
     -------                          -----------
     BOot [PRI|ALT|<path>]            Boot from specified path
     PAth [PRI|ALT] [<path>]          Display or modify a path
     SEArch [DIsplay|IPL] [<path>]    Search for boot devices

     COnfiguration menu               Displays or sets boot values
     INformation menu                 Displays hardware information
     SERvice menu                     Displays service commands

     DIsplay                          Redisplay the current menu
     HElp [<menu>|<command>]          Display help for menu or command
     RESET                            Restart the system
----
Main Menu: Enter command or menu >
```

The interface to the PDC is called the Boot Console Handler (BCH). In our discussion the commands we are issuing are PDC commands and the interface through which we issue them is the BCH. Many of the commands in this menu are helpful to system administrators, but are sometimes ignored because they are pre-operating system and therefore usually overlooked.

We'll skip the first three commands for the time being and start by looking at the *COnfiguration* menu command by typing *co*. The uppercase "*CO*" in *COnfiguration* means that you could type just *co* as an abbeviation for configuration:

```
Main Menu: Enter command or menu > co

---- Configuration Menu -----------------------------------------------------

     Command                          Description
     -------                          -----------
     AUto [BOot|SEArch|STart] [ON|OFF] Display or set specified flag
     BootID [<proc>] [<bootid>]       Display or set Boot Identifier
     BootINfo                         Display boot-related information
     BootTimer [0 - 200]              Seconds allowed for boot attempt
     CPUconfig [<proc>] [ON|OFF]      Config/Deconfig processor
     DEfault                          Set the system to predefined values
     FAn [HI|NORmal]                  Display or change fan speed
     FastBoot [ON|OFF]                Display or set boot tests execution
     PAth [PRI|ALT] [<path>]          Display or modify a path
     SEArch [DIsplay|IPL] [<path>]    Search for boot devices
     TIme [c:y:m:d:h:m:[s]]           Read or set the real time clock in GMT

     BOot [PRI|ALT|<path>]            Boot from specified path
     DIsplay                          Redisplay the current menu
     HElp [<command>]                 Display help for specified command
     RESET                            Restart the system
     MAin                             Return to Main Menu
----
Configuration Menu: Enter command >
```

Under the *COnfiguration* menu are several useful commands. You can, for example, enable *FastBoot,* which gives control over tests run at boot time. We'll issue the **FastBoot** command to see how our system is currently set, but not change the value of *FastBoot*:

```
Configuration Menu: Enter command > FastBoot

   Fastboot:             OFF

Configuration Menu: Enter command >
```

Our system has *FastBoot off*, meaning that we'll run all available tests at boot time. If you would like to reduce system boot time and are willing to bypass boot tests you would turn *on FastBoot*.

There are a variety of other commands under the *COnfiguration* menu that you may want to issue, depending on your needs, such as *CPUconfig* to configure and deconfigure processors.

Let's now issue the *Main* command to get back to the *Main* menu and then *IN* to get access to the *INformation* menu:

```
Configuration Menu: Enter command > main

---- Main Menu ---------------------------------------------------------------

     Command                          Description
     -------                          -----------
     BOot [PRI|ALT|<path>]            Boot from specified path
     PAth [PRI|ALT] [<path>]          Display or modify a path
     SEArch [DIsplay|IPL] [<path>]    Search for boot devices

     COnfiguration menu               Displays or sets boot values
     INformation menu                 Displays hardware information
     SERvice menu                     Displays service commands

     DIsplay                          Redisplay the current menu
     HElp [<menu>|<command>]          Display help for menu or command
     RESET                            Restart the system
----
Main Menu: Enter command or menu > IN

---- Information Menu --------------------------------------------------------

     Command                          Description
     -------                          -----------
     ALL                              Display all system information
     BootINfo                         Display boot-related information
     CAche                            Display cache information
     ChipRevisions                    Display revisions of major VLSI
     COprocessor                      Display coprocessor information
     FRU                              Display FRU information
     FwrVersion                       Display firmware version
     IO                               Display I/O interface information
     LanAddress                       Display Core LAN station address
     MEmory                           Display memory information
     PRocessor                        Display processor information
     WArnings                         Display selftest warning messages

     BOot [PRI|ALT|<path>]            Boot from specified path
     DIsplay                          Redisplay the current menu
     HElp [<command>]                 Display help for specified command
     RESET                            Restart the system
     MAin                             Return to Main Menu
----
Information Menu: Enter command >
```

The *INformation* menu is a hidden gem for system administrators. Among the menu selections is one to display *ALL* system information, which produces the following report for our rp54xx system. I abbreviated this report at the point where you see the three dots near the end of the report:

```
Information Menu: Enter command > all

   Model:                    9000/800/L2000-44

PROCESSOR INFORMATION
```

```
                      HVERSION  SVERSION              Processor
  Processor   Speed     Model    Model/Op  CVERSION      State
  ---------  --------  --------  --------  --------  -------------
      0      440  MHz   0x05c4    0x0491     2.  4   Active
      3      440  MHz   0x05c4    0x0491     2.  4   Idle

  Central Bus Speed (in MHz)  :         82
  Software ID (dec)           :  143901527
  Software ID (hex)           : 0x0893c357
  Software Capability         : 0x01f0


COPROCESSOR INFORMATION

                   Coprocessor   Coprocessor        Coprocessor
       Processor      Model        Revision           State
      -----------   -----------   -----------     -----------------
           0        0x00000010             1        Functional
           3        0x00000010             1        Functional

CACHE INFORMATION

       Processor    Instruction Cache Size    Data Cache Size
      -----------   ----------------------   ---------------
           0               512 KB                    1 MB
           3               512 KB                    1 MB

MEMORY INFORMATION


  MEMORY STATUS TABLE (MB) (Current Boot Status)

Slot 0a  512M  Active
Slot 0b  512M  Active
Slot 1a  512M  Active
Slot 1b  512M  Active

Slot 2a   -
Slot 2b   -
Slot 3a   -
Slot 3b   -

Slot 4a   -
Slot 4b   -
Slot 5a   -
Slot 5b   -

Slot 6a   -
Slot 6b   -
Slot 7a   -
Slot 7b   -

Subtotal 2048M

   TOTAL =  2048 MB
            ---------


                    Memory Installation Guidelines
                    ------------------------------

- For DIMMs to work, both DIMMs in a slot pair (a/b) must be the same type.
    (Same part number = same type)

- For proper cooling, install DIMMs in the following order:
    0a/b 1a/b 2a/b 3a/b 4a/b 5a/b 6a/b 7a/b.


   Active, Installed Memory          :       2048  MB of SDRAM
   Deallocated Pages                 :          0  Pages
                                       -----------
   Available Memory                  :       2048  MB

   Good Memory Required by OS        :         12  MB
```

```
         Memory
   HVERSION  SVERSION
   -------- ----------
    0x0950  0x00000900

I/O MODULE INFORMATION

                        Path   Slot                          IODC
Type                    (dec)  Number    HVERSION  SVERSION  Vers
----                    -----  ------    --------  --------  ----
System bus adapter      0                0x5820    0xb10     0x0
Local bus adapter       0/0    1         0x7820    0xa00     0x0
Local bus adapter       0/1    6         0x7820    0xa00     0x0
Local bus adapter       0/2    8         0x7820    0xa00     0x0
Local bus adapter       0/3    10        0x7820    0xa00     0x0
Local bus adapter       0/4    12        0x7820    0xa00     0x0
Local bus adapter       0/5    7         0x7820    0xa00     0x0
Local bus adapter       0/6    9         0x7820    0xa00     0x0
Local bus adapter       0/7    11        0x7820    0xa00     0x0

PCI DEVICE INFORMATION
                             Path              Vendor  Device  Bus  Slot
Description                  (dec)               Id      Id     #    #
-----------                  -----              ----    ----   ---  ---
Ethernet cntlr               0/0/0/0           0x1011  0x19    0    1
SCSI bus cntlr               0/0/1/0           0x1000  0xb     0    1
SCSI bus cntlr               0/0/1/1           0x1000  0xb     0    1
SCSI bus cntlr               0/0/2/0           0x1000  0xf     0    1
SCSI bus cntlr               0/0/2/1           0x1000  0xf     0    1
Comp. ser cntlr              0/0/4/0           0x103c  0x1048  0    2
Comp. ser cntlr              0/0/5/0           0x103c  0x1048  0    2
Ethernet cntlr               0/3/0/0           0x1011  0x19    24   10
SCSI bus cntlr               0/4/0/0           0x1000  0xf     32   12
SCSI bus cntlr               0/4/0/1           0x1000  0xf     32   12
SCSI bus cntlr               0/7/0/0           0x1000  0xf     56   11
SCSI bus cntlr               0/7/0/1           0x1000  0xf     56   11

BOOT INFORMATION

   Processor           Boot ID
   ---------           -------
       0                  2
       3                  2

   Autoboot:             ON
   Autosearch:           ON
   Autostart:            ON
   Fastboot:             OFF

   Primary boot path:    intscsib.2
                         0/0/1/1.2
                         0/0/01/01.2    (hex)
   Alternate boot path:  intscsia.2
                         0/0/2/0.2
                         0/0/02/0.2     (hex)
   Console path:         0/0/4/0.0
                         0/0/04/0.0     (hex)
   Keyboard path:        0/0/4/0.0
                         0/0/04/0.0     (hex)

   LAN Station Address:  001083-fc9288

   Wed Apr  19 22:02:46 GMT 2000    (20:00:04:19:22:02:46)

FIRMWARE INFORMATION

   Firmware Version:          39.46

        Module              Revision
        ------              --------
        System Board        A443938
        PA 8500 CPU Module  2.4
        PA 8500 CPU Module  2.4

FRU INFORMATION
```

```
FRU Name:                 SYS_BD
Part Number:            A5191-60001
Serial Number:          A56405282277
Physical Location:      00ffff0001ffff69
Engineering Date Code: 3938
Art Work Revision:      A4
Scan Revision:
FRU Specific Info:      USS40130E3

FRU Name:                 IO_BP
Part Number:            A5191-60002
Serial Number:          52SCFK23WX
Physical Location:      00ffff0002ffff69
Engineering Date Code: 3942
Art Work Revision:      A2
Scan Revision:
FRU Specific Info:

FRU Name:       LAN_SCSI_CORE_IO
Part Number:            A5191-60011
Serial Number:            52SCFK28DE
Physical Location:      000000ffff01ff85
Engineering Date Code: 3933
Art Work Revision:      A5
Scan Revision:
FRU Specific Info:

-
-
-

FRU Name:               DIMM_512
Part Number:            A5798-60001
Serial Number:          A56E02093896
Physical Location:      0000ff00001bff74
Engineering Date Code: 3938
Art Work Revision:      A4
Scan Revision:
FRU Specific Info:

Information Menu: Enter command >
```

This report would be an excellent addition to a system administration notebook. Knowing such information as the firmware revision levels and the settings of the boot flags is valuable.



In the previous example, we used the BCH to issue the PDC commands *INformation* and then *ALL*. Both *INformation* and *ALL* are PDC commands issued through the BCH interface. Sometimes PDC and BCH are used interchangeably in documentation, however, knowing that BCH is the interface through which PDC commands are issued will usually serve you well when looking for information on specific commands.

Let's now go back to the *Main* menu and then look at the *SERvice* menu:

```
Service Menu: Enter command > main

---- Main Menu -------------------------------------------------------------

     Command                         Description
     -------                         -----------
     BOot [PRI|ALT|<path>]           Boot from specified path
     PAth [PRI|ALT] [<path>]         Display or modify a path
     SEArch [DIsplay|IPL] [<path>]   Search for boot devices

     COnfiguration menu              Displays or sets boot values
     INformation menu                Displays hardware information
     SERvice menu                    Displays service commands

     DIsplay                         Redisplay the current menu
     HElp [<menu>|<command>]         Display help for menu or command
     RESET                           Restart the system
----
Main Menu: Enter command or menu > ser
```

```
---- Service Menu ------------------------------------------------------------

     Command                          Description
     -------                          -----------
     CLEARPIM                         Clear (zero) the contents of PIM
     SCSI [option] [<path>] [<val>]   Display or set SCSI controller values
     MemRead <address> [<len>]        Read memory and I/O locations
     PDT [CLEAR]                      Display or clear the PDT
     PIM [<proc>] [HPMC|LPMC|TOC]     Display PIM information
     ProductNum <O|C> [<number>]      Display or set Product Number
     SELftests [ON|OFF]               Enable/disable self test execution

     BOot [PRI|ALT|<path>]            Boot from specified path
     DIsplay                          Redisplay the current menu
     HElp [<command>]                 Display help for specified command
     RESET                            Restart the system
     MAin                             Return to Main Menu
----
Service Menu: Enter command >
```

Among the options you have in *SERvice* menu is to manipulate the Processor Internal Memory, display SCSI controller values, and enable and disable self-tests. In the following example, we run *SCSI* to get information on our SCSI interfaces and then return to the *Main* menu:

```
Service Menu: Enter command > SCSI

Path (dec)     Initiator ID     SCSI Rate     Auto Term
------------   --------------   ----------    ---------------
0/0/1/0        7                Fast          Unknown
0/0/1/1        7                Ultra         Unknown
0/0/2/0        7                Ultra         Unknown
0/0/2/1        7                Ultra         Unknown

Service Menu: Enter command > main

---- Main Menu ---------------------------------------------------------------

     Command                          Description
     -------                          -----------
     BOot [PRI|ALT|<path>]            Boot from specified path
     PAth [PRI|ALT] [<path>]          Display or modify a path
     SEArch [DIsplay|IPL] [<path>]    Search for boot devices

     COnfiguration menu               Displays or sets boot values
     INformation menu                 Displays hardware information
     SERvice menu                     Displays service commands

     DIsplay                          Redisplay the current menu
     HElp [<menu>|<command>]          Display help for menu or command
     RESET                            Restart the system
----
Main Menu: Enter command or menu >
```



Now that we've looked at some of the PDC commands, let's get back to the process of booting by looking at the first three commands under *Main* menu.

BOot allows you to specify the path from which you'll boot your system. *PAth* allows you to display or modify the boot paths. *SEArch* will display boot paths. In the following example, we'll *SEArch* to show all potential boot devices on our rp54xx system, run *PAth* to display our existing boot path, and then specify the device from which we want to boot our system:

```
Service Menu: Enter command > main

---- Main Menu ---------------------------------------------------------------

     Command                          Description
     -------                          -----------
     BOot [PRI|ALT|<path>]            Boot from specified path
```

```
     PAth [PRI|ALT] [<path>]          Display or modify a path
     SEArch [DIsplay|IPL] [<path>]    Search for boot devices

     COnfiguration menu               Displays or sets boot values
     INformation menu                 Displays hardware information
     SERvice menu                     Displays service commands

     DIsplay                          Redisplay the current menu
     HElp [<menu>|<command>]          Display help for menu or command
     RESET                            Restart the system
----
Main Menu: Enter command or menu >
Main Menu: Enter command or menu > sea

Searching for potential boot device(s)
This may take several minutes.

To discontinue search, press any key (termination may not be immediate).


   Path#  Device Path (dec)  Device Path (mnem)  Device Type
   -----  -----------------  ------------------  -----------
   P0     0/0/1/1.2          intscsib.2          Random access media
   P1     0/0/2/0.2          intscsia.2          Random access media


Main Menu: Enter command or menu > pa

   Primary boot path:    intscsib.2
                         0/0/1/1.2
                         0/0/01/01.2    (hex)

   Alternate boot path:  intscsia.2
                         0/0/2/0.2
                         0/0/02/0.2     (hex)

   Console path:         0/0/4/0.0
                         0/0/04/0.0     (hex)

   Keyboard path:        0/0/4/0.0
                         0/0/04/0.0     (hex)

Main Menu: Enter command or menu > bo p0
Interact with IPL (Y, N, or Cancel)?> y
```

The *SEArch* command shows two potential boot device: in this case, our two internal disks on two different SCSI buses. The path of the two boot devices is composed of numbers separated by slashes (/), which indicate bus converters, and dots (.), which indicate cards, slot numbers, and addresses. You sometimes end up decoding these paths to figure out what boot devices map to what hardware devices on your system. In our case we have only the two internal disks, one on bus 1 and the other on bus 2.

The *PAth* command shows that the primary boot device is the disk on internal bus 1, and the alternate boot device is the disk on internal bus 2.

When I issued the *BOot* command in the previous example, I specified a device of *p0,* which corresponds to the disk on internal bus1 shown in the *SEArch* command results. I responded that I did indeed want to interact with IPL, which would normally not be the case when booting the system; however, I want to look briefly at IPL in the upcoming IPL section.



PDC commands issued through BCH are a mystery to many new system administrators. I covered enough in this section to get you comfortable enough on your system to look at the non-intrusive commands, that is, those that supply useful information without changing the settings on your system. Please be careful if you issue commands that change your configuration. Under

MAIN-CON-CPU

for instance you can configure and reconfigure processors. You would not want to experiment with this command unless your system is unused.

PDC Commands

Table 1-2 is a list of PDC commands available on an rp7400 system at the time of this writing. Although an rp54xx system was used in the examples in this chapter, the PDC for the rp54xx and rp7400 are similar. Keep in mind that the PDC commands for your system may be somewhat different from those listed. The PDC is updated occasionally, so the list for the rp7400 may also be somewhat different from that shown in Table 1-2.

Table 1-2. PDC Commands for rp7400

Command	Explanation
ALL	Display the collection of all information provided by other display commands typically resident in INFORMATION menu.
AUTO	Used to display or set status of AUTOBOOT, AUTOSEARCH, or AUTOSTART flags.
BOOT	Initiate boot sequence.
BOOTID	Display or modify boot *id* for the processors present.
BOOTINFO	Display PDC-level information about the configured parameters used for system boot.
BOOTTIMER	Sets a delay value in the system to wait for external mass storage devices to come on-line.
CACHE	Displays information about the cache memory portion of all installed processors.
CHASSISCODES	Displays a queue of the most recent chassis codes.
CHIPREVISIONS	Used to display the revisions of major Very Large Scale Integration (VLSI) in the system.
CLEARPIM	Used to clear (zero) the contents of the Processor Internal Memory (PIM).
CONFIGURATION	Used to enter the Configuration sub-menu.
COPROCESSOR	Displays information about all installed coprocessors.
CPUCONFIG	Allows the user to configure or deconfigure processors in the system.
DEFAULT	Used to set the system to pre-defined defaults.
DISPLAY	Used to redisplay the current menu.
FAN	Used to display or set the speed of system internal fans.
FASTBOOT	Used to display or set the fastboot flag.
FWRVERSION	Displays the revision of currently installed firmware.
HELP	Returns help information for the specified command, menu, or the system itself.
INFORMATION	Used to access the Information menu.
IO	Displays I/O interface on all I/O modules in the system.
LANADDRESS	Allows the user to display station address.
LANCONFIG	Used to configure the LAN card.
MAIN	User interface for PDC.
MEMORY	Displays memory information for total amount of physical memory as well as configured memory in a system.
MEMREAD	Used to read memory locations.
MONITOR	Allows the user to view and change the monitor type for graphic cards.

PATH	Used to set and/or display the system paths from Stable Storage.
PDT	Display or clear the Page Deallocation Table (PDT).
PIM	Displays Processor Internal Memory (PIM) Information.
PROCESSOR	Displays information about the processor(s) in the system.
RESET	Resets the machine state.
SEARCH	Search for boot devices in the system.
SECURE	Used to display or set the secure mode flag.
SERVICE	Allows the user to go to the Service menu.
TIME	Read or set the real time clock in GMT.
WARNINGS	Display any warning messages that may have resulted from the previous PDC self test execution.

[Team LiB]

PREVIOUS NEXT

Initial System Loader



As mentioned earlier, the Initial System Loader (ISL) is run after the PDC. You would normally just run an "autoboot" sequence from ISL; however, there are a number of commands that you could run from the ISL prompt.

Picking up where we left off in the previous example, we have chosen to *BOot* off device *p0* and interact with IPL as shown in the following example:

```
Main Menu: Enter command or menu > bo p0
Interact with IPL (Y, N, or Cancel)?> y

Booting...
Boot IO Dependent Code (IODC) revision 1


HARD Booted.

ISL Revision A.00.38  OCT 26, 1994

ISL> ?

    HELP          Help Facility
    LS            List ISL utilities
    AUTOBOOT      Set or clear autoboot flag in stable storage
    AUTOSEARCH    Set or clear autosearch flag in stable storage
    PRIMPATH      Modify primary boot path in stable storage
    ALTPATH       Modify alternate boot path in stable storage
    CONSPATH      Modify system console path in stable storage
    DISPLAY       Display boot and console paths in stable storage
    LSAUTOFL      List contents of autoboot file
    FASTSIZE      Sets or displays FASTSIZE
    800SUPPORT    Boots the s800 Support Kernel from the boot device
    700SUPPORT    Boot the s700 Support Kernel from the boot device
    READNVM       Displays contents of one word of NVM
    READSS        Displays contents of one word of stable storage
    LSBATCH       List contents of batch file
    BATCH         Execute commands in batch file
    LSEST         List contents of EST (Extended Self Test) file
    EST           Execute commands in EST (Extended Self Test) file

Enter 'LS' to see a list of the ISL utilities.

ISL>
```



Issuing a *?* produces a list of ISL commands that we could issue. Issuing the *DISPLAY* command shows the boot and console paths and *LS* lists the ISL utilities available, as shown in the following example:

```
ISL> display

    Fastsize value is 0000000F

    Autoboot is ON (enabled)

    Autosearch is ON (enabled)

    Primary boot path is 0/0/1/1.2.0.0.0.0.0
    Primary boot path is (hex) 0/0/1/1.2.0.0.0.0.0

    Alternate boot path is 0/0/2/0.2.0.0.0.0.0
```

```
    Alternate boot path is (hex) 0/0/2/0.2.0.0.0.0.0

    System console path is 0/0/4/0.0.0.0.0.0.0
    System console path is (hex) 0/0/4/0.0.0.0.0.0.0

ISL>  ls


    Utilities on this system are:

filename   type   start   size     created
=======================================================
ODE        -12960 584     1216     00/01/21 16:45:48
HPUX       -12928 3480    800      99/10/28 15:23:53

ISL>
```

DISPLAY produced the information we expected based on what we had seen in the previous section produced by PDC. *LS* produced two utilities available to us: *ODE* and *HPUX*. *ODE* is the Offline Diagnostics Environment. The following example shows listing the *ODE* utilities available on the system by running *ODE*, the *HELP* command, to see what commands are available, and *LS* to list the *ODE* utilities:

```
ISL> ode


**************************************************************************
******                                                              ******
******             Offline Diagnostic Environment                   ******
******                                                              ******
******  (C) Copyright Hewlett-Packard Co 1993-2000                  ******
******                     All Rights Reserved                      ******
******                                                              ******
******  HP shall not be liable for any damages resulting from the   ******
******  use of this program.                                        ******
******                                                              ******
******                TC  Version A.02.20                           ******
******                SysLib Version A.00.74                        ******
******                Loader Version A.00.59                        ******
******                Mapfile Version A.01.23                       ******
******                                                              ******
**************************************************************************

Type HELP for command information.
ODE> help

ODE Help

    Basic Commands
    --------------
    HELP -- Prints detailed information to the screen, when "help <command>"
            or "help <var>" is typed
    LS -- List modules available on boot medium
    <Module_Name> -- Load and initialize a module by typing its name
                     (For more help, type "help module_name")
    MENU -- Launch ODE's ease-of-use interface
    RUN -- Run a module (after setting desired environment variables)
    Control-Y|Control-C -- Abort an ODE command; pause a module run
    RESUME -- Restart a paused module
    DISPLOG -- After running a module, display contents of a log
    EXIT -- Return to next higher level prompt

    Environmental Variables
    -----------------------
    SHOWSTATE -- Display the value of the following environment variables:
        LOOP -- Run a test this many times
        ERRPRINT [ON|OFF] -- Print low-level error messages to console
                             (primarily for manufacturing use)
        ERRNUM [ON|OFF] -- Print one-line, numbered errors to the console

Continue ([y]/n)? y

        ERRPAUSE [ON|OFF] -- Pause module upon error detection
        ERRONLY [ON|OFF] -- Print ONLY error messages; disable non-error
                            and isolation message printing
```

```
        INFOPRINT [ON|OFF] -- Print informational messages to the console
        ISOPRINT [ON|OFF] -- Print fault isolation messages to the console
        ISOPAUSE [ON|OFF] -- Pause module when isolation message is generated
    LOGSIZE -- Set the size of a message log
    DEFAULT -- Reset environment variables to default state

ODE> ls

          Modules on this boot media are:

filename    type    size     created   description
-------------------------------------------------------------------------------
MAPPER2     TM      126      00/01/21  64 bit version of the system mapping ut
IOTEST2     TM      88       00/01/21  64 bit version that runs ROM-based self
PERFVER2    TM      124      00/01/21  64 bit version that runs ROM-based self

ODE>
```

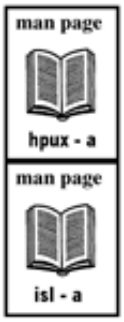
When we ran *ODE* in this example and issued *LS* the three utilities at the end of the example were listed. I always load *ODE* as a part of system installations to help in the event of a possible system hardware problem that an HP CE may need to diagnose.

Next we'll proceed with the boot process by running the *HPUX* utility:

[Team LiB]

PREVIOUS NEXT

HPUX Secondary System Loader (hpux)



As mentioned earlier, the *hpux* utility manages loading the HP-UX kernel and gives control to the kernel. ISL can have **hpux** run an "autoexecute" file, or commands can be given interactively. In most situations, you would just want to automatically boot the system, however, in our example so far, we have decided to interact with IPL in the interest of looking at some of the functionality in ISL. We proceed with the boot process by simply issuing the *HPUX* utility name as shown in the following example:

```
ISL> hpux

Boot
: disk(0/0/1/1.2.0.0.0.0.0;0)/stand/vmunix
7094272 + 849200 + 724128 start 0x241068


alloc_pdc_pages: Relocating PDC from 0xf0f0000000 to 0x7f9ab000.
gate64: sysvec_vaddr = 0xc0002000 for 1 pages
NOTICE: nfs3_link(): File system was registered at index 4.
NOTICE: autofs_link(): File system was registered at index 6.

    System Console is on the Built-In Serial Interface
Entering cifs_init...
Initialization finished successfully... slot is 8
Logical volume 64, 0x3 configured as ROOT
Logical volume 64, 0x2 configured as SWAP
Logical volume 64, 0x2 configured as DUMP
    Swap device table:  (start & size given in 512-byte blocks)
        entry 0 - major is 64, minor is 0x2; start = 0, size = 2097152
Starting the STREAMS daemons-phase 1
Checking root file system.
file system is clean - log replay is not required
Root check done.
Create STCP device files
Starting the STREAMS daemons-phase 2
    B2352B/9245XB HP-UX (B.11.00) #1: Wed Nov  5 22:38:19 PST 1997

Memory Information:
    physical page size = 4096 bytes, logical page size = 4096 bytes
    Physical: 2097152 Kbytes, lockable: 1574352 Kbytes, available: 1815496 Kbytes

/sbin/ioinitrc:
/sbin/bcheckrc:
Checking for LVM volume groups and Activating (if any exist)
Volume group "/dev/vg00" has been successfully changed.
vxfs fsck: sanity check: root file system OK (mounted read/write)
Checking hfs file systems
/sbin/fsclean: /dev/vg00/lvol1 (mounted) ok
HFS file systems are OK, not running fsck
Checking vxfs file systems
/dev/vg00/lvol8 :
vxfs fsck: sanity check: /dev/vg00/lvol8 OK
/dev/vg00/lvol3 :
vxfs fsck: sanity check: root file system OK (mounted read/write)
/dev/vg00/lvol4 :
vxfs fsck: sanity check: /dev/vg00/lvol4 OK
/dev/vg00/lvol5 :
vxfs fsck: sanity check: /dev/vg00/lvol5 OK
/dev/vg00/lvol6 :
vxfs fsck: sanity check: /dev/vg00/lvol6 OK
/dev/vg00/lvol7 :
vxfs fsck: sanity check: /dev/vg00/lvol7 OK
```

```
(c)Copyright 1983-1997 Hewlett-Packard Co.,  All Rights Reserved.
(c)Copyright 1979, 1980, 1983, 1985-1993 The Regents of the Univ. of California
(c)Copyright 1980, 1984, 1986 Novell, Inc.
(c)Copyright 1986-1992 Sun Microsystems, Inc.
(c)Copyright 1985, 1986, 1988 Massachusetts Institute of Technology
(c)Copyright 1989-1993  The Open Software Foundation, Inc.
(c)Copyright 1986 Digital Equipment Corp.
(c)Copyright 1990 Motorola, Inc.
(c)Copyright 1990, 1991, 1992 Cornell University
(c)Copyright 1989-1991 The University of Maryland
(c)Copyright 1988 Carnegie Mellon University
(c)Copyright 1991-1997 Mentat, Inc.
(c)Copyright 1996 Morning Star Technologies, Inc.
(c)Copyright 1996 Progressive Systems, Inc.
(c)Copyright 1997 Isogon Corporation


                         RESTRICTED RIGHTS LEGEND
Use, duplication, or disclosure by the U.S. Government is subject to
restrictions as set forth in sub-paragraph (c)(1)(ii) of the Rights in
Technical Data and Computer Software clause in DFARS 252.227-7013.


                         Hewlett-Packard Company
                         3000 Hanover Street
                         Palo Alto, CA 94304 U.S.A.

Rights for non-DOD U.S. Government Departments and Agencies are as set
forth in FAR 52.227-19(c)(1,2).

.
.
.
Console Login:
```

I abbreviated this listing where the three dots appear before the *Console Login:* prompt.

We have a number of options we can issue with the *HPUX* utility. Table 1-3 is a list of examples of some common *HPUX* utility booting options, some of which are from the man page in Appendix A.

Table 1-3. hpux Examples

Command (all at *ISL>* prompt)	Comments
Automatic Boot	No interaction - autoboot sequence.
hpux -is	Bring up system at run level *s* for single user mode.
hpux	Default boot sequence from **autoexecute** normally object file is **/stand/vmuinx**.
hpux vmunix.test	Boot object file **vmunix.test**.
hpux (52.5.0.0) /stand/vmunix	Boot from **/stand/vmunix** on the disk at the path *52.5.0.0.*
hpux lan(32) /stand/vmunix	Boot from LAN.
hpux -v	Get HP-UX version numbers..
hpux ll /stand	List contents of **/stand** on root disk.

The part of the boot that takes place after the **hpux** command is issued is in the four following parts:

1. HP-UX initializes the system hardware and devices - The HP-UX kernel locates and initializes system hardware such as memory, I/O busses and devices, and so on. Kernel device drivers are associated

with I/O devices at this time. You see many messages fly-by on the system console as this process is taking place.

2. HP-UX kernel data structures are created and initialized - There are many tables for system processes and memory, file systems, and so on that are created. You also see the status of this part of the boot on the system console.

3. HP-UX searches for the root file system - The base file system contains critical system files and is usually found on the disk from which HP-UX boots. Many commands are run as part of this process.

4. HP-UX starts the init process called /**sbin/init** - 

The init process, which has an ID of one, starts all other processes on the system. The init process reads **/etc/inittab** for direction. There is a detailed description of the contents of **/etc/inittab** in Chapter 7.



I encourage you to issue some of the PDC, ISL, and hpux utility commands covered earlier, especially those that do not modify your system in any way but only provide information about the system.

The next section covers configuring the Secure Web Console.



Consoles

You have a choice concerning the type of console to use with your HP system. Although the console hardware varies among different HP system models, there are usually the following ports available:

Local Console	The local console port is a serial port to which you would connect a terminal, or use a terminal emulator, to get direct console access. This is usually the default port to which console messages are sent.
Remote Console	This port is used for a modem connection to the console.
LAN Console	This console port allows a console connection using a LAN. The networking stack is implemented in firmware on the GSP. At the time of this writing this is a 10Base-TX connection and not a 100 MBit connection. The functionality of accessing the console over the network is nearly identical to using the local console port. this port has its own IP address, subnet mask, and gateway.
Secure Web Console	The Secure Web Console is a card inside your HP-UX system that allows you to use your browser as a console. On older systems this was an external device that connects to the HP system console port and provided the same Web server functionality.

There is not much configuration for the local console other than connecting a terminal. The remote console required modem configuration. The LAN console is configured through the GSP interface as we'll see in the next example. The following listing shows several steps we'll perform to boot an nPartition in the upcoming example. This example is from Chapter 17 which covers nPartitions. I'll issue a variety of commands to give you an idea of some steps you can perform during the boot process process at various prompts. We'll do the following:

- Go from the Management Processor to the *Command Menu* with **cm**.
- List the default customer LAN settings of the Managment Processor with **ls**.
- Change the customer LAN settings with **lc**.
- List the new LAN settings with **ls**.

```
Main Menu: Enter command or menu >

    GSP MAIN MENU:

         CO: Consoles
        VFP: Virtual Front Panel
         CM: Command Menu
         CL: Console Logs
         SL: Show chassis Logs
         HE: Help
          X: Exit Connection

GSP> cm                                          <-- Go to Command Menu

               Enter HE to get a list of available commands


GSP:CM> ls                                       <-- List customer LAN

Current configuration of GSP customer LAN interface
  MAC address : 00:10:83:fd:77:f7
  IP address  : 192.168.1.1     0xc0a80101
  Name        : gsp0
  Subnet mask : 255.255.255.0   0xffffff00
```

```
  Gateway     : 192.168.1.1     0xc0a80101
  Status      : UP and RUNNING


Current configuration of GSP private LAN interface
  MAC address : 00:a0:f0:00:ce:1f
  IP address  : 192.168.2.11    0xc0a8020b
  Name        : priv-01
  Subnet mask : 255.255.255.0   0xffffff00
  Gateway     : 192.168.2.11    0xc0a8020b
  Status      : UP and RUNNING


GSP:CM> lc                                          <-- Modify customer LAN

This command modifies the LAN parameters.

Current configuration of GSP customer LAN interface
  MAC address : 00:10:83:fd:77:f7
  IP address  : 192.168.1.1     0xc0a80101
  Name        : gsp0
  Subnet mask : 255.255.255.0   0xffffff00
  Gateway     : 192.168.1.1     0xc0a80101
  Status      : UP and RUNNING


    Do you want to modify the configuration for the customer LAN? (Y/[N]) y

    Current IP Address is: 192.168.1.1
    Do you want to modify it? (Y/[N]) y
    Enter new IP Address: 157.235.10.249
    New IP Address will be: 157.235.10.249
    Please confirm (Y/[N]) y

    -> IP Address will be updated.

    Current GSP Network Name is: gsp0
    Do you want to modify it? (Y/[N]) n

    Current Subnet Mask is: 255.255.255.0
    Do you want to modify it? (Y/[N]) n

    Current Gateway is: 192.168.1.1
    Do you want to modify it? (Y/[N]) (Default will be IP address.) 157.235.10.2

    Current Gateway is: 192.168.1.1
    Do you want to modify it? (Y/[N]) (Default will be IP address.) y
    Enter new Gateway: 157.235.10.246
    New Gateway will be: 157.235.10.246
    Please confirm (Y/[N]) y

    -> Gateway will be updated.

Current configuration of GSP private LAN interface
  MAC address : 00:a0:f0:00:ce:1f
  IP address  : 192.168.2.11    0xc0a8020b
  Name        : priv-01
  Subnet mask : 255.255.255.0   0xffffff00
  Gateway     : 192.168.2.11    0xc0a8020b
  Status      : UP and RUNNING


    Do you want to modify the configuration for the private LAN? (Y/[N]) n


    -> Parameters have been updated.
```

You can see in this example that we changed the default LAN console information using the GSP.

Next we'll configure the Secure Web Console.

Secure Web Console

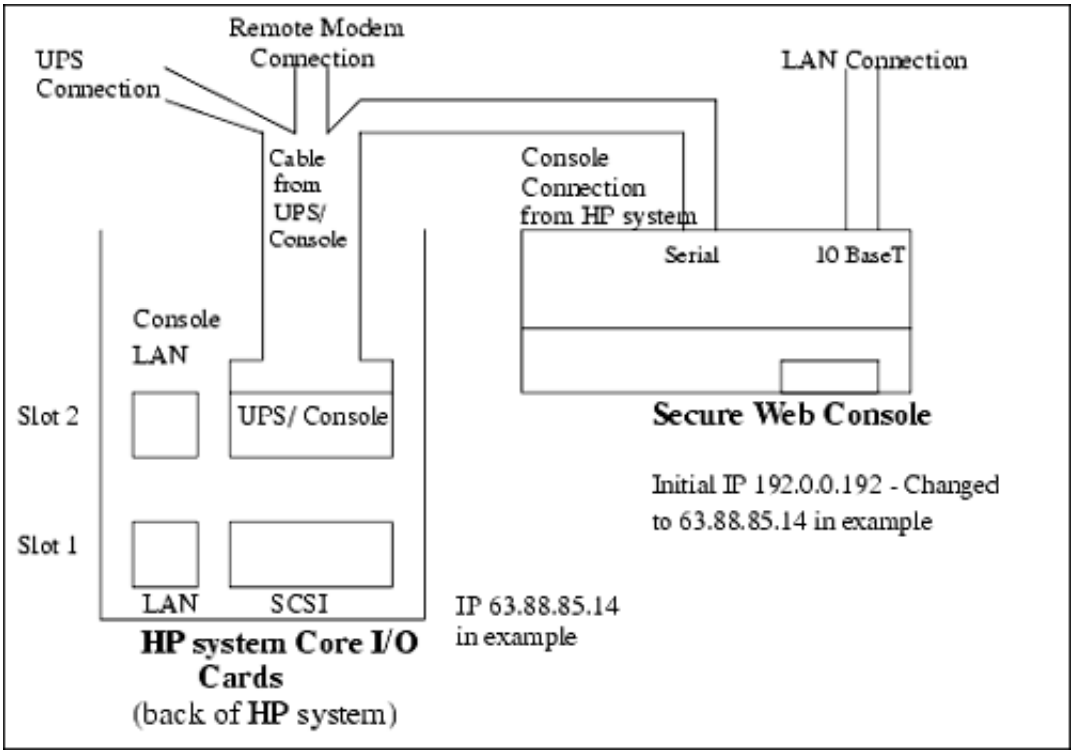
You have a choice concerning the type of console to use with your HP-UX. You can connect a terminal to the console port or use a browser and the Secure Web Console (there is also a LAN console port available on systems as another alternative). There is also a console consolidation solution that supports hundreds of consoles that I won't cover here. The Secure Web Console is a card in your HP-UX system that allows you to use your browser as a console. This means that from one system running a browser, you can open up several windows that are the console screens for different HP-UX systems. This obviates the need to walk from console to console when you want to perform system administration functions - you can sit at one system running a browser and adminstrate many systems. On older systems there was an external Secure Web Console device that connects to the HP-UX console port and provides the same Web server. I'll show a diagram shortly for connecting the old device.

In this section I'll cover configuring a Secure Web Console and include several screen shots so you can decide whether you prefer the Secure Web Console or a dedicated terminal as a console.

Connecting and Configuring the Secure Web Console

The new Secure Web Console is a card in your HP-UX system. The old Secure Web Console is a device that is connected to the console port on your HP-UX system and to your LAN. On an rp54xx system, for instance, slot 2 has in it a Core I/O card that includes a LAN console port and a connector for the UPS and console - the serial port on the Secure Web Console is plugged into this console cable. There is also a networking port on the Secure Web Console so you can connect it to your LAN. Figure 1-1 depicts the general setup of the Secure Web Console:

Figure 1-1. External Secure Web Console Connectivity on rp54xx Used in Upcoming Example



Keep in mind that the HP system connections on your system may differ from that shown in the figure. The cable connected to the UPS/Console port turns into the three connections shown in Figure 1-1. The Secure Web Console connects to the right-most of the three connectors when facing the back of the computer on my system. Its other connection is a networking connection to the LAN. The Secure Web Console also has its own power source so you can connect to it from your browser even when the computer is powered off.

The Secure Web Console has a default IP address of 192.0.0.192. This address is assigned so that you can connect to the Secure Web Console upon installation and reconfigure it.

With the hardware connections complete, we'll proceed to setup the system running a browser and connect to the Secure Web Console using the following steps:

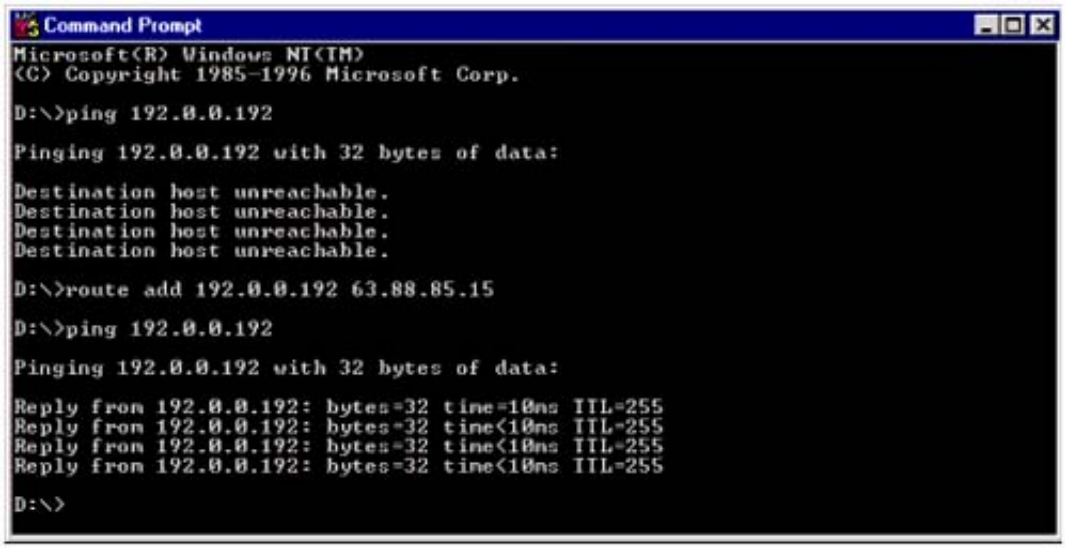
Disable Proxies	Assuming that you are running a supported version of a browser, you want to "Disable Proxies" or use "No Proxies" until the Secure Web Console configuration is complete.
Add a Route	Use the following command to add a route from the computer where your browser is running to the Secure Web console: **route add 192.0.0.192 63.88.95.15**

In this example the computer on which I'm running my browser has an IP address of 63.88.95.15

Check connection	Use the **ping** command to check the connection between the computer running the browser and the default IP address of the Secure Web Console: **ping 192.0.0.192**
Use **arp**	If you did not receive a response using **ping** you may have to use **arp** as shown below for a PC and UNIX system: PC: **arp -s 192.0.0.192 00-10-83-fa-3f-11** UNIX: **arp -s 192.0.0.192 00: 10: 83: fa: 3f: 11** I have not had to use the **arp** command in my configurations, however, the document at the URL earlier mentioned recommends the **arp** command if indeed your **ping** fails. Re-issue the **ping** command after using **arp** to check your connection. The *-s* option is for a static **arp**.
Web Browser	With a successful **ping** you can now access the Secure Web Console through your browser.

Figure 1-2 shows the steps I performed. You can see **route** and **ping** on my PC at the command line to get access to the Secure Web Console:

Figure 1-2. route and ping with Secure Web Console



This screen shot does not show the "disable proxies" step I performed prior to the commands issued at the prompt.

Note: I sometimes configure the Secure Web Console by connecting a crossover LAN cable from the Secure Web Console directly to my laptop computer. After I have completed the initial

configuration steps I then connect the Secure Web Console to the LAN and use a desktop system to access it with a browser. The connection to my portable is only a temporary connection in order to configure the Secure Web Console with its final IP address.

With a connection having been established and the browser properly configured on my computer (the proxy server was earlier disabled), I can use the browser to connect to the Web server on the Secure Web Console. You must be on the same subnet for this to work in most situations. Figure 1-3 is a screen shot showing that I specified the pre-configured IP address of the Secure Web Console with http://192.0.0.192:

Figure 1-3. Initial Secure Web Console Screen



The "Welcome" screen confirms that we performed the necessary configuration to make our initial connection to the Secure Web Console. After you select "OK" on this screen, you can proceed with the configuration of the Secure Web Console.

The next screen in Figure 1-4 shows the initial information related to the Secure Web Console for which you will be prompted when you select "OK". This information is for an administrator of the Secure Web Console, not for the HP system. In the upcoming examples, I zoom in so that you can see the Secure Web Console information more clearly. The browser information is not shown in the figures, but it is insignificant as far as the Secure Web Console configuration is concerned from this point foward.

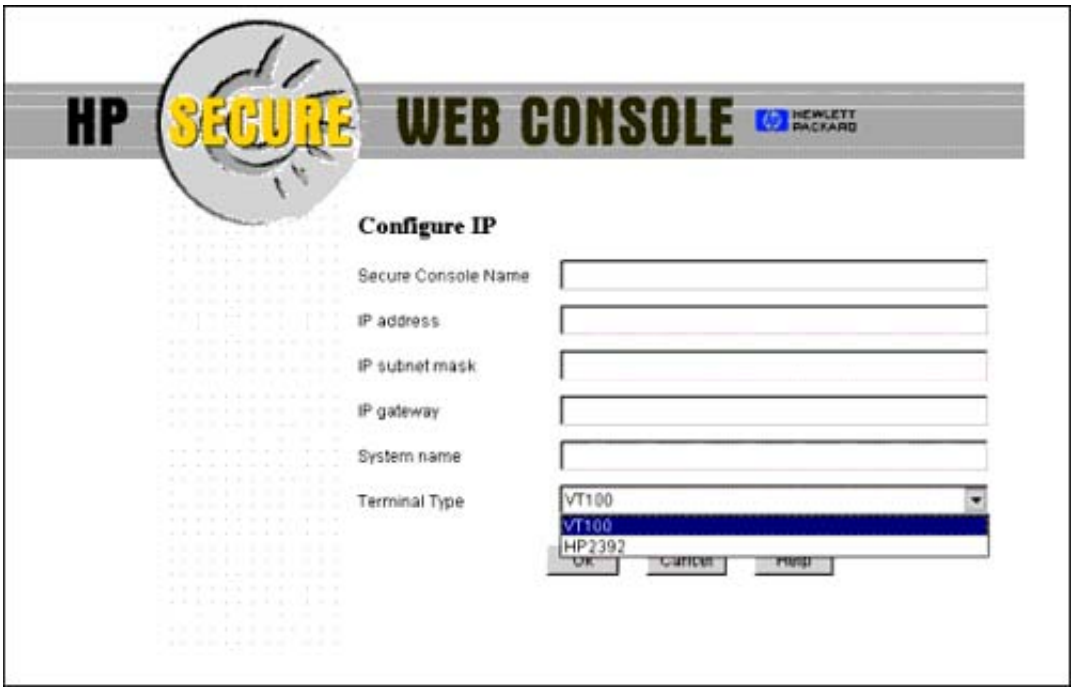
Figure 1-4. Secure Web Console Screen for First Administrator Account



After adding all of this information, you have configured the initial administrator account. As we'll see later, you can add subsequent administrator and operator accounts to the Secure Web Console.

The next screen, in Figure 1-5, shows the network-related configuration we'll make to the Secure Web Console. Since it came with a pre-configured IP address, we'll now assign the dedicated IP address for the device.

Figure 1-5. Secure Web Console Screen to Configure Final IP Address



In Figure 1-5 I show the two types of terminals that you can select. In this "Configure IP" screen, you'll

enter both a name for the Secure Web Console and the computer to which it is connected. All of the IP-related information is entered in this screen as well. The Secure Web Console requires a dedicated IP address because you connect directly to it to make a console connection to the HP system.

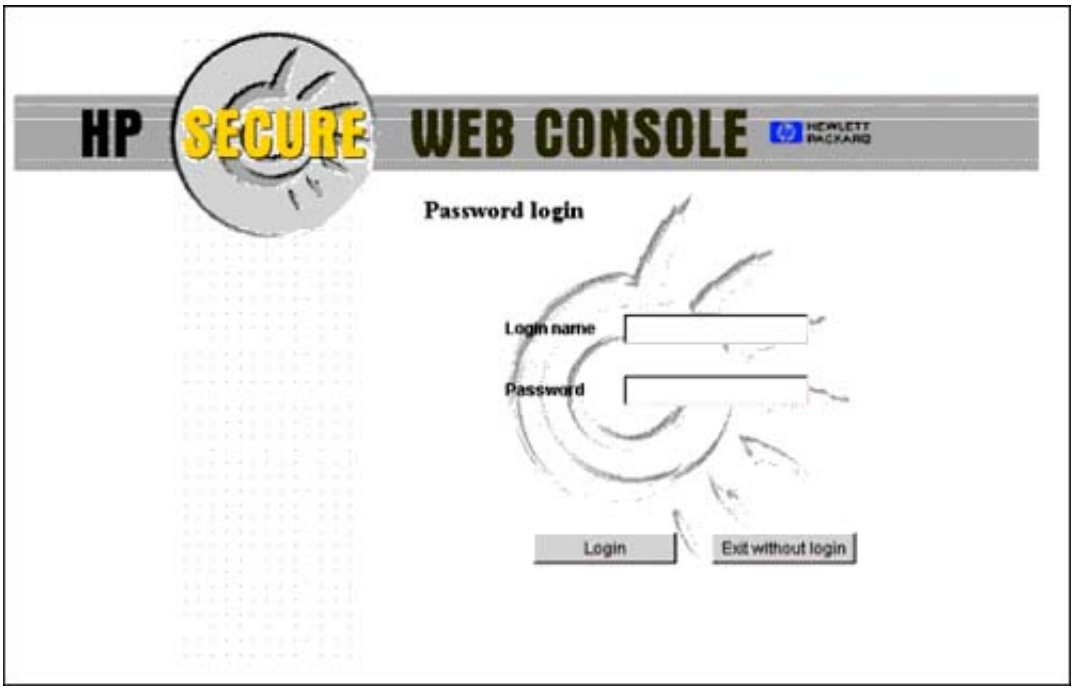
When all of the information has been added, we select "OK" on this screen and are shown the following message in Figure 1-6:

Figure 1-6. Initial Secure Web Console Setup Complete



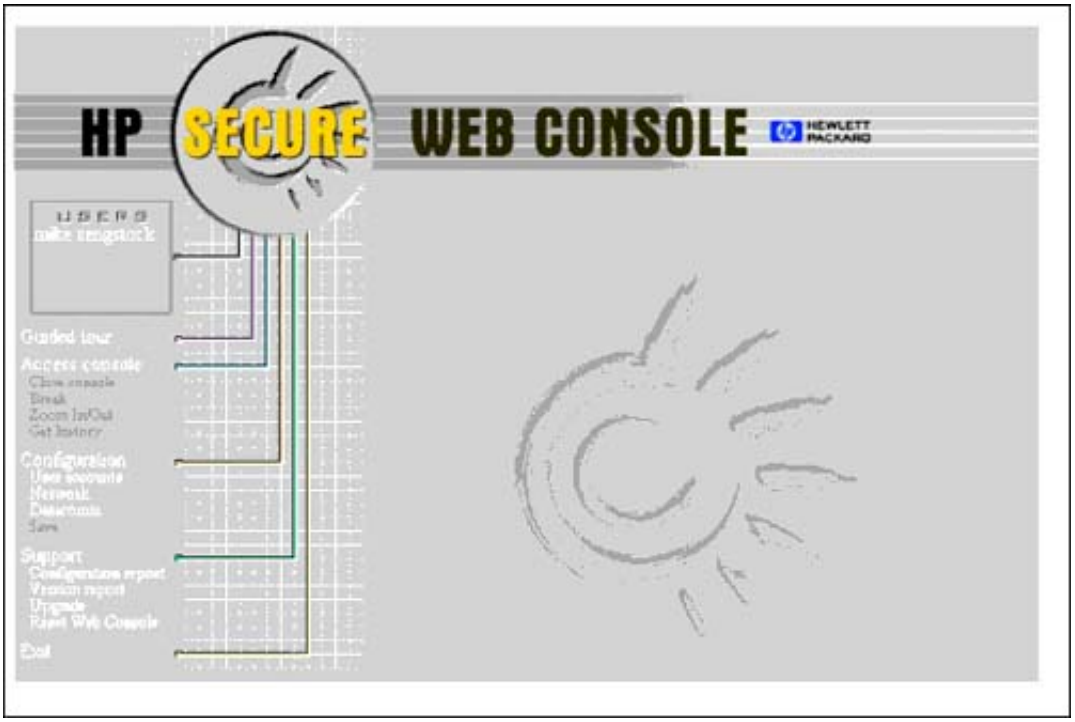
After we select "OK" in Figure 1-6, we can reconnect to the Secure Web Console using the IP address we assigned to it during the configuration. In our case, we would use *http://15.88.85.14* to reconnect, as shown in Figure 1-7:

Figure 1-7. Secure Web Console Login Screen



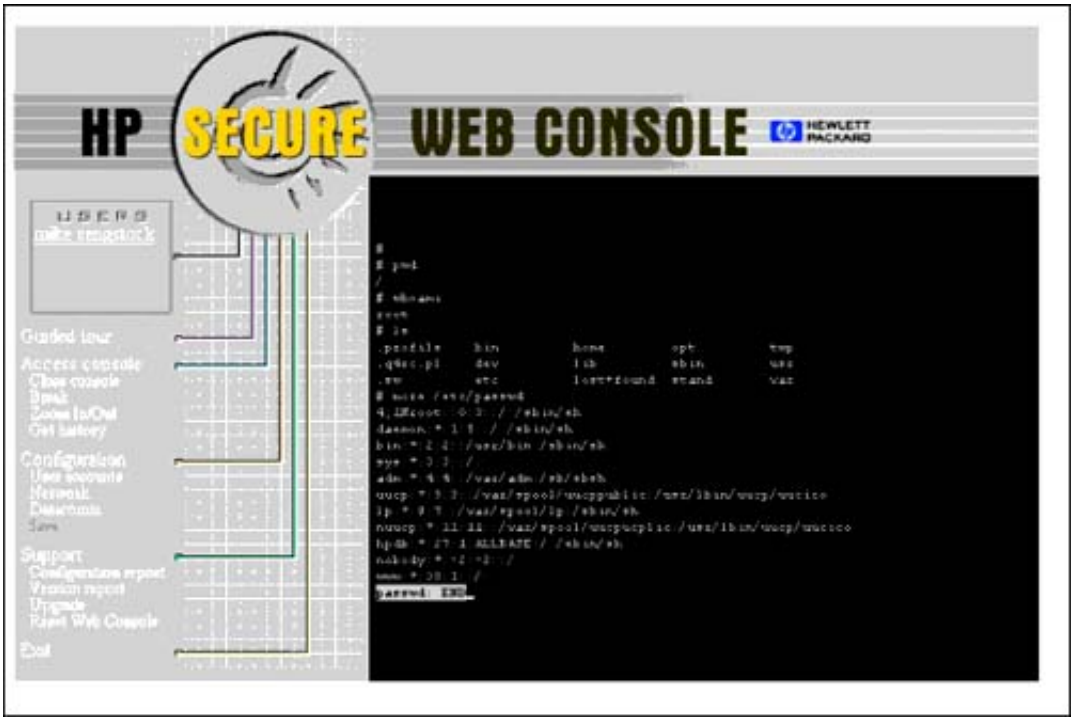
We log in with the name and password we earlier defined for the Secure Web Console administrator and are then connected as shown in Figure 1-8:

Figure 1-8. Secure Web Console Menu



At this point, you can select from a number of Secure Web Console-related functions to perform, such as configuring additional users, or you can access the console on your HP system, which is the purpose of connecting the device. The *Access console* selection gives us the console of the HP system as shown in Figure 1-9:

Figure 1-9. Initial Secure Web Console Screen



Notice that we are logged in as root when we select *Access console*. You could use the Secure Web Console menu to configure additional administrators and operators (we configured only one administrator in our example and no operators) to give multiple users access to your HP system through the Secure Web Console.

The Secure Web Console can be a good alternative to using a dedicated console on each HP-UX system. I wanted to provide you with the basic information for configuring the Secure Web Console so that you know how this device works.

[Team LiB]

PREVIOUS NEXT



Configuring the Guardian Service Processor (GSP)

The Guardian Service Processor (GSP) is a built-in processor on most HP systems that can be used for either local or remote system administration functions. I won't cover the use of the GSP, but I will cover its initial configuration. The initial configuration is important because the first person to gain access to the GSP before it has been configured is a GSP administrator by default.

You gain access to the GSP with ^b (control b). If the GSP has not been configured, anyone who gets access to the console can type ^b and gain access to the system. I recommend you perform GSP configuration as soon as possible after installing your system.

The following example shows issuing a ^b, which results in the *GSP>* prompt, and issuing **he** for help:

```
Service Processor login:
Service Processor password:



             Hewlett-Packard Guardian Service Processor


             9000/800/L2000-44 System Name: uninitialized

GSP Host Name:  uninitialized
GSP>



GSP Host Name:  uninitialized
GSP> he


HE
      Hardware Revision 8  Firmware Revision  A.01.06  Aug  2 1999,11:38:47
AC : Alert Display Configuration       PC : Remote Power Control
AR : Config. Automatic System Restart  PS : Power Management Module status
CA : Configure serial port parameters  RP : Reset password configuration
CE : Log repair info in history buffer RS : System reset through RST signal
CL : Display console history           SDM: Set Display Mode (hex or text)
CO : Return to Console Mode            SE : Activate a system session
DC : Default configuration             SL : Display SPU status logs
DI : Disconnect remote or LAN console  SO : Security options & access control
DR : Disable remote or LAN console     SS : System's processor status
ER : Enable remote or LAN console      TC : System reset through INIT signal
HE : Display the available commands    TE : Sends a message to other terminals
IT : Modify GSP inactivity timeouts    VFP: Activates Alert Log Display
LC : Configure LAN console             WHO: Display list of GSP connected users
LS : Display LAN console status        XD : GSP Diagnostics and Reset
MR : Modem Reset                       XU : Upgrade the GSP Firmware
MS : Display the status of the Modem

GSP Host Name:  uninitialized
GSP>
```

You can see that in order to log in after the ^b was issued neither a login name or password were required. This is because GSP users have not yet been configured, so anyone can get access to this menu with ^b at the system console.

There are several commands here to which you don't want unauthorized users to have access, so the first step we'll take is to setup security with **SO**:

```
GSP> so


SO
```

```
This command allow you to modify the security options and access control.

GSP wide parameters are:
   . Login Timeout: 1  minutes.
   . Number of Password Faults allowed: 3
   . Flow Control Timeout: 5  minutes.

Do you want to modify the GSP wide parameters? (Y/[N])


User number 1 parameters are:
   . User's Name:
   . User's Login:
   . Organization's Name:
   . Dial-back configuration: Disabled
   . Access Level: Operator
   . Mode: Single
   . User's state: Disabled

Do you want to modify the user number 1 parameters? (Y/[N]/Q to quit) :
```

From this menu, you can change GSP-wide parameters and establish GSP users. The first user added the first time you enter this screen will be the GSP system administrator. This is the user you want to set initially so that your system is secure. You can add one administrator and 19 users. This is an important step since users will now have to add a user name and password to gain access to GSP. The user information you add is shown in the previous listing and is self-explanatory.

In addition to configuring users, there are many useful features of GSP, so I encourage you to experiment with it. One feature I often use is to issue **CL** to display a console history as shown in the following listing for the rp54xx system used throughout the examples in this chapter:

```
GSP> cl


CL



Firmware Version  39.46

Duplex Console IO Dependent Code (IODC) revision 1

------------------------------------------------------------------------------
   (c) Copyright 1995-1998, Hewlett-Packard Company, All rights reserved
------------------------------------------------------------------------------

  Processor   Speed            State           CoProcessor State  Cache Size
  Number                                       State              Inst    Data
  ---------  --------   --------------------  -----------------  ------------
      0      440  MHz   Active                Functional         512 KB   1 MB
      3      440  MHz   Idle                  Functional         512 KB   1 MB

  Central Bus Speed (in MHz)  :         82
  Available Memory            :    2097152  KB
  Good Memory Required        :      11468  KB

Press Q/q to quit, Enter to continue:


   Primary boot path:    0/0/1/1.2
   Alternate boot path:  0/0/2/0.2
   Console path:         0/0/4/0.0
   Keyboard path:        0/0/4/0.0

 WARNING:  The non-destructive test bit was set, so memory was not tested
           destructively.  Information only, no action required.


Processor is booting from first available device.

To discontinue, press any key within 10 seconds.

10 seconds expired.
```

```
Proceeding...

Trying Primary Boot Path
------------------------
Booting...
Boot IO Dependent Code (IODC) revision 1



Press Q/q to quit, Enter to continue:

HARD Booted.

ISL Revision A.00.38  OCT 26, 1994

ISL booting  hpux

Boot
: disk(0/0/1/1.2.0.0.0.0.0;0)/stand/vmunix
6340608 + 821576 + 695024 start 0x210ce8


alloc_pdc_pages: Relocating PDC from 0xf0f0000000 to 0x7f9ab000.
gate64: sysvec_vaddr = 0xc0002000 for 1 pages
Unexpected interrupt on EIRR bit 32
NOTICE: autofs_link(): File system was registered at index 3.
NOTICE: nfs3_link(): File system was registered at index 5.


    System Console is on the Built-In Serial Interface
Logical volume 64, 0x3 configured as ROOT
Logical volume 64, 0x2 configured as SWAP
Logical volume 64, 0x2 configured as DUMP
    Swap device table:  (start & size given in 512-byte blocks)
        entry 0 - major is 64, minor is 0x2; start = 0, size = 2097152
Starting the STREAMS daemons-phase 1
Checking root file system.
log replay in progress
replay complete - marking super-block as CLEAN
Root check done.
Create STCP device files
Starting the STREAMS daemons-phase 2
    B2352B/9245XB HP-UX (B.11.00) #1: Wed Nov  5 22:38:19 PST 1997

Memory Information:
    physical page size = 4096 bytes, logical page size = 4096 bytes

    Physical: 2097152 Kbytes, lockable: 1575156 Kbytes, available: 1816368 Kbyts

/sbin/ioinitrc:
/dev/vg00/lvol1: 34 files, 0 icont, 25716 used, 173665 free (81 frags, 21698 bl)

/sbin/bcheckrc:
Checking for LVM volume groups and Activating (if any exist)
Volume group "/dev/vg00" has been successfully changed.
Activated volume group
Volume group "/dev/vg01" has been successfully changed.
Resynchronized volume group /dev/vg00
Resynchronized volume group /dev/vg01
vxfs fsck: sanity check: root file system OK (mounted read/write)
Checking hfs file systems
/sbin/fsclean: /dev/vg00/lvol1 (mounted) ok
HFS file systems are OK, not running fsck

Checking vxfs file systems
/dev/vg00/lvol8 :
vxfs fsck: sanity check: /dev/vg00/lvol8 needs checking
log replay in progress
replay complete - marking super-block as CLEAN
/dev/vg01/lvol09 :
vxfs fsck: sanity check: /dev/vg01/lvol09 needs checking
log replay in progress
replay complete - marking super-block as CLEAN
/dev/vg00/lvol3 :
vxfs fsck: sanity check: root file system OK (mounted read/write)
```

```
/dev/vg00/lvol4 :
vxfs fsck: sanity check: /dev/vg00/lvol4 needs checking
log replay in progress
replay complete - marking super-block as CLEAN
/dev/vg00/lvol5 :
vxfs fsck: sanity check: /dev/vg00/lvol5 needs checking
log replay in progress
replay complete - marking super-block as CLEAN
/dev/vg00/lvol6 :
vxfs fsck: sanity check: /dev/vg00/lvol6 needs checking
log replay in progress
replay complete - marking super-block as CLEAN
/dev/vg00/lvol7 :
vxfs fsck: sanity check: /dev/vg00/lvol7 needs checking
log replay in progress
replay complete - marking super-block as CLEAN
```


```
(c)Copyright 1983-1997 Hewlett-Packard Co.,  All Rights Reserved.
(c)Copyright 1979, 1980, 1983, 1985-1993 The Regents of the Univ. of California
(c)Copyright 1980, 1984, 1986 Novell, Inc.
(c)Copyright 1986-1992 Sun Microsystems, Inc.
(c)Copyright 1985, 1986, 1988 Massachusetts Institute of Technology
(c)Copyright 1989-1993  The Open Software Foundation, Inc.
(c)Copyright 1986 Digital Equipment Corp.
(c)Copyright 1990 Motorola, Inc.
(c)Copyright 1990, 1991, 1992 Cornell University
(c)Copyright 1989-1991 The University of Maryland
(c)Copyright 1988 Carnegie Mellon University
(c)Copyright 1991-1997 Mentat, Inc.
(c)Copyright 1996 Morning Star Technologies, Inc.
(c)Copyright 1996 Progressive Systems, Inc.
(c)Copyright 1997 Isogon Corporation


                          RESTRICTED RIGHTS LEGEND
Use, duplication, or disclosure by the U.S. Government is subject to
restrictions as set forth in sub-paragraph (c)(1)(ii) of the Rights in
Technical Data and Computer Software clause in DFARS 252.227-7013.


                          Hewlett-Packard Company
                          3000 Hanover Street
                          Palo Alto, CA 94304 U.S.A.

Rights for non-DOD U.S. Government Departments and Agencies are as set
forth in FAR 52.227-19(c)(1,2).
```


```
/sbin/auto_parms: DHCP access is disabled (see /etc/auto_parms.log)

/sbin/rc: failed to read row and column info from screen

     HP-UX Start-up in progress
     ___________________________

     Configure system crash dumps ............................. OK
     Mount file systems ....................................... OK
     Update kernel and loadable modules ....................... N/A
     Initialize loadable modules .............................. N/A
     Setting hostname ......................................... OK
     Set privilege group ...................................... N/A
     Display date ............................................. N/A
     Save system crash dump if needed ......................... N/A
     Enable auxiliary swap space .............................. OK
     Start syncer daemon ...................................... OK
     Configure HP Fibre Channel interfaces .................... OK
     Configure Loopback interfaces (lo0) ...................... OK
     Start Software Distributor agent daemon .................. OK
     Configuring all unconfigured software filesets ........... OK
     Recover editor crash files ............................... OK
     Clean UUCP ............................................... OK
     List and/or clear temporary files ........................ N/A
     Clean up old log files ................................... OK
     Start system message logging daemon ...................... OK
     Start pty allocator daemon ............................... OK
     Start network tracing and logging daemon ................. OK
```

```
     Configure HP Ethernet interfaces ......................... OK
     Configure HP 100BT interfaces ............................ OK
     Configure HP SPP 100BT interfaces ........................ N/A
     Configure LAN interfaces ................................. OK
     Start name server daemon ................................. N/A
     Start NFS core subsystem ................................. OK
     Start NIS+ server subsystem .............................. OK
     Start NIS+ client subsystem .............................. OK
     Start NIS server subsystem ............................... OK
     Start NIS client subsystem ............................... OK
     Start NFS client subsystem ............................... OK
     Start the Trusted Mode with Nis+ subsystem ............... N/A
     Configure pseudo devices for MAC/LLA access .............. OK
     Start multicast routing daemon ........................... N/A
     Start Internet services daemon ........................... OK
     Start dynamic routing daemon ............................. N/A
     Start router discover protocol daemon .................... N/A
     Configuring PPP Interface ................................ OK
     Start RARP protocol daemon ............................... N/A
     Start remote system status daemon ........................ N/A
     Configuring man pages for Internet Services .............. OK
     Starting mail daemon ..................................... OK
     Starting outbound connection daemons for DDFA software .... N/A
     Start SNMP Master Network Management daemon .............. OK
     Start OSPF MIB Network Management subAgent ............... N/A
     Start SNMP HP-UNIX Network Management subAgent ........... OK
     Start SNMP MIB-2 Network Management subAgent ............. OK
     Start SNMP Trap Dest Network Management subAgent ......... OK
     Start DCE daemons ........................................ N/A
     Start RPC daemon if needed ............................... OK
     Start the Isogon License Server daemon ................... N/A
     Start remote boot daemon ................................. OK
     Starting X Font Server at TCP port 7000 .................. OK
     Start vt daemon .......................................... OK
     Start time synchronization ............................... N/A
     Start accounting ......................................... N/A
     Starting the password/group assist subsystem ............. OK
     Starting HP Disk Array Manager daemons. .................. OK
     Starting disk array monitor daemons. ..................... OK
     Start print spooler ...................................... N/A
     Starting HP Distributed Print Service .................... OK
     Start clock daemon ....................................... OK
     Support Tools Informational Fileset ...................... OK
     Start environment monitoring daemon ...................... OK
     Start auditing subsystem ................................. N/A
     Start audio server daemon ................................ N/A
     Start Distributed Single Logical Screen daemon ........... OK
     SAM System administration configuration .................. OK
     Reinitialize Software Distributor agent daemon ........... OK
     Configure HP Fibre Channel Mass Storage interfaces ....... OK
     Start NFS server subsystem ............................... OK
     Start X print server(s) .................................. N/A
     Starting ColdFusion Application Server ................... OK
     Start CDE login server ................................... OK


The system is ready.

GenericSysName [HP Release B.11.00] (see /etc/issue)
Console Login:
************************10/100 Mb/s LAN/9000 Networking*******************@#%
Fri May 26 EDT 2000 15:43:26.401827  DISASTER     Subsys:BASE100      Loc:00000
<6001> HPCORE 10/100BASE-T driver detected bad cable connection between
       the adapter in slot 0 and the hub or switch.
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

************************10/100 Mb/s LAN/9000 Networking*******************@#%
Fri May 26 EDT 2000 17:18:31.401884  DISASTER     Subsys:BASE100      Loc:00000
<6001> HPCORE 10/100BASE-T driver detected bad cable connection between
       the adapter in slot 0 and the hub or switch.
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

GenericSysName [HP Release B.11.00] (see /etc/issue)
Console Login: root
Password:
```

This listing shows a smooth boot; however, there are often boot-related problems you may encounter and you can produce this listing with ^b anytime to review the sequence of events that took place at the time of boot.

You can obtain the status of power modules with **ps** as shown in the following listing:

```
GSP> ps


PS

Power Monitor Status:
Temperature     : Normal      Over temperature: Enable
Power Switch    : ON          Soft power      : Disable
Global fan state: Normal      System power state: On
          Power supplies          |  Fan
 #  State          Type           |  States
-------------------------------------------------------------
0  Normal         1220 Watt       | Normal
1  Normal         1220 Watt       | Normal
2  Not Installed      -           | Normal
3    -                -           | Normal
4    -                -           | Normal
5    -                -           | Normal
6    -                -           | Normal
7    -                -           | Normal


GSP Host Name:  uninitialized
GSP>
```

This shows that we have two power supplies installed and a third is not installed.

You can view processor status with **ss** as shown in the following listing:

```
GSP> ss


SS

System Processor Status:

   Monarch Processor: 0

   Processor 0 is : Installed and Configured
   Processor 1 is : Not Installed
   Processor 2 is : Not Installed
   Processor 3 is : Installed and Configured


GSP Host Name:  uninitialized
GSP>
```

After you have completed your GSP-related work you can return to console mode from the GSP prompt simply by issuing **co**, as shown in the following listing:

```
GSP> co


CO

Leaving Guardian Service Processor Command Interface and entering
Console mode. Type Ctrl-B to reactivate the GSP Command Interface.
GenericSysName [HP Release B.11.00] (see /etc/issue)
Console Login:
```

Anytime that you are on the console, you can issue ^b and get access to GSP to issue any GSP commands and then get back to console mode with **co**.

Table 1-4 is a list of GSP commands available on an rp7400 system at the time of this writing. Although an rp54xx system was used in the examples in this chapter, the GSP commands for platforms are similar. Keep in mind the GSP commands for your system may be somewhat different from those listed. GSP is updated occassionally, so the list for the rp7400 may also be somewhat different from that shown in Table 1-4.

Table 1-4. List of GSP Commands for rp7400

Command (Function)	Explanation
AR (Config)	Configure the Automatic System Restart.
CA (Config)	Configure Asynchronous and Modem parameters.
CE (Chassis Codes)	Log a chassis code in the SAS chassis code history.
CL (Console)	Display console history.
CO (Console)	Return to console mode.
DC (Config)	Default configuration.
DI (Remote)	Disconnect remote or LAN console.
DR (Remote)	Disable remote or LAN console access.
ER (Remote)	Enable remote or LAN console access.
HE (Help)	Display the list of available commands.
IT (Admin)	Modify SAS inactivity timeouts.
LC (Config)	Configure LAN connected and PPP console.
LR (Paging)	Reserve modem for paging.
LS (Console)	Display LAN connected and PPP console status.
MR (Remote)	Modem Reset.
MS (Remote)	Display the status of the modem.
PC (Remote)	Remote power control.
PG	Configure Paging.

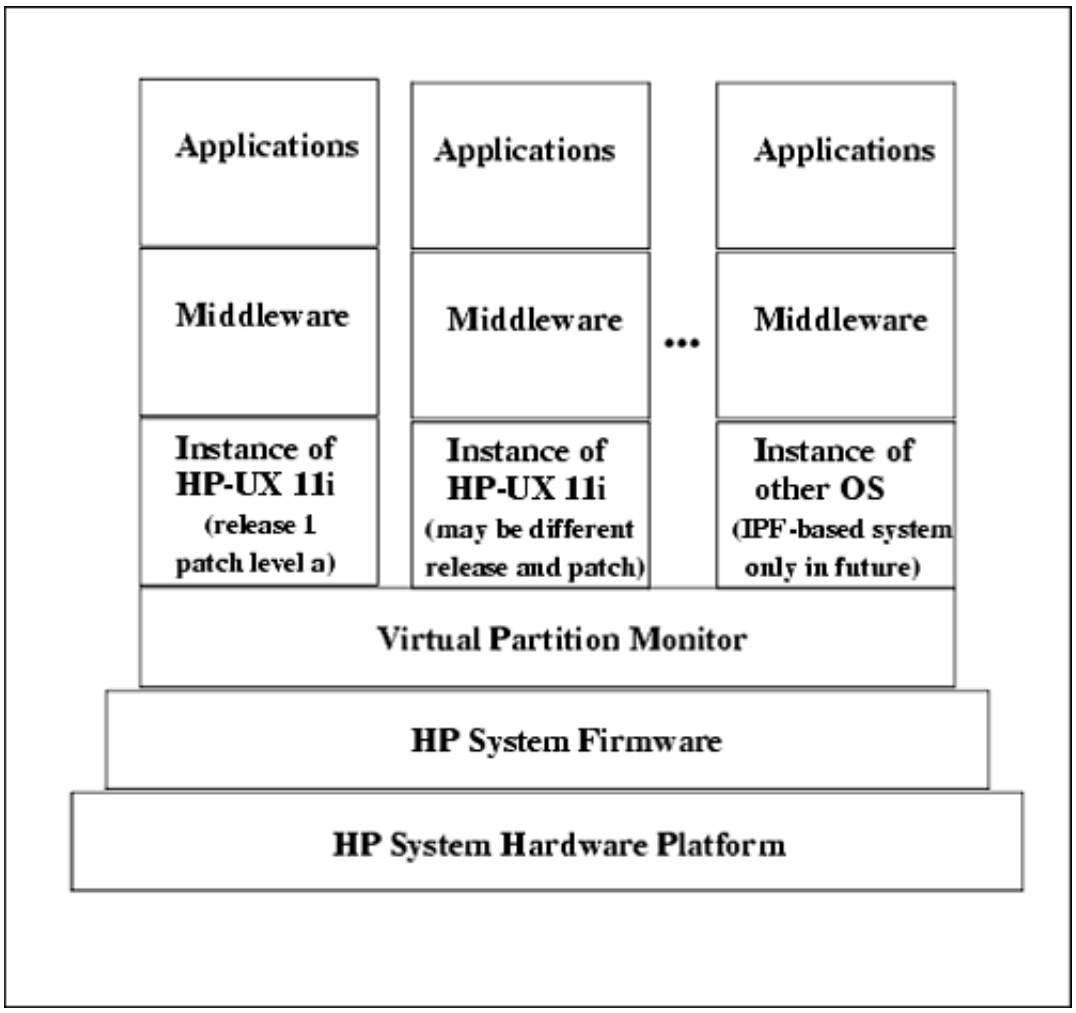
(Paging)	
PS (Status)	Display the status of the power.
RP (Reset)	System reset through RST signal.
SE (Session)	Activate a system session on local or remote port. (One session/remote and local console.)
SL (Status)	Display SPU status logs.
SO (Admin)	Configure security options and access control.
SS (Reset)	Display the status of the system processors.
TC (Session)	System reset through INIT signal.
TE (Admin)	Sends a message to other terminals.
TN (Session)	Start a Telnet session on local or remote port.
UR (Remote)	Unlock remote support modern access.
VSC (Config)	Configure the Virtual Front Panel display.
VFD (Chassis Codes)	Activates the immediate display of the Virtual Front Panel.
WHO (Admin)	Display a list of SAS connected users.
XD	SAS Diagnostics and Reset.
XU	Upgrade the GSP firmware.
ZTOGCCD (Chassis Codes)	DEBUG feature: Change the way the chassis codes are displayed by SL command and VFPD.

Virtual Partition Boot Process Overview

Booting partitions is similar to booting without partitions. The primary difference is the *Virtual Partition Monitor* - what I'll call **vpmon** in this chapter. **vpmon** is loaded at the time of boot and is located in **/stand/vpmon**.

The **vpmon** sits between the firmware and operating system on your HP system. Figure 1-10 depicts the position of the **vpmon** relative to other HP system components.

Figure 1-10. Virtual Partitions Software Stack



Notice in the figure that different versions and patch levels of HP-UX 11i may be running in vPars. On Itanium Processor Family (IPF) systems, operating systems other than HP-UX may be running in vPars. This functionality was not available at the time of this writing, so only HP-UX 11i-based vPars are covered in this book.

In the process of creating vPars, a partition database is produced that tracks the resources associated with vPars. The **vpmon** manages these resources, loads kernels, and performs other functions that make it look as though each virtual partition is its own system.

Rather than booting a kernel directly from the ISL (see the non-vPars-specific portion of this chapter for detailed information on all aspects of booting, including ISL), you boot the **vpmon**. The **vpmon** then loads

the partition database **/stand/vpdb** and creates the vPars based on the resources allocated to the vPars in the database. **vpdb** is the default database, which contains partition-related information. A copy of **vpdb** is kept on the disk for every partition and is automatically kept synchronized. When a change is made to any partition, the master **vpdb** is updated and then the local copies on other vPars are automatically updated. This synchronization occurs every few seconds and ensures that **vpdb** on all running partitions remains synchronized. If a partition is not running its **vpdb** cannot be updated.

Without vPars you would boot the HP-UX kernel directly from ISL, as shown below:




```
ISL> hpux /stand/vmunix
```

You can boot the **vpmon** from ISL with the command below:

```
ISL> hpux /stand/vpmon
```

The **vpmon** is invoked with this command and the **vpmon** prompt appears from which you can load vPars.

From the **vpmon** prompt we could issue the **vparload** command to load one or more Virtual Partitions. There are many options to the **vparload** command. There is no man page for **vparload** in Appendix B because it is a **vpmon** command. **vparboot** is a shell command so there is a man page in Appendix B for it. The following are the three forms of **vparboot** at the time of this writing:

```
form1: vparload -all

form2: vparload -auto

form3: vparload -p vp_name [-b kernelpath] [-o boot_options]
[-B hardware_path]
```

form1 boots all vPars. *form2* boots all vPars that have the *autoboot* attribute set. *form3* allows you to specify options such as: the *kernelpath* to boot; the *boot_options,* such as *"is"* for single user mode; or *hardware_path,* which specifies the boot device to be used for the vPar.

Issuing the **/stand/vpmon** command at the *ISL>* prompt gives us the *MON>* prompt. To use **vparload** to boot all Virtual Partitions on a server we would issue the following command:

```
MON> vparload -all
```

To use **vparload** to boot the Virtual Partition *symbol1,* we would issue the following command:

```
MON> vparload -p symbol1
```

You could perform the steps to load both the **vpmon** and virtual partition *symbol1* from ISL with the command below at the ISL prompt:




```
ISL> hpux /stand/vpmon vparload -p symbol1
```

This command boots both **vpmon** and then *symbol1*. You may perform experimentation with the kernels of the vPars on your system and have to boot different kernels. The following example shows booting from a kernel called **vmunix_test1**:

```
MON> vparload -p symbol1 -b /stand/vmunix_test1
```

As a side note, the kernel path above is loaded with this **vparload**, but no permanent changes were made. To make a permanent change to the vPars database you would issue the following command:

```
# vparmodify -p symbol1 -b "stand/vmunix_test1"
```

The vPar database has now been modified to have a default kernel of **/stand/vmunix_test1**.

You can boot a vPar in single-user mode with the following command:

```
MON> vparload -p symbol1 -o "is"
```

There are a variety of other options that you can use for booting vPars. As you can see from this discussion, the options are similar to options you would use on a non-vPars system.

There are a variety of other commands that you can issue from *MON*>. The following shows some of the more common commands:

help Displays all of the commands available in *MON*> as shown below (**?** produces the same results):

```
MON> help

Supported Commands:

?               Print list of commands
cat             Dump contents of file to screen
cbuf            Dump contents of console buffer
getauto         Print the AUTO file
help            Print list of commands
lifls           List files in LIF directory
log             View the event log
ls              List files in a directory
readdb          Read a partition DB
reboot          Reboot system
scan            Scan the system
toddriftreset   Reset the TOD drift of all vpars
vparload        Load vPar
vparinfo        Display vPar info
```

scan Displays all hardware discovered by the Virtual Partition monitor and indicates which vPar, if any, owns the device. Note in the following listing that some components, such as the System Bus Adapters (SBAs) at *0* and *1* and the memory controller at *192,* are owned by the Virtual Partition monitor and are therefore owned by *VPAR-ALL*. Other components are owned by vPar *uhnjlvp1/2*. This is an informative listing of all components and the vPar that owns them, as shown below:

```
MON> scan
0           BUSCONV      sv_model= 12  HPA=0xfffffffffed00000   VPAR=ALL
0/0         BUS_BRIDGE   sv_model= 10  HPA=0xfffffffffbffe0000   VPAR=uhnjlvp1
0/1         BUS_BRIDGE   sv_model= 10  HPA=0xfffffffffbffe2000   VPAR=NONE
0/2         BUS_BRIDGE   sv_model= 10  HPA=0xfffffffffbffe4000   VPAR=NONE
0/4         BUS_BRIDGE   sv_model= 10  HPA=0xfffffffffbffe8000   VPAR=NONE
0/5         BUS_BRIDGE   sv_model= 10  HPA=0xfffffffffbffea000   VPAR=NONE
0/8         BUS_BRIDGE   sv_model= 10  HPA=0xfffffffffbfff0000   VPAR=uhnjlvp2
0/10        BUS_BRIDGE   sv_model= 10  HPA=0xfffffffffbfff4000   VPAR=NONE
0/12        BUS_BRIDGE   sv_model= 10  HPA=0xfffffffffbfff8000   VPAR=uhnjlvp1
1           BUSCONV      sv_model= 12  HPA=0xfffffffffed40000   VPAR=ALL
1/0         BUS_BRIDGE   sv_model= 10  HPA=0xfffffffffece0000   VPAR=NONE
1/2         BUS_BRIDGE   sv_model= 10  HPA=0xfffffffffece4000   VPAR=NONE
1/4         BUS_BRIDGE   sv_model= 10  HPA=0xfffffffffece8000   VPAR=NONE
1/8         BUS_BRIDGE   sv_model= 10  HPA=0xfffffffffecf0000   VPAR=NONE
1/10        BUS_BRIDGE   sv_model= 10  HPA=0xfffffffffecf4000   VPAR=NONE
1/12        BUS_BRIDGE   sv_model= 10  HPA=0xfffffffffecf8000   VPAR=uhnjlvp2
36          BUSCONV      sv_model= 12  HPA=0xfffffffffed24000   VPAR=NONE
37          NPROC        sv_model=  4  HPA=0xfffffffffed25000   VPAR=uhnjlvp1
44          BUSCONV      sv_model= 12  HPA=0xfffffffffed2c000   VPAR=NONE
45          NPROC        sv_model=  4  HPA=0xfffffffffed2d000   VPAR=uhnjlvp2
192         MEMORY       sv_model=  9  HPA=0xfffffffffedc0000   VPAR=ALL
MON> vparinfo uhnjlvp1

Resources assigned to partition 0 (uhnjlvp1)...
-----------------------------------
0           0xfffffffffed00000      1       0  TYPE= 7  SV_MODEL= 12
0/0         0xfffffffffbffe0000     1       0  TYPE=13  SV_MODEL= 10
0/12        0xfffffffffbfff8000     1       0  TYPE=13  SV_MODEL= 10
1           0xfffffffffed40000      1       0  TYPE= 7  SV_MODEL= 12
37          0xfffffffffed25000      1       0  TYPE= 0  SV_MODEL=  4
192         0xfffffffffedc0000      1       0  TYPE= 1  SV_MODEL=  9

Effective Size: 1572864 kb
```

```
Boot: 0/0/2/0.6.0
Console: 0/0/4/0
Boot Image: (0/0/2/0.6.0;)/stand/vmunix
Boot Options:
AUTOBOOT:off
DYNAMIC
```

vparinfo *[partition_name]* Displays resources assigned to the specified vPar or those resources that are unassigned if the vPar name is not specified. The following listing shows the resources assigned to vPar *uhnjlvp2*:

```
MON> vparinfo uhnjlvp2

Resources assigned to partition 1 (uhnjlvp2)...
-----------------------------------
0           0xfffffffffed00000      1       0  TYPE= 7  SV_MODEL= 12
0/8         0xfffffffbfff0000       1       0  TYPE=13  SV_MODEL= 10
1           0xfffffffffed40000      1       0  TYPE= 7  SV_MODEL= 12
1/12        0xfffffffffecf8000      1       0  TYPE=13  SV_MODEL= 10
45          0xfffffffffed2d000      1       0  TYPE= 0  SV_MODEL=  4
192         0xfffffffffedc0000      1       0  TYPE= 1  SV_MODEL=  9

Effective Size: 524288 kb

Boot: 1/12/0/0.12.0
Console: Default
Boot Image: (1/12/0/0.12.0;)/stand/vmunix
Boot Options:
AUTOBOOT:off
DYNAMIC
```

vparload The three forms of this command are listed below. The example after the three forms shows the beginning of loading all vPars with **vparload -all**:

```
form1: vparload -all

form2: vparload -auto

form3: vparload -p vp_name [-b kernelpath] [-o boot_options]
[-B hardware_path]



MON> vparload -all
[MON] Booting uhnjlvp2...
[MON] Booting uhnjlvp1...
[MON] Console client set to uhnjlvp2

[MON] uhnjlvp2 loaded

[MON] uhnjlvp1 loaded

              .
              .
              .
```

lifls Displays the files in the LIF area as shown below:

```
MON> lifls
volume ISL10 data size 7802 directory size 8
filename   type   start   size     implement  created
=======================================================
ODE        -12960 584     848
MAPFILE    -12277 1432    128
SYSLIB     -12280 1560    353
CONFIGDATA -12278 1920    218
SLMOD2     -12276 2144    140
SLDEV2     -12276 2288    134
SLDRV2     -12276 2424    168
SLSCSI2    -12276 2592    116
MAPPER2    -12279 2712    142
IOTEST2    -12279 2856    89
PERFVER2   -12279 2952    125
PVCU       -12801 3080    64
```

```
SSINFO     -12286 3144    2
ISL        -12800 3152    306
AUTO       -12289 3464    1
HPUX       -12928 3472    848
LABEL      -23951 4320    8
```

getauto Displays the contents of the LIF area auto file as shown below:

```
MON> getauto
hpux
```

log Displays all information in the Virtual Partition monitor log. The information is displayed in chronological order as shown below:

[View full width]

```
MON> log
INFO:CPU0:MON:[20:16:31 10/1/2001 GMT] VPAR Monitor version 0.2 started
INFO:CPU0:MON:Version string: @(#) $Revision: vpmon:    vw: -- selectors: CUP
11.11_BL2001_0616                 'cup_cvinod_vpar_ncf_trial' 'cup_shep_r11.11'  Tue Sep
➥ 25 18:2
1:12 PDT 2001 $
INFO:CPU0:MON:Partition uhnjlvp1 monarch set to 37
INFO:CPU0:MON:Partition uhnjlvp2 monarch set to 45
```

ls *directory* Lists the contents of a directory in much the same way as UNIX **ls** command. At the time of this writing, the *directory* must be HFS. The **/stand** directory is used by default. You can also add *-a* for all entries, *-l* for long listing, *-n* for numerical entries, *-i* for inode, and *-F* for file type appended to the output, such as a slash (/) after directories, as shown below using the *-l* option:

```
MON> ls -l /stand
drwxr-xr-x    2 0     0           8192 lost+found
-rw-rw-rw-    1 0     3           5416 ioconfig
drwxr-xr-x    4 0     3           2048 build
-rwxr-xr-x    1 0     0       15273152 vmunix
drwxrwxrwx    5 0     3           1024 dlkm.vmunix.prev
-rw-r--r--    1 0     3             19 bootconf
-r--r--r--    1 0     3             82 kernrel
-rw-------    1 0     0             12 rootconf
-r--r--r--    1 0     3           1040 system
-r--r--r--    1 0     3           1035 system.prev
-rwxr-xr-x    1 0     3       14488016 vmunix.prev
drwxrwxrwx    5 0     0           1024 dlkm
drwxr-xr-x    2 0     3           1024 system.d
drwxr-xr-x    2 0     3           1024 krs
drwxr-xr-x    2 0     0           1024 krs_tmp
drwxr-xr-x    2 0     0           1024 krs_lkg
-r-xr-xr-x    1 2     2         845680 vpmon
-rw-------    1 0     0         102640 vpmon.dmp
-rw-------    2 0     0           8232 vpdb
```

cbuf *partition_name* Displays the information in the console buffer for the specified Virtual Partition. If no information is present in the buffer, you'll receive the following message:

```
MON> cbuf uhnjlvp1
    Buffer is empty
```

Many of these commands at *MON>* are informative. Let's now move on to vPars states.

Virtual Partition Boot States

When we boot a Virtual Partition, it progresses through *load*, then *boot*, and finally, *up*. There are other states in which you may find a Virtual Partition as well. Table 1-5 summarizes the states of Virtual Partitions at the time of this writing:

Table 1-5. Virtual Partitions States

vPars State	Description
load	The kernel image of a Virtual Partition is being loaded into memory. This is done by the Virtual Partition monitor.
boot	The Virtual Partition is in the process of booting. The kernel image has been successfully loaded by the Virtual Partition monitor.
up	The Virtual Partition has been successfully booted and is running.
shut	The Virtual Partition is in the process of shutting down.
down	The Virtual Partition is not running and is down.
crash	The Virtual Partition has experienced a panic and is crashing.
hung	The Virtual Partition is not responding and is hung.

These states are used later when we create and boot Virtual Partitions and watch them boot by issuing successive **vparstatus** commands. The following example shows the process of the Virtual Partition *cable2* booting:

```
# vparboot -p cable2
# vparstatus
[Virtual Partition]                                                  Boot
Virtual Partition Name         State Attributes Kernel Path          Opts
============================== ===== ========== ======================== =====
cable1                         Up    Dyn,Manl   /stand/vmunix
cable2                         Load  Dyn,Manl   /stand/vmunix

[Virtual Partition Resource              CPU    Num        Memory (MB)
 Summary]                      CPU     Bound/   IO   # Ranges/
Virtual Partition Name         Min/Max  Unbound  devs  Total MB    Total MB
==============================  ================  ====  ====================
cable1                           2/  4    2   0     4    0/  0         2048
cable2                           2/  2    2   0     4    0/  0         1024
# vparstatus
[Virtual Partition]                                                  Boot
Virtual Partition Name         State Attributes Kernel Path          Opts
============================== ===== ========== ======================== =====
cable1                         Up    Dyn,Manl   /stand/vmunix
cable2                         Boot  Dyn,Manl   /stand/vmunix

[Virtual Partition Resource              CPU    Num        Memory (MB)
 Summary]                      CPU     Bound/   IO   # Ranges/
Virtual Partition Name         Min/Max  Unbound  devs  Total MB    Total MB
==============================  ================  ====  ====================
cable1                           2/  4    2   0     4    0/  0         2048
cable2                           2/  2    2   0     4    0/  0         1024
# vparstatus
[Virtual Partition]                                                  Boot
Virtual Partition Name         State Attributes Kernel Path          Opts
============================== ===== ========== ======================== =====
cable1                         Up    Dyn,Manl   /stand/vmunix
cable2                         Up    Dyn,Manl   /stand/vmunix

[Virtual Partition Resource              CPU    Num        Memory (MB)
 Summary]                      CPU     Bound/   IO   # Ranges/
Virtual Partition Name         Min/Max  Unbound  devs  Total MB    Total MB
==============================  ================  ====  ====================
cable1                           2/  4    2   0     4    0/  0         2048
cable2                           2/  2    2   0     4    0/  0         1024
#
```

This example shows *cable2* progressing through the *load*, then *boot*, and finally, *up*. *cable2* was booted from *cable1* running on the same hardware. The system had already gone through the boot process when *cable1* booted. The boot time for *cable2* is very quick since most of the hardware is already running. The boot time for the first Virtual Partition is comparable to the boot time of a non-Virtual Partition system, but the subsequent vPars boot much more quickly.

setboot Command and vPars

The **setboot** command on a non-vPars system reads from and writes to stable storage. On a vPars system the **setboot** command interacts with the Virtual Partition database. Running **setboot** on a vPars system has the effects shown in Table 1-6:

Table 1-6. setboot and Virtual Partitions

vPars setboot Option	Description
-a	Changes the alternate boot path of the Virtual Partition. To set the alternate boot path: `# setboot -a 0/8/0/0.8.0.5.0.0.0`
-b	Sets the autoboot attribute of the Virtual Partition. To set Autoboot *on*: `# setboot -b on`
-p	Changes the primary boot path of the Virtual Partition. To set the primary boot path: `# setboot -p 0/0/1/1.2.0`
-s	Has no effect.
no options	Displays information about boot attributes.

The **setboot** command is one of the aspects of working with vPars that is different from a non-vPars system.



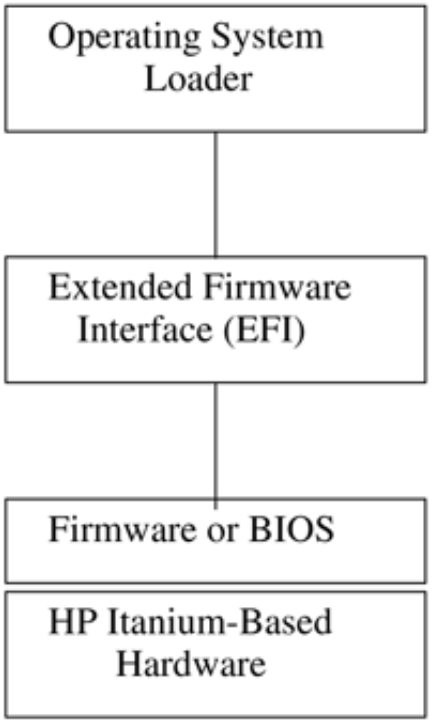
You can also set the primary and alternate boot paths with **vparmodify**.

Extensible Firmware Interface (EFI)

EFI is an interface between operating systems and platform firmware. EFI is implemented on HP Itanium-based systems. There are many components to EFI that can be found on the Intel Web site. At the time of this writing the following is the home URL for EFI: http://www.intel.com/technology/efi/index.htm.

Figure 1-11 shows where EFI fits in between platform firmware and the operating system loader:

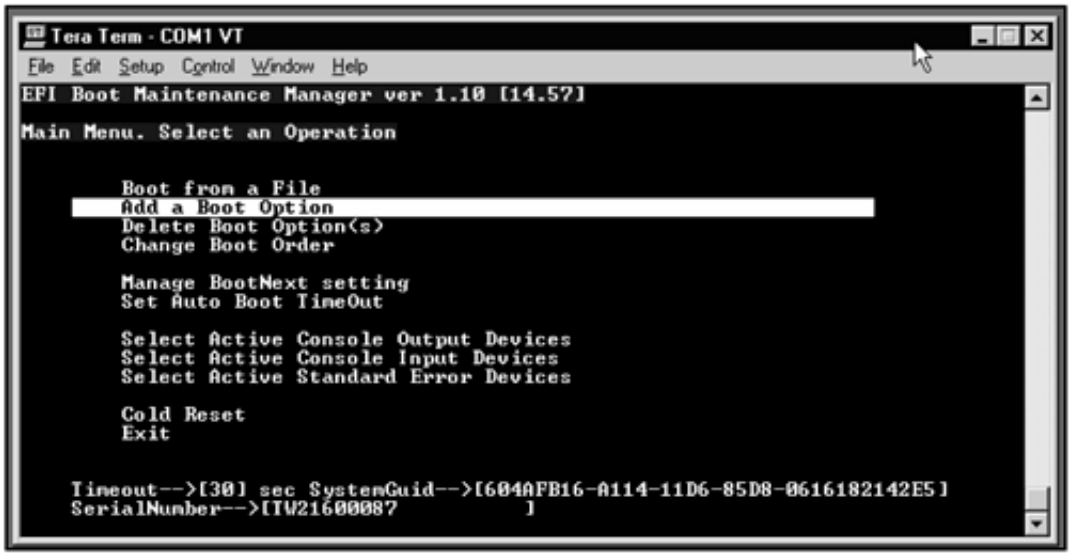
Figure 1-11. EFI on HP Itanium-Based Systems



We've covered the HP firmware extensively in this chapter as well as the HP-UX operating system loader. EFI is an additional component that is independent of the operating system and provides a shell for interfacing to multiple operating systems.

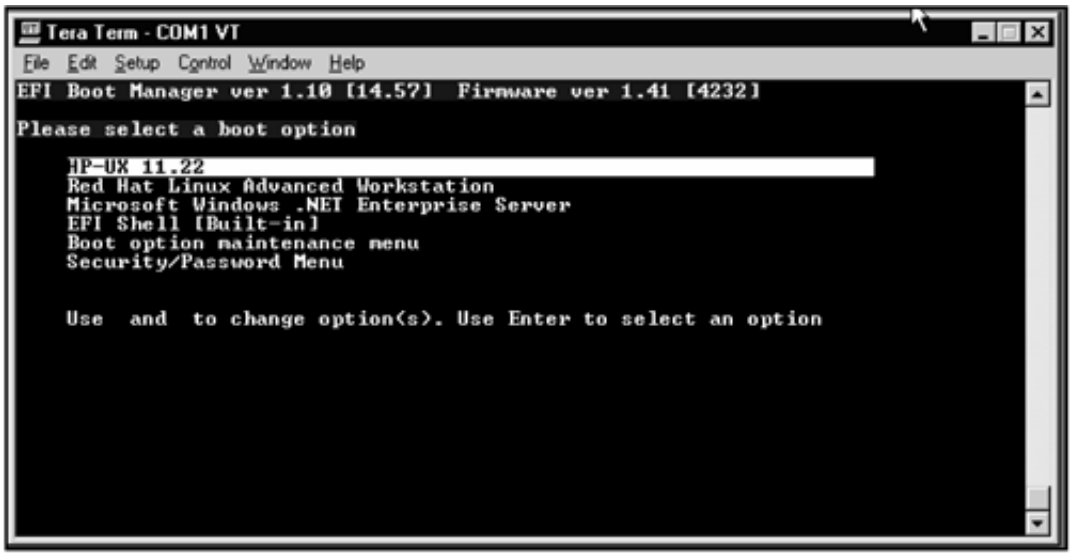
Working with EFI is straightforward in that it is menu-driven so you make the desired selections and traverse the menu hierarchy. Figure 1-12 shows the main EFI screen from which you can make various boot-related selections:

Figure 1-12. EFI Boot Administration Main Screen



From the menu shown in Figure 1-12, you can perform a variety of boot-related tasks, such as *Change Boot Order* of the operating systems you can boot. Figure 1-13 shows the EFI boot manager:

Figure 1-13. EFI Main Screen



In Figure 1-13 you can see that there are three operating systems from which you can boot on this Itanium-based system. There is an *HP-UX,* a *Red Hat Linux*, and a *Windows* operating system. You can boot any one of these.

Through EFI you can control the boot-related setup on your Itanium-based system. With multiple options for operating systems you can run on Itanium-based systems, you'll use this interface often to coordinate this operating systems.

Manual Pages for Commands Used in Chapter 1

There are several manual pages from commands used in this chapter in Appendix A. The following list of commands from this chapter appears in Appendix A except for **vparmodify** which appears in Appendix B:

boot
hpux
isl
pdc
vparmodify



Chapter 2. Installing HP-UX

Topics Covered in this Chapter

Background on HP-UX 11i Operating Environments

Installing the HP-UX 11i Operating Environment

Setting the System Parameters after Booting

Software Distributor Example

Loading Patches

Software Distributor Background

Manual Pages for Commands Used in Chapter 2





Topics Covered in this Chapter

We'll cover several topics related to installing HP-UX in this chapter. These topics are covered in the order in which you would install the software when initially loading your system:

- Background on HP-UX 11i *Operating Environments*.
- Installing the HP-UX 11i *Operating Environment*.
- Setting system parameters at the time of first boot after loading HP-UX 11i with **set_parms**.
- Viewing *Mission Critical Operating Environment* components and installing application software with Software Distributor.
- Download and install select patches on your system.
- Software Distributor background.

You would typically load software in the order shown above: install the base *Operating Environment*; boot your system and use **set_parms**; load application software; and finally, download and install select patches.





Background on HP-UX 11i Operating Environments

New with HP-UX 11i are *Operating Environments*. These are grouping of software provided by HP that are integrated, tested and supported by HP. There are the five operating environments at the time of this writing:

- Basic HP-UX OE
- Enterprise HP-UX OE
- Mission Critical HP-UX OE
- Minimal Technical OE
- Technical Computing OE

You can expect these OEs to change over time as the needs of 11i users change but these are the five OEs available at the time of this writing. The following two tables (2-1 and 2-2) show high level contents of the three commercial OEs and two technical OEs respectively:

Table 2-1. HP-UX 11i Commercial OEs

Basic HP-UX 11i OE	Enterprise OE	Mission Critical OE
Apache Web Server	All items in *Basic* + the following:	All items in *Basic* + *Enterprise* + the following:
CIFS/9000 Server	EMS HA Monitors	ECM Toolkit
CIFS/9000 Client	MirrorDisk/UX	MC/ServiceGuard (v11.13)
Java JPI	Online JFS (v3.3)	ServiceGuard NFS
Java Runtime Env (v1.2)	OV GlancePlus Pak (English)	Cluster Object Manager
Netscape Communicator (v4.75)	OV GlancePlus Pak (Japanese)	Workload Manager
PAM Kerberos	Process Resource Manager	Workload Manager Toolkit
ServiceControl Manager		
Customer Selectable Software		
100Base-T (HP-PB, EISA)		
ATM (PCI, HSC)		
FDDI (HSC, HP-PB, EISA)		
HyperFabric (PCI, HSC)		
MUX (PCI, EISA)		
TokenRing (PCI, HP-PB, EISA)		
HP-UX Install Utilities 11.11 (IUX)		
Base VxVM Volume Manager		

Online Diagnostics
Perl
Netscape Directory Server
WebQoS Peak Package Edition
Always-Installed NW Drivers
Gigabit Ethernet (PCI, HSC)
FDDI (PCI)
FibreChannel [Tachlite] (PCI)
SCSI RAID (PCI)
Contents of HPUXBaseAux
DMI&SCR
EMS Framework
Judy Libraries
ObAM5
Partition Manager
Software Distributor
HP-UX 11i Core Functionality
HPUXBase64 (64-bit)
HPUXBase32 (32-bit)

Table 2-2. HP-UX 11 Technical OEs

Minimal Technical OE	Technical Computing OE
3D Graphics Developers Kit and Runtime Env	All items in *Minimal Technical OE* + the following:
Netscape Communicator	
Customer Selectable Software	Apache Web Server
100Base-T (HP-PB, EISA)	CIFS/9000 Server
ATM (PCI, HSC)	CIFS/9000 Client
FDDI (HSC, HP-PB, EISA)	FirstSpace VRML Viewer
HyperFabric (PCI, HSC)	Java 3D
MUX (PCI, EISA)	Java JPI
TokenRing (PCI, HP-PB, EISA)	Java Runtime Env (v1.2)
HP-UX Install Utilities 11.11 (IUX)	MLIB
*Online Diagnostics	MPI
*Perl	PAM Kerberos
Netscape Directory Server	Visualize Conference

Always-Installed NW Drivers
Gigabit Ethernet (PCI, HSC)
FDDI (PCI)
FibreChannel [Tachlite] (PCI)
SCSI RAID (PCI)
Contents of HPUXBaseAux
DMI&SCR
EMS Framework
Judy Libraries
ObAM5
Partition Manager
Software Distributor
HP-UX 11i Core Functionality
HPUXBase64 (64-bit)
HPUXBase32 (32-bit)

Please keep in mind that the contents of the OEs shown in the tables will be adjusted over time and the OEs themselves may change but these OEs and their contents are accurate at the time of this writing. In the installation example in this chapter we'll install the Mission Critical OE.

[Team LiB]

Installing the HP-UX 11i Operating Environment

Installing HP-UX means installing one of the 11i *Operating Environments* and later building your complete, fully functional HP-UX system by installing both HP and non-HP applications. The initial system is loaded from the 11i media, or from another system on the network using Ignite-UX. This chapter covers installing from media. Chapter 9 covers installing from a server.

You can have your system delivered with instant ignition, which means that HP-UX has been loaded on your system and you'll only have to add to it the application software you'll need. I cover the complete installation process from media so you can see the process from start to finish. If you have instant ignition on your system, you may need to install additional software on your system in the future and can, therefore, use some of the techniques described in this section to load that software.

One of the features of moving any version of the 11.x operating system (including 11i) is the option of taking advantage of 64-bit computing. Whereas previous versions of HP-UX supported only the 32-bit processor, with 11.x, you have the option of running either the 32-bit or the 64-bit version. Which version of the operating system you choose is dependent on two things: what hardware you are using and your application requirements. The 64-bit version is not supported on some Series 700 workstations and many of the low-end Series 800s, but it is required for new systems. With other Series 800s, most notably the very old K Class and T Class servers, you have the option to install either the 32-bit or the 64-bit version. But don't worry too much if you are unsure whether your hardware supports 64-bit or not. If your hardware doesn't support the 64-bit version, you won't be prompted as to which version to install. The 32-bit version will be installed automatically. If you have the option to install the 32-bit or 64-bit version, then your hardware supports either.

Your application requirements will also help determine whether you will install the 32-bit or 64-bit operating system. If your applications are 32-bit, then you really have no reason to run the 64-bit version. If, however, you will be running a 64-bit application, then you obviously will need to install the 64-bit operating system. The good news is that if you install the 32-bit operating system and then find that an application you want to install is 64-bit, you can upgrade to the 64-bit operating system, assuming, of course, that your hardware supports it.

Booting the System and "Advanced Installation"

In order to install HP-UX software, place the core operating system media for HP-UX 11i into the DVD-ROM or CD-ROM drive. At the time of this writing, the CD-ROM was labeled *HP-UX Release 11.11*. There were two CD-ROMs as part of the core operating system I was working with because I was loading the *Mission Critical Operating Environment* which was on two CD-ROMs (more about the *Operating Environments* later*).* Be sure to insert the CD-ROM Install media before you begin the installation. As your HP 9000 unit boots, you will see a variety of messages fly by, including information about your processors, buses, boot paths, and so on. See Chapter 1, which covered the boot process, to get more information about booting.

The following example shows several steps that were taken. The example begins at the end of early boot. The early boot is the first part of the load process from media. There were many early boot-related messages that appeared before the point where our example begins. We discontinued the boot process by pressing a key. After discontinuing the boot we run *SEArch* to find bootable devices. Among the devices shown is the DVD-ROM drive containing our operating system media. We select *p0* to boot off with the **bo p0** command. We then choose not to interact with IPL.

```
************ EARLY BOOT VFP *************
End of early boot detected
*****************************************

Firmware Version  39.46

Duplex Console IO Dependent Code (IODC) revision 1

------------------------------------------------------------------------------
   (c) Copyright 1995-1998, Hewlett-Packard Company, All rights reserved
------------------------------------------------------------------------------

  Processor   Speed            State           CoProcessor State  Cache Size
  Number                                       State              Inst    Data
  ---------  --------   ---------------------  -----------------  ------------
```

```
      0      440  MHz   Active                Functional         512 KB   1 MB
      3      440  MHz   Idle                  Functional         512 KB   1 MB

  Central Bus Speed (in MHz)  :         82
  Available Memory            :    2097152  KB
  Good Memory Required        :      16908  KB

   Primary boot path:    0/0/1/1.2
   Alternate boot path:  0/0/2/0.2
   Console path:         0/0/4/0.0
   Keyboard path:        0/0/4/0.0

 WARNING:  The non-destructive test bit was set, so memory was not tested
           destructively.  Information only, no action required.


Processor is booting from first available device.
To discontinue, press any key within 10 seconds.

Boot terminated.


---- Main Menu -------------------------------------------------------------

     Command                          Description
     -------                          -----------
     BOot [PRI|ALT|<path>]            Boot from specified path
     PAth [PRI|ALT] [<path>]          Display or modify a path
     SEArch [DIsplay|IPL] [<path>]    Search for boot devices

     COnfiguration menu               Displays or sets boot values
     INformation menu                 Displays hardware information
     SERvice menu                     Displays service commands

     DIsplay                          Redisplay the current menu
     HElp [<menu>|<command>]          Display help for menu or command
     RESET                            Restart the system
----
Main Menu: Enter command or menu > search

Searching for potential boot device(s)
This may take several minutes.

To discontinue search, press any key (termination may not be immediate).


   Path#  Device Path (dec)  Device Path (mnem)  Device Type
   -----  -----------------  ------------------  -----------
   P0     0/0/1/0.3          extscsi.3           Random access media
   P1     0/0/1/1.2          intscsib.2          Random access media
   P2     0/0/2/0.2          intscsia.2          Random access media



Main Menu: Enter command or menu > bo P0
Interact with IPL (Y, N, or Cancel)?> n

Booting...
```

The *SEArch* command in the previous listing showed us the bootable devices. There are three possible boot devices. The second two are in the internal disks, one of which would normally be the primary boot device and the other, which is normally the alternate boot device. These two disks are on two different SCSI busses internal to the rp54xx (formerly know as the L-Class). The first device was the external DVD-ROM that contains the HP-UX 11i CD-ROM, which is operating system media off which we want to boot.

In order to boot off the media and not interact with IPL (for more information on IPL, see Chapter 1), we issued the following command:

```
Main Menu: Enter command or menu > bo p0
Interact with IPL (Y or N)?> N
```

After booting off *P0,* we are given the *Welcome to the HP-UX installation/recovery process!* menu, shown in the following example:

```
          Welcome to the HP-UX installation/recovery process!

Use the <tab> key to navigate between fields, and the arrow keys
within fields.  Use the <return/enter> key to select an item.
Use the <return> or <space-bar> to pop-up a choices list.  If the
menus are not clear, select the "Help" item for more information.

Hardware Summary:         System Model: 9000/800/L2000-44
+---------------------+----------------+-------------------+ [ Scan Again  ]
| Disks: 2  ( 33.9GB) |  Floppies: 0   | LAN cards:   2    |
| CD/DVDs:        1   |  Tapes:    0   | Memory:   2048Mb  |
| Graphics Ports: 0   |  IO Buses: 8   | CPUs:        2    | [ H/W Details ]
+---------------------+----------------+-------------------+

                    [      Install HP-UX      ]

                    [   Run a Recovery Shell   ]

                    [    Advanced Options      ]


        [  Reboot  ]                              [  Help  ]
```

The *Welcome to the HP-UX installation/recovery process!* menu is the first menu displayed. It gives a summary of the hardware on your system. If you want to see more detail, select the *H/W Details* option on the right side of the screen. This takes you to a detailed listing of your hardware. It includes items such as hardware paths, disk drive capacities, and LAN addresses. We won't perform any other functions in this example other than our selection of *Install HP-UX*.

When we select *Install HP-UX,* we get the *User Interface and Media Options* menu shown in the following example:

```
                      User Interface and Media Options

  This screen lets you pick from options that will determine if an
  Ignite-UX server is used, and your user interface preference.

Source Location Options:
  [ * ]  Media only installation
  [   ]  Media with Network enabled (allows use of SD depots)
  [   ]  Ignite-UX server based installation

User Interface Options:
  [   ]  Guided Installation   (recommended for basic installs)
  [ * ]  Advanced Installation (recommended for disk and filesystem management)
  [   ]  No user interface - use all the defaults and go


  Hint: If you need to make LVM size changes, or want to set the
        final networking parameters during the install, you will
        need to use the Advanced mode (or remote graphical interface).



  [   OK   ]                 [ Cancel ]                        [  Help  ]
```

This menu gives you the option of installing from the media only, from the media combined with the software depots on your network, or from the Ignite-UX product. We'll be installing from the media only in our example.

We next select *Advanced Installation* because this gives us the greatest level of flexibility when installing. *Guided Installation* leads you through a basic system configuration setup. It allows for only a few system-specific options. *Advanced Installation* is much more flexible and allows for extensive system-specific parameters to be set. We'll walk through the *Advanced Installation* steps.

The display that now appears is similar to that used by Ignite-UX. In fact, it is the same except that Ignite-UX uses a Graphical User Interface (GUI) versus the Terminal User Interface (TUI). Figure 2-1 shows this display.

Figure 2-1 Ignite-UX Display *Basic* Tab Area Using TUI

```
                    /opt/ignite/bin/itool

/-------\/----------\/--------\/-------------\/----------\
| Basic ||  Software|| System || File System || Advanced |
\       \-------------------------------------------------/
 Configurations:  [ HP-UX B.11.11 Default  ->] [ Description...  ]

 Environments:    [ Mission Critical OE-64bit ->] (HP-UX B.11.11)

 [ Root Disk...  ] SEAGATE ST318203LC, 0/0/1/1.2.0, 17366 M

 File System:     [ Logical Volume Manager (LVM) with VxFS  ->]

 [ Root Swap (MB)... ] 1024  Physical Memory (RAM) =   2048 MB
       [  Languages...  ] English  [Keyboards...  ] [ Additional...  ]

-------------------------------------------------------------------------

 [ Show Summary...  ]                        [ Reset Configuration ]

-------------------------------------------------------------------------
      [ Go! ]              [ Cancel ]             [Help]
```

The *Advanced Installation* menu lets you choose from among the menu tab areas with the ability of going back and forth among them until you are satisfied with your choices.

Across the top of the menu display are five tab areas: *Basic, Software, System, File System*, and *Advanced*. By pressing the tab key, each Tab area can be highlighted. To select the highlighted tab area, press the Enter/Return key. This will cause that tab area's screen to be displayed. Within each of these areas are several parameters that can be modified for your specific system. Listed below are the main features of each tab area:

- *Basic* - configuration and environment information.
- *Software* - ability to choose optional software to be installed. Mostly the same options that appear under *Guided Installation*.
- *System* - networking parameters. Also configurable via the **set_parms** command.
- *File System* - disk space allocation.
- *Advanced* - advanced disk, file system, logical volume, and volume group parameters.

We configure our system beginning with the *Basic* screen as shown in Figure 2-1.

Items of particular importance are discussed below:

- Configuration - we use *HP-UX B.11.11 Default*. HP-UX 11i is called by its original name, 11.11, in some cases.
- Environments - there are four 11i environments to select from at the time of this writing. This is a major change in the software distribution method for HP-UX 11i. The *Operating Environments* are bundles of software that make installing 11i easier. We'll select the first of the four, which is the top level, or most elaborate, of the *Operating Environments* as shown in the following example:

  ```
  Mission Critical OE-64bit .>] (HP-UX B.11.11)
                          Enterprise OE-64bit
                          HP-UX 11i OE-64bit
                          HP-UX 11i Base OS-64bit
  ```

- Root Disk - the default selection for the root disk is the first internal disk drive, which is the disk with a path of 0/0/1/1.2.0 in our example.
- File System - we are given the option of choosing wholedisk (not LVM) with HFS, Logical Volume Manager with HFS, or Logical Volume Manager with VxFS. If you have not reviewed the Logical

Volume Manager section of this chapter, you will want to do so before you make this selection (LVM is Logical Volume Manager). I am a strong advocate of using Logical Volume Manager with VxFS whenever possible.

- Root Swap - the system automatically selects an amount twice the size of your main memory, or a maximum of 1024 MBytes. You will want to consider your primary swap space very carefully. The rp54xx system (L-Class) in our example has 2 GBytes of memory and we'll go with the default of 1 GByte of swap as shown.

- Languages - We'll install English on our system; however, there are many languages available for 11i systems.

- Additional - this is the pick at the bottom right corner of the screen, not the tab area. Here is where you can configure such things as a second swap area, adding a second disk drive to the root volume, and disabling DHCP. With 11i, DHCP, Dynamic Host Configuration Protocol, works with an Ignite-UX server that automatically assigns system name, IP address, and so on.

The major difference between the selections just discussed for 11i and earlier releases of 11.x are the *Operating Environments.* 11i is the first HP-UX release for which *Operating Environments* are available.

Moving to the *Software* tab area, we find software on the installation CD-ROM for 11i that has been marked for installation. Since we have selected the *Mission Critical Operating Environment,* there is a lot of software automatically selected as part of this installation.

Figure 2-2 shows some of the software we have selected in the *Software* tab area. We have selected *All* so that we can see all of the software that has been selected as part of the *Mission Critical Operating Environment*. We could scroll down to see additional software.

Figure 2-2 *Software* Tab Area

```
                    /opt/ignite/bin/itool

/-------\/----------\/--------\/-------------\/----------\
| Basic ||  Software|| System || File System || Advanced |
\--------/          \-----------------------------------/

 Category         Marked?   Product       Description
-------------------------------------------------------------
 All               Yes       GigEther-00  PCI/HSC/EFF-HSCGigEther
 HPUXAdditions     Yes       HPUXBaseAux  HP-UX Base OS Auxilary
 OrderedApps       No        HyprFabrc-00 PCI/HSC HyperFabric Driver
 Uncategorized     No        J4258BA      Netscape Directory Server
                   No        J4274AA      HP WebQoS Peak Packaged Ed
                   No        RAID-00      PCI RAID; Supptd HW=A5856A
                   No        TermIO-00    PCI MUX; Supptd HW=J3592A


 [ Change Depot Location... ]

 ---------------------------------------------------------------------

 [ Show Summary... ]                            [ Reset Configuration ]

 ---------------------------------------------------------------------
      [ Go! ]              [ Cancel ]                [Help]
```



Other software that you may want to install can be selected from the installation CD. In the example the first two items have been selected automatically as part of the *Mission Critical Operating Environment*. At this point, we could select additional software, such as *Hyperfabric* and *Netscape Directory Server* in this window. Keep in mind that the software shown in this window is on the core operating system CD-ROM for 11i. You may later want to install application software from the HP-UX Applications CD set. This is done using Software Distributor. An overview of the Software Distributor product used for installing all HP-UX 11i software appears later in this chapter. You may want to take a look at this overview to get a feeling for the type of functionality that Software Distributor offers. The **swinstall** program is the

Software Distributor program used to install software. If you have application software to be installed, you will interact with **swinstall** and possibly be asked for codeword information for some of the software to be installed. If your software is protected, you will have to enter the codeword information. If you need a codeword, it should be printed on the CD-ROM certificate you received with your software. This codeword is tied to the ID number of a hardware device in your system.

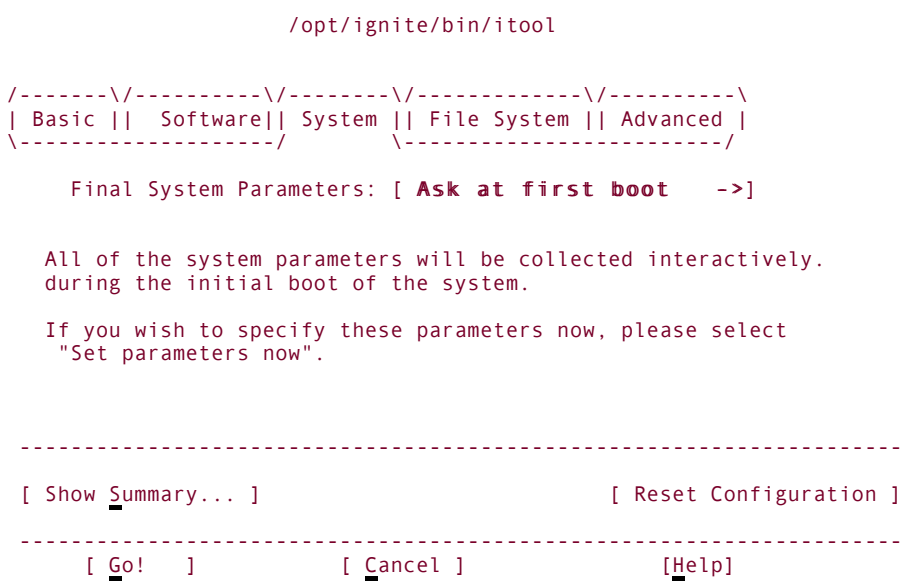
We'll select some additional software components so we can view them after installation has completed. With the software we wish to load on our system having been selected, we can move on to the next area.

The *System* tab area, shown in Figure 2-3, is where system identification-related configuration information can be found. Since we want to configure networking and related information after the installation is complete, we have changed only the first item on this screen. The options for the first item are:

Final system parameters:	[Set parameters now]
	[Ask at first boot]

We selected *Ask at first boot*.

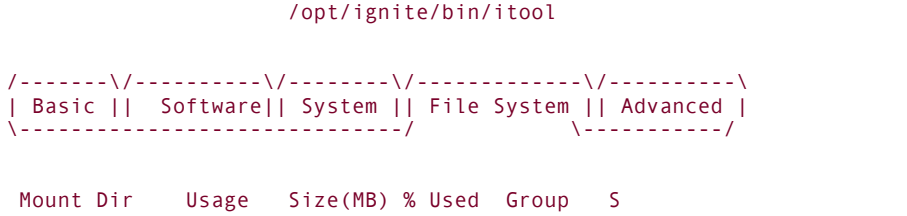
Figure 2-3 *System* Tab Area



When the system first boots, we'll be asked to enter system identification-related information.

The *File System* tab area, shown in Figure 2-4, is of particular importance. Here is where you can change file system sizes. You will not normally be satisfied with the default sizes of some of the logical volumes. I normally spend some time in this tab area increasing the sizes of some of the logical volumes. Figure 2-4 also shows the layout of the *File System* screen and the values of three of the logical volumes with their updated sizes. Root is highlighted in this example, so the parameters related to it appear under *Usage, Size,* and so on.

Figure 2-4 *File System* Tab Area



```
/stand       HFS     300      7       vg00    F ^ [ Add    ]
primary      SWAP+D  1024     0       vg00    R   [ Modify ]
/              VxFS     400       9        vg00     F v [ Remove ]

Usage:  [ VxFS      ->]   Group: [ vg ->]   Mount Dir: /

Size:  [ Fixed MB      ->]  400   Avail: 13360 MB
[ Add/Remove Disks...  ]    [ ---- Additional Tasks ---- ->]

---------------------------------------------------------------------

[ Show Summary... ]                          [ Reset Configuration ]

---------------------------------------------------------------------
     [ Go! ]              [ Cancel ]             [Help]
```

To make logical volume size changes, you select the mount directory of the logical volume, tab down to *Size,* and enter the desired new size. In addition to *Size,* there are several other parameters related to the logical volume that you can change.

You will notice that *Avail* shows you how much disk space is left to be allocated on your disk drive. It is perfectly all right to leave some disk space unallocated. This will give you a cushion for when you need to increase disk space down the road.

After making all the volume size-related modifications, we are ready to go ahead and install the system. However, first we want to choose the *Show Summary* option toward the bottom of the screen. This option will show us a summary of all the changes we made. This gives us a chance to make sure that we didn't forget something. Figure 2-5 shows the *General Summary* screen.

Figure 2-5 *General Summary* Screen

```
                     Summary View

/---------------\/---------------------\
|General Summary ||Hardware Inventory |
\               \---------------------/


                                Operating System: HP-UX B.11.11
Product                    Description
-------                    -----------
CDE-English                English CDE Environment             ^
FDDI-00                    PCI FDDI;Supported HW=A3739A/A3739B
GigEther-00                PCI/HSC/EFF-HSC GigEther,Supptd HW= v
---------------------------------------------------------------------

Grp Name  Usage     Size(MB)  % Used  Mount Dir
--------  -----     --------  - ----  ---------
vg00      HFS       300       7       /stand                    ^
vg00      SWAP+D    1024      0       primary
vg00      VxFS      400       9       /
vg00      VxFS      64        0      /tmp                       v

---------------------------------------------------------------------

 [   OK   ]                                          [  Help  ]
```

This screen provides information on the software we have selected to load and information on our logical volumes.

Figure 2-6 shows the *Hardware Inventory Summary* screen.

Figure 2-6 *Hardware Inventory Summary* Screen

```
                    Summary View

/---------------\/--------------------\
|General Summary ||Hardware Inventory |
\---------------/\                    \

Hardware Summary:         System Model: 9000/800/L2000-44       ^
+--------------------+----------------+-------------------+
| Disks: 2  ( 33.9GB) |  Floppies: 0   | LAN cards:   2    |
| CD/DVDs:        1   |  Tapes:    0   | Memory:   2048Mb  |
| Graphics Ports: 0   |  IO Buses: 8   | CPUs:        2    |
+--------------------+----------------+-------------------+

Disk Drives:
H/W Path              Capacity(Mb)     Model
0/0/1/1.2.0            17366           SEAGATE_ST318203LC
0/0/2/0.2.0            17366           SEAGATE_ST318203LC

                                                          v
F<                                                             >G


-------------------------------------------------------------------

 [   OK   ]                                          [  Help  ]
```

The *Hardware Inventory Summary* screen information provides a short summary of system hardware.

Since we are satisfied with all of the modifications we have made we are ready to load the operating system. We choose *Go!* which appears at the bottom of all the tab area screens and the screen in Figure 2-7 appears:

Figure 2-7 *Go!* Screen

```
                   itool Confirmation

All data will be destroyed on the following disks:

Addr         Disk Size(MB)  Description
----         -------------  -----------
0/0/1/1.2.0  17366 MB       SEAGATE_ST318203LC


The results of the pre-install analysis are:


WARNING: The disk at: 0/0/1/1.2.0 (SEAGATE_ST318203LC)
appears to contain a file system and boot area.  Continuing
the installation will destroy any existing data on this
disk.



-------------------------------------------------------------------
   [  Go!  ]              [ Cancel ]             [Help]
```

This screen warns us that there is an operating system already present on our target disk, but we want to proceed with the installation anyway and select *Go!*. The load of the HP-UX is automatic at this point and you can come back in an hour or so to check the log file and see if loading the operating system completed successfully.

It may be that your HP-UX 11i *Operating Environment* requires a second CD-ROM to complete the installation, as is the case with our *Mission Critical Operating Environment*. As shown in the upcoming listing, you'll be prompted to load the second CD-ROM:

```
=====================================================================
                     USER INTERACTION REQUIRED:

To complete the installation you must now insert the
```

```
"MC_OE" CD.

Once this is done, press the <Return> key to continue:

       * Starting swinstall of the source (MC_OE).
       * Running command: "/usr/sbin/swinstall -s/tmp/ign_configure/SD_CDROM
         -f/tmp/ign_configure/software_file  -x os_release=B.11.11 -x
         os_name=HP-UX:64 "
```

When the operating system load is complete we're asked for the system identification-related information we did not earlier enter. The following listing shows the first of these screens:

```
_____________________________________________________________________________

                      Welcome to HP-UX!

Before using your system, you will need to answer a few questions.

The first question is whether you plan to use this system on a network.

Answer "yes" if you have connected the system to a network and are ready
to link with a network.

Answer "no" if you:

     * Plan to set up this system as a standalone (no networking).

     * Want to use the system now as a standalone and connect to a
       network later.
_____________________________________________________________________________

Are you ready to link this system to a network?

Press [y] for yes or [n] for no, then press [Enter]
```

If you choose to connect your system network, you'll be asked a variety of questions about the networking configuration. You don't have to enter this information now because there is a command called **set_parms** that we'll cover in the next section, which is an alternative way of entering system identification-related information. We'll cover this command in the next section.

Let's now move to the next section, in which we'll specify the system-related information we have put off during our installation.

After we specify the system-related information, we'll come back to the software we just loaded so that you can see what was loaded and the components of which the *Mission Critical Operating Environment* is comprised.

[Team LiB]

PREVIOUS NEXT



Setting the System Parameters after Booting

When the system comes up after installation, a series of windows appear that allow you to configure your system name, time zone, root password, Internet Protocol (IP) address, subnet mask, and other networking settings (IP address and subnet mask background are provided in Chapter 12). One of the first questions you will be asked is whether or not you wish to use DHCP to obtain networking information. Dynamic host configuration protocol works with an Ignite-UX server that automatically assigns system name, IP address, and so on. Since our installation does not use this, I answered "no" which means that we'll have to enter all of our information manually.

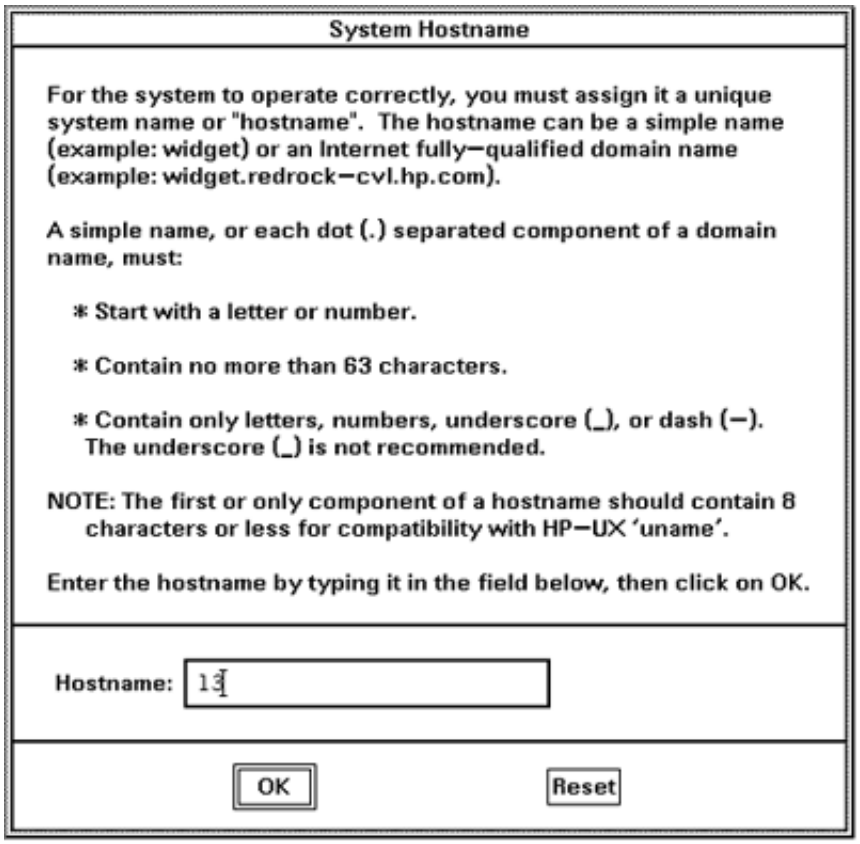
The system-specific information to be entered next can also be entered, after your system boots, by running **/sbin/set_parms**. This program can be used to set an individual system parameter or all the system parameters that would be set at boot time. **/sbin/set_parms** uses one of the arguments in Table 2-3, depending on what you would like to configure.

Table 2-3. /sbin/set_parms Arguments

set_parms Argument	Comments
Hostname	Set hostname.
Timezone	Set time zone.
date_time	Set date and time.
root_passwd	Set root password.
ip_address	Set Internet Protocol address (see Chapter 12 for networking background).
addl_network	Configure subnet mask, Domain Name System, and Network Information Service.
Initial	Go through the entire question-and-answer session you would experience at boot time.

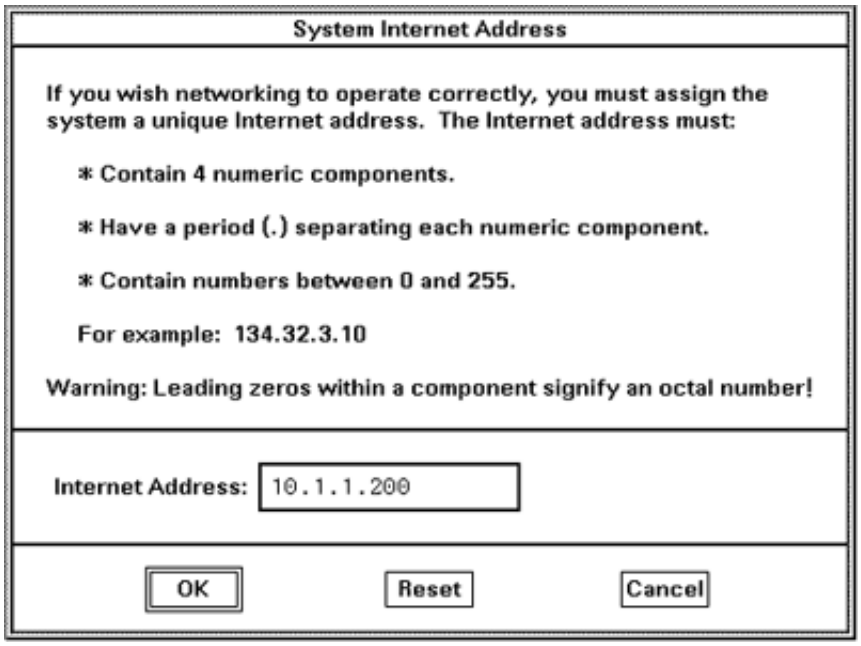
If you use the **initial** argument, you'll interact with a variety of dialog boxes asking you for information. The System Hostname dialog box is shown in Figure 2-8.

Figure 2-8. Entering Hostname on Series 800 with set_parms



You'll then be asked for your time zone and root password. Figure 2-9 shows the dialog box for entering your IP address:

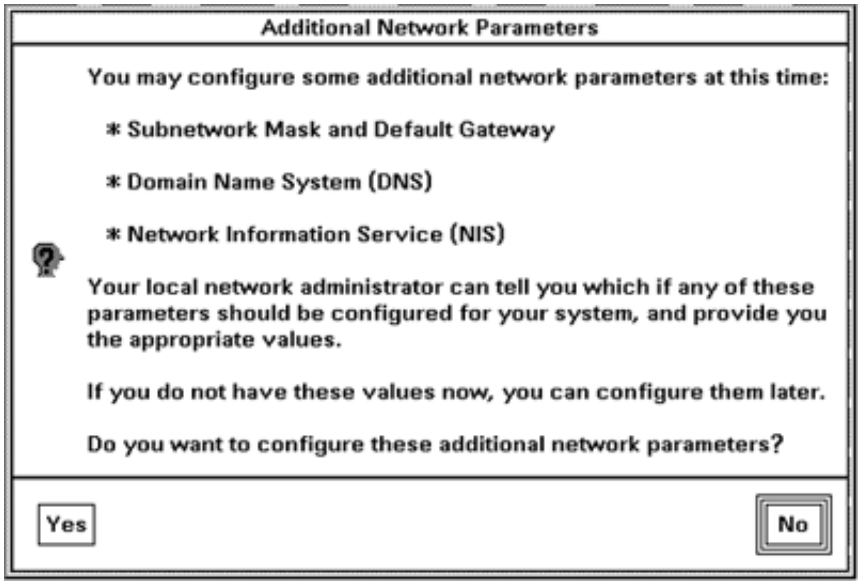
Figure 2-9. Entering IP Address on Series 800 with set_parms



You can then configure your subnet mask and other networking parameters.

Configuration includes the items shown in Figure 2-10:

Figure 2-10. Additional Configuration with set_parms setparms3.bmp



Running **set_parms initial** or **set_parms** with other options allows you to specify all of the initial information related to your system setup. This saves you the trouble of finding all of the relevant files in which you'd have to place the information you provide to **set_parms**.

Among the information that is updated is the contents of **/etc/rc.config.d/netconf**, which contains the IP address of the LAN card, netmask, and default router we just entered. The **/etc/hosts** file is updated with the name and IP address of this host. These files and their configuration are described in Chapter 12 which covers networking.

[Team LiB]

PREVIOUS NEXT

Software Distributor Example

Before we load additional software, let's take a look at the software that was loaded as part of our initial installation. We loaded the *Mission Critical Operating Environment* and some additional software products. We'll check the operating system revision with **uname -a** and software bundles that have been loaded on the system with **swlist** in the following listing:

[View full width]

```
# uname -a
HP-UX l3 B.11.11 U 9000/800 104911547 unlimited-user license
# swlist
# Initializing...
# Contacting target "l3"...
#
# Target:  l3:/
#

#
# Bundle(s):
#

  100BaseT-01           B.11.11.01     HP-PB 100BaseT;Supptd HW=A3495A;SW=J2759BA
  B5725AA               B.3.5.89       HP-UX Installation Utilities (Ignite-UX)
  BUNDLE11i             B.11.11.0102.2 Required Patch Bundle for HP-UX 11i
  Base-VXVM             B.03.20.1      Base VERITAS Volume Manager 3.2 for HP-UX
  CDE-English           B.11.11        English CDE Environment
  FDDI-00               B.11.11.02     PCI FDDI;Supptd HW=A3739A/A3739B;SW=J3626AA
  FibrChanl-00          B.11.11.09     PCI/HSC FibreChannel;Supptd HW=A6684A,A6685A,A5158A
  GigEther-00           B.11.11.14     PCI/HSC GigEther;Supptd HW=A4926A/A4929A/A4924A/A49
  GigEther-01           B.11.11.01     PCI/PCI-X GigEther;Supptd HW=A6794A
  HPUX11i-OE-MC        B.11.11.0112   HP-UX Mission Critical Operating Environment
➥ Component
  HPUXBase64            B.11.11        HP-UX 64-bit Base OS
  HPUXBaseAux           B.11.11.0112   HP-UX Base OS Auxiliary
  HWEnable11i         B.11.11.0112.5 Hardware Enablement Patches for HP-UX 11i, December
➥ 2002
  Ignite-UX-11-11     B.3.5.89     HP-UX Installation Utilities for Installing 11.11
➥ Systems
  OnlineDiag            B.11.11.05.10  HPUX 11.11 Support Tools Bundle, Dec 2001
  RAID-00               B.11.11.01     PCI RAID; Supptd HW=A5856A
  perl                  B.5.6.1.C      Perl Programming Language
#
```

This listing shows that HP-UX 11i has indeed been loaded, shown as 11.11 in the listing, and that several software bundles that we selected have been loaded as well, including the *Mission Critical Operating Environment*.

There are many products loaded as part of the *Mission Critical Operating Environment*. The following **swlist** shows the components of which *Mission Critical Operating Environment* is comprised at the time of this writing:

[View full width]

```
# swlist -l product HPUX11i-OE-MC
# Initializing...
# Contacting target "l3"...
#
# Target:  dem4415f:/
#

# HPUX11i-OE-MC                        B.11.11.0112   HP-UX Mission Critical Operating
                                                      Environment Component
  HPUX11i-OE-MC.ATS-CORE               A.11.13        Service Guard Advanced Tape
                                                      Services
  HPUX11i-OE-MC.Workload-Mgr           A.01.02        HP-UX Workload Manager
  HPUX11i-OE-MC.WLM-Toolkits           A.01.00        HP-UX Workload Manager Toolkits
  HPUX11i-OE-MC.ServiceGuard           A.11.13        Service Guard
```

```
  HPUX11i-OE-MC.ServControlMgr          A.02.05        HP-UX ServiceControl Manager
  HPUX11i-OE-MC.SG-Sybase-Tool        B.01.07      MC/ServiceGuard Sybase Script Templates
  HPUX11i-OE-MC.SG-Sgosb-Tool         B.01.07      C. Cluster intgrtn w/ Oracle Standby DB
  HPUX11i-OE-MC.SG-Progress-Tool         B.01.07        MC/ServiceGuard Progress
                                                        Script Templates
  HPUX11i-OE-MC.SG-Oracle-Tool        B.01.07        MC/ServiceGuard Oracle Script
➥ Templates
  HPUX11i-OE-MC.SG-NFS-Tool            A.11.11.02     MC/ServiceGuard NFS Script Templates
  HPUX11i-OE-MC.SG-Informix-Tool         B.01.07        MC/ServiceGuard Informix
                                                        Script Templates
  HPUX11i-OE-MC.SG-Fndation-Tool         B.01.07        MC/ServiceGuard Foundation Bundle
                                                        Script Templates
  HPUX11i-OE-MC.SG-FasTrack-Tool         B.01.07        MC/ServiceGuard Netscape FastTrack
                                                        Script Templates
  HPUX11i-OE-MC.SG-Domain-Tool           B.01.07        MC/ServiceGuard HP Domain
                                                        Script Templates
  HPUX11i-OE-MC.SG-Db2-Tool              B.01.07        C. Cluster intgrtn w/ DB2 DB
  HPUX11i-OE-MC.Proc-Resrc-Mgr           C.02.00        Process Resource Manager
                                                        Proc-Resrc-Mgr product
  HPUX11i-OE-MC.Package-Manager          A.11.13        HP Package-Manager
  HPUX11i-OE-MC.PRM-Sw-Lib               C.02.00        Process Resource Manager
                                                        PRM-Sw-Lib product
  HPUX11i-OE-MC.PAM-NTLM             A.01.06      HP NTLM Pluggable Authentication Module
  HPUX11i-OE-MC.PAM-Kerberos             B.11.11        PAM-Kerberos Version 1.0
  HPUX11i-OE-MC.OnlineJFS            B.11.11       Online features of the VxFS File System
  HPUX11i-OE-MC.OVOPC-SE              A.05.57       IT/Operations Special Edition Product
  HPUX11i-OE-MC.OVOPC-SE-GUI            A.05.57       IT/Operations Special Edition JavaUI
  HPUX11i-OE-MC.OVOPC-SE-DOC             A.05.57        IT/Operations Special
                                                        Edition Documentation
  HPUX11i-OE-MC.OVOPC-DFLT               A.05.57        IT/Operations Special Edition
                                                        Default Instrumentation
  HPUX11i-OE-MC.OVOPC-AGT                A.05.57        IT/Operations Agent
  HPUX11i-OE-MC.OE                       B.11.11.0112   HP-UX OE control script product
  HPUX11i-OE-MC.NS-communicate           B.11.00.05     Netscape Communicator 4.75
  HPUX11i-OE-MC.MeasurementInt          C.03.50.00    HP-UX Measurement Interface for 11i
  HPUX11i-OE-MC.MeasureWare              C.03.50.00     MeasureWare Software/UX
  HPUX11i-OE-MC.LVM                      B.11.11        LVM
  HPUX11i-OE-MC.Java2RTE_perf            1.2.2.08B.02   Java 2 RTE (700/800), PA2.0
  HPUX11i-OE-MC.Java2RTE_doc           1.2.2.08B.02   Java 2 RTE (700/800), Distribution Lic
  HPUX11i-OE-MC.Java2RTE_base           1.2.2.08B.02   Java 2 RTE (700/800), PA1.1 Portable
  HPUX11i-OE-MC.Java-PlugIn1-2           1.2.2.08B.02   Plug-In for Java 2
  HPUX11i-OE-MC.Glance                   C.03.50.00     HP GlancePlus/UX
  HPUX11i-OE-MC.EMS-RdbmsMon             A.03.20.01     EMS Database Monitor
  HPUX11i-OE-MC.EMS-MIBMonitor           A.03.20.01     EMS MIB Resource Monitor Product
  HPUX11i-OE-MC.EMS-DiskMonitor          A.03.20.01     EMS Disk Resource Monitor
  HPUX11i-OE-MC.Cluster-OM               A.01.03        HP Cluster API
  HPUX11i-OE-MC.Cluster-Monitor          A.11.13        HP Cluster Monitor
  HPUX11i-OE-MC.CM-Provider-MOF          A.11.13        CM Provider and MOF
  HPUX11i-OE-MC.CIFS-Server              A.01.07        CIFS/9000 (Samba) File and Print
                                                        Services
  HPUX11i-OE-MC.CIFS-Development         A.01.07        CIFS/9000 server source code files
  HPUX11i-OE-MC.CIFS-Client              A.01.06        HP CIFS/9000 Client
  HPUX11i-OE-MC.ApacheStrong             1.3.19.21      HP Apache-based Web Server
                                                        with Strong (128bit) Encryption
#
```

The initial **swlist** output was deceiving in that the *Mission Critical Operating Environment* was shown as only one entry. You can see from the display of the *product* level details of *Mission Critical Operating Environment* in the last listing that many products included in this operating environment, including MC/ServiceGuard, Process Resource Manager, GlancePlus/UX, CIFS/9000, and many other products covered in this book.



Now that we know the initial software that has been loaded, let's move on to load additional products. Software Distributor-HP-UX (I'll call this Software Distributor throughout the book; HP documentation typically uses SD-UX) is the program used in HP-UX 11i to perform all tasks related to software management. Software Distributor will be used in an example to install software on the same system on which we loaded our operating system earlier in this chapter. Software Distributor is a standards-based way to perform software management. It conforms to the Portable Operating System Interface (POSIX) standard for packaging software and utilities related to software management. The Software Distributor product described in this section comes with your HP-UX system. Additional functionality can be obtained by buying the OpenView Software Distributor (SD-OV) product. SD-OV

provides support for additional platforms, allows you to push software out to target systems, features centralized monitoring, and provides a job browser to assist in managing software on target systems. In this section, I won't cover SD-OV but will make some comments about SD-OV functionality where appropriate.

Software Distributor can be invoked using the commands described in this section, by using SAM (covered in Chapter 10), or by installing software for the first time as described earlier in this chapter.

The following are the four phases of software installation performed with Software Distributor:

- Selection(1) - You can select the source and software you wish to load during this phase. In the upcoming example, the Graphical User Interface of Software Distributor is used and you'll see how easily you can select these.
- Analysis(2) - All kinds of checks are performed for you, including free disk space; dependencies; compatibility; mounted volumes; and others. One of the very useful outputs of this phase is the amount of space the software you wish to load will consume on each logical volume. This will be shown in the example.
- Load(3) - After you are satisfied with the analysis, you may proceed with loading the software.
- Configuration(4) - The software you load may require kernel rebuilding and a system reboot. Startup and shutdown scripts may also need to be modified.



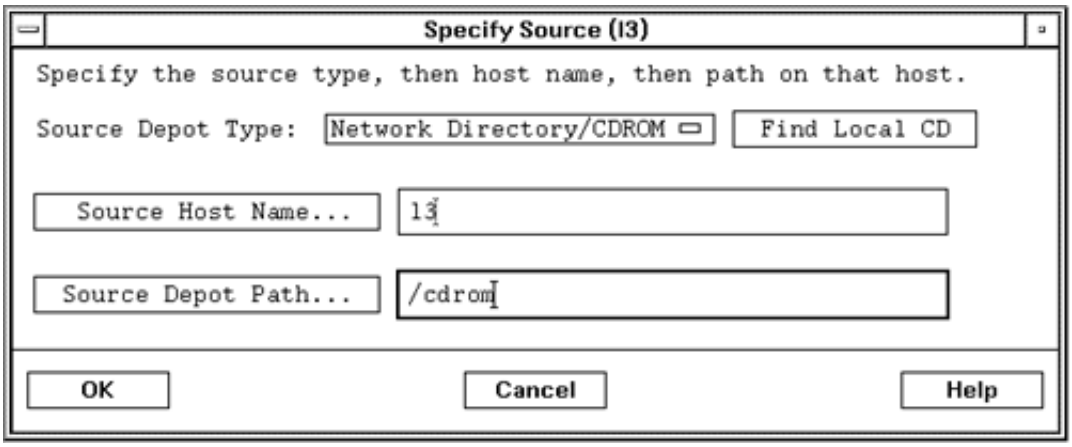
There is some terminology associated with Software Distributor that I tend to use somewhat loosely. I have nothing but good things to say about Software Distributor, but I don't tend to conform to the official Software Distributor terminology as much as I should. I tend, for instance, to use the word "*system*" a lot, which could mean many different things in the Software Distributor world. For instance, Software Distributor uses "*local host*" (a system on which Software Distributor is running or software is to be installed or managed by Software Distributor), "*distribution depot*" (a directory that is used as a place for software products), and "*development system*" (a place where software is prepared for distribution). I will use the word *system* to mean the system on which we are working in the examples, because software is loaded onto the system from media.

The example of Software Distributor in this section describes the process of loading software from CD-ROM or DVD to the local system. What I show here only begins to scratch the surface of functionality you have with Software Distributor, but since I want to get you up and running quickly, this overview should be helpful. You can load software from a variety of media as well as across the network. You can run **swinstall** through the graphical interface used throughout this section, the character user interface, or the command line. You can use the **swinstall** command from the command line specifying source, options, target, and so on. I would recommend using the character or graphical user interface because they are so much easier. If, however, you like to do things the "traditional UNIX" way, you can issue the **swinstall** command with arguments. You can look at the manual page for **swinstall** to understand its arguments and options and use this command from the command line. The graphical user interface of Software Distributor works with the **sd** (this is an SD-OV command and may also be invoked with **swjob -i**), **swcopy**, **swremove, swlist**, and **swinstall** commands. There is also an interactive terminal user interface for these commands if you don't have a graphics display.



On the rp54xx system (L-Class) used in the upcoming example, there wasn't much to do in order to load software from a CD-ROM. I just physically put the CD-ROM in the DVD drive and typed **swinstall** at the command line. To mount the DVD-ROM device manually, I issued the command **mount /dev/dsk/c0t3d0 /cdrom**. To find the disk devices on your system, including DVD-ROMs and CD-ROMs, you issue the command **ioscan -funC disk**. I was given the screen shown in Figure 2-11 after invoking **swinstall,** which shows the source for the software installation:

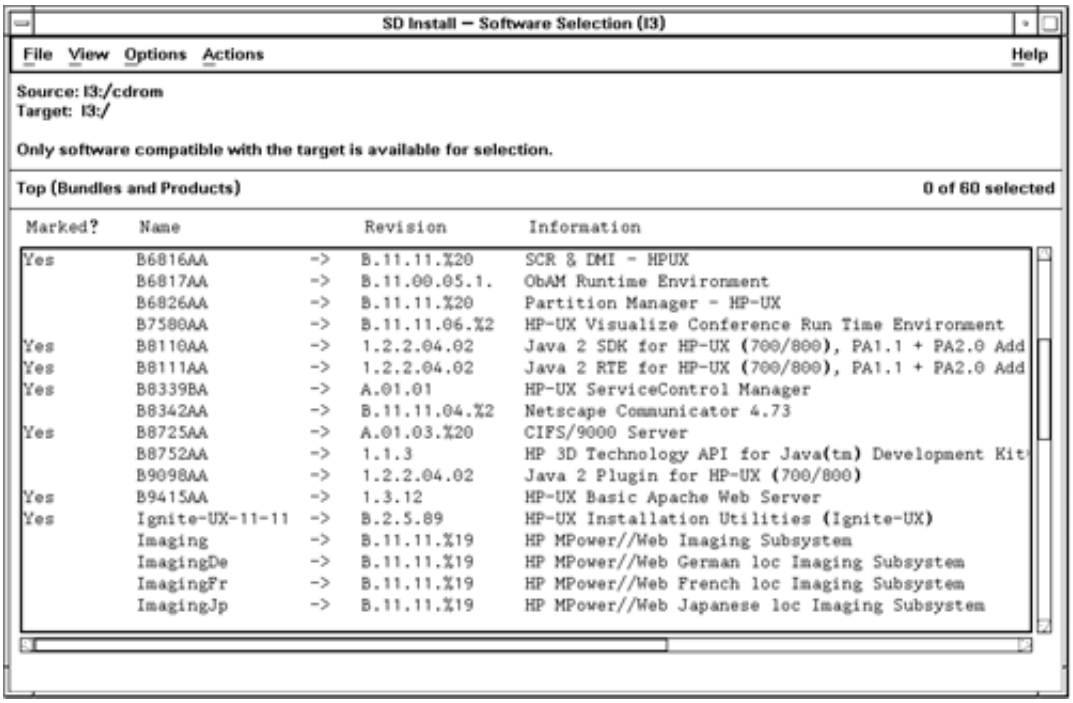
Figure 2-11. Specify Source Software Distributor Screen





swinstall filled in the information shown in the dialog box. I did not have to create the directory **/cdrom** or issue the **mount** command at the command line as I often had to do in past releases of HP-UX. **swinstall** filled in all of the information shown in Figure 2-12. The applications CD-ROM from which we'll be loading applications in this example was labeled *HP-UX Release 11.11 and Application Products*.

Figure 2-12. Software Selection Software Distributor Screen



After accepting the information shown in the figure, I proceeded to select (Step 1 - Selection) the software from the list that I wanted to load by "marking" it.

When selecting software to load, you may receive a "Yes" in the "Marked?" column or a "Partial." "Yes" means all of the filesets associated with your selection will be loaded, and "Partial" means only some will be loaded. Figure 2-12 shows "Yes" in the "Marked?" column for software that has been selected.

A bundle of software you select to install may be composed of products, subproducts, and filesets. You can select any item you have "Marked" for loading to see of what filesets it is comprised. I have done this for a Java products for 11i. I selected this *bundle* to see the software of which it is comprised in Figure 2-13:

Figure 2-13. Components of a Software Product

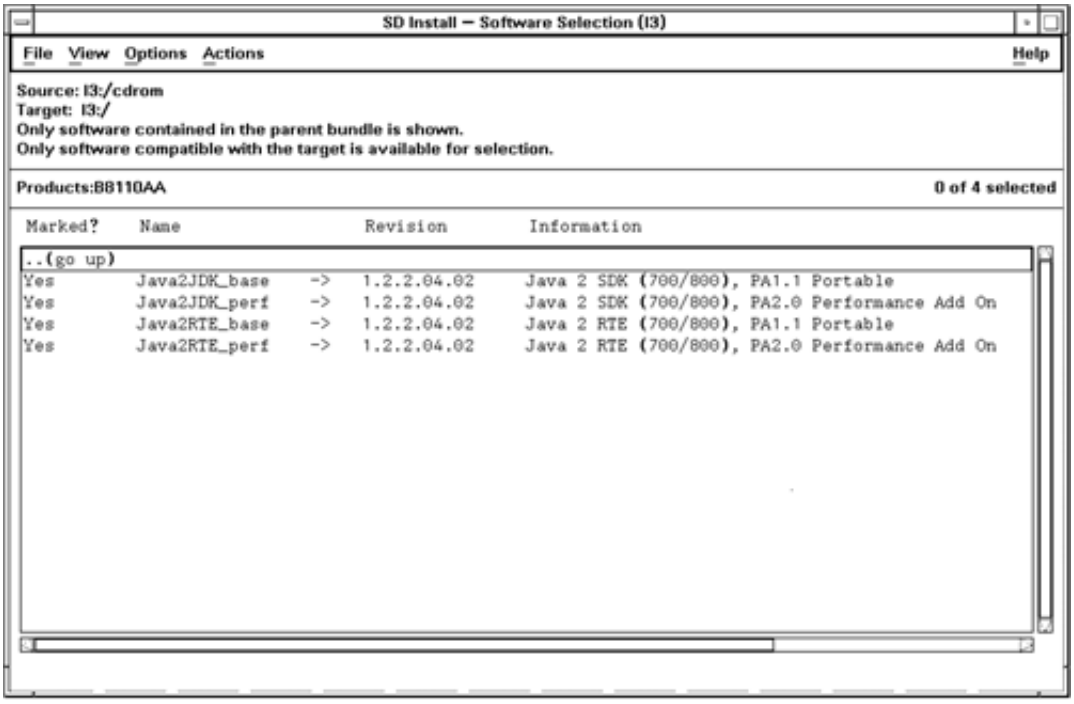
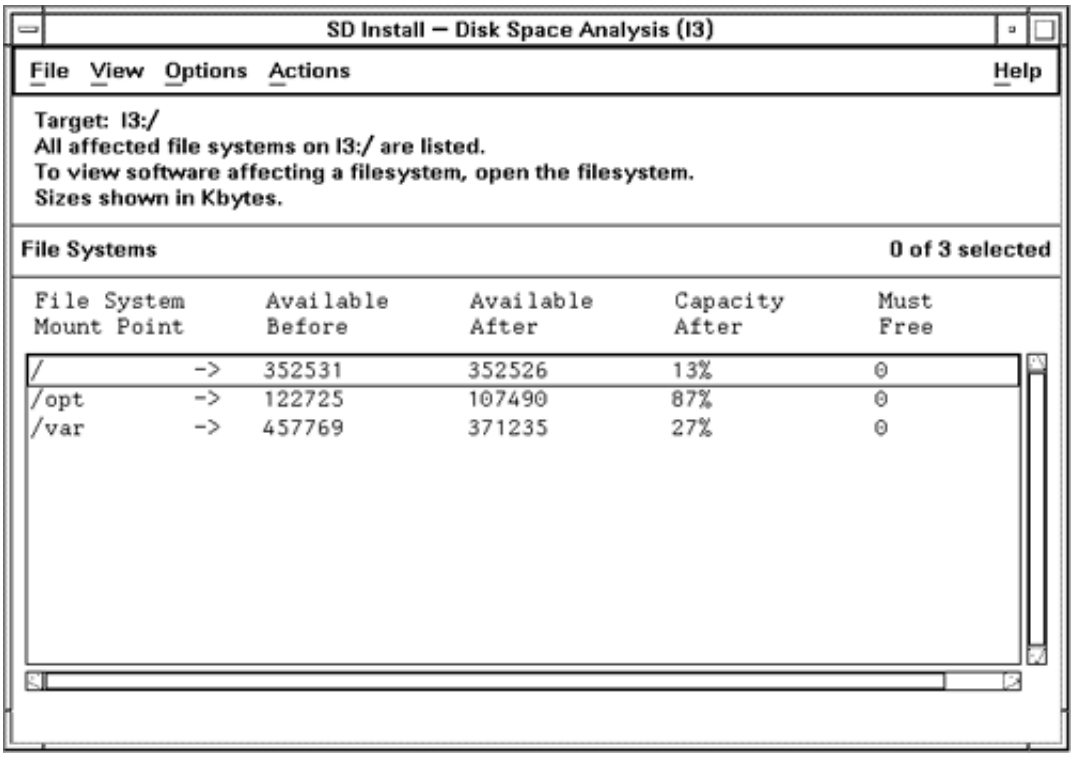


Figure 2-13 shows that the Java software is indeed composed of many components. To go back to the top, we select *(go up)*.

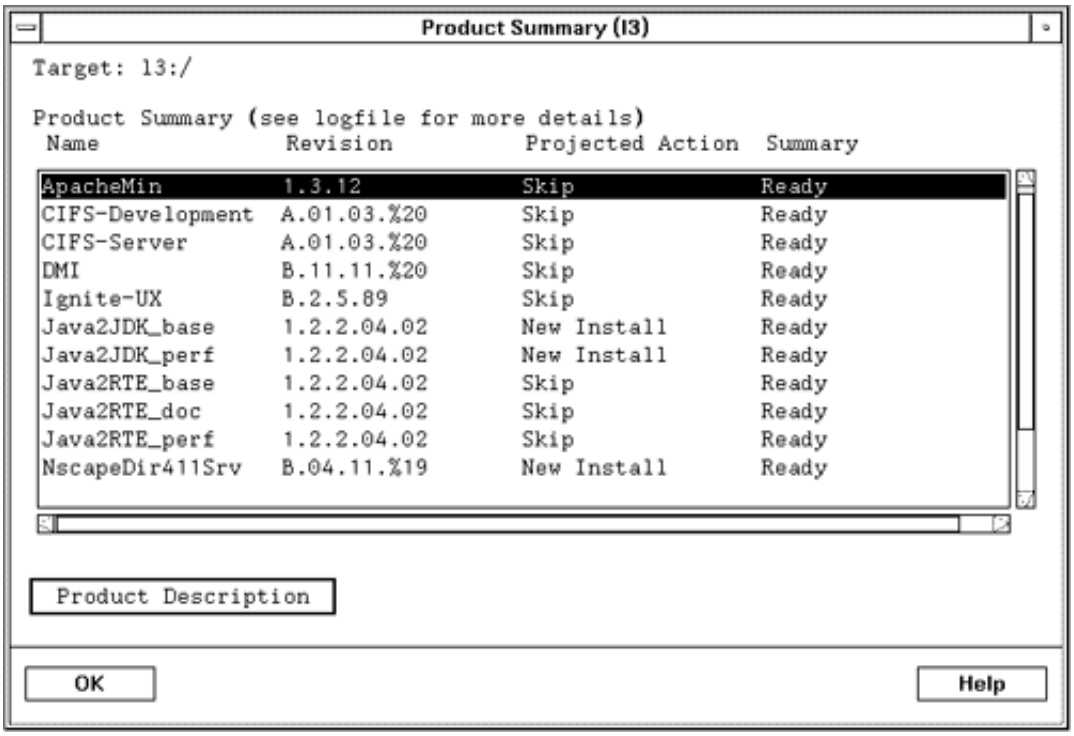
Selecting *Install* runs analysis (Step 2 - Analysis) on the software you have selected to load. After the analysis has been completed, you can take a look at the logfile, view the disk space analysis, and perform other tasks. I normally take a look at the disk space analysis just to see the impact the software I am loading is having on free disk space, as shown in Figure 2-14:

Figure 2-14. Disk Space Analysis Software Distributor Screen



I also look at the products to ensure that I'm loading the software I expect to load, as shown in Figure 2-15 for the products earlier marked for installation:

Figure 2-15. Product Summary Software Distributor Screen



After you are satisfied with the analysis, you can proceed with the installation (Step 3 - Load.) Figure 2-16 shows the type of status you are provided as software is loaded on the system:

Figure 2-16. Install Window Software Distributor Screen Showing Status



In this case, we have loaded about *37%* of the total software we have selected. The software currently being loaded is *Netscape Directory Server*. We could view the logfile to see the status of what has been loaded thus far.

After the load of your software is complete, you can either exit the session or you will be given a message indicating that a reboot will be required to complete the configuration (Step 4 - Configuration). Not all software requires a reboot in order for configuration to take place. Most software will run configuration routines after the software is loaded without the need for a reboot. If a reboot is required, you will be asked if you wish to reboot; but it could be a bad time to reboot, so you may want to wait, or delay the reboot.

The process of loading software from media using Software Distributor is easy and thorough. The user interface to Software Distributor makes it easy to select and load software from media.

[Team LiB]

PREVIOUS NEXT

Loading Patches

HP-UX patches can be obtained from a variety of sources. Most people start at *IT Resource Center (*us-support.external.hp.com*)* because a variety of useful HP-UX information can be found there using your browser.

When you set up your account initially with *IT Resource Center*, you will want to enter your system handle, and the serial number of one of your systems if you have it to get full access to all of the information on the site.

The process of viewing individual patches on *IT Resource Center* is self-explanatory when you log in to the site. Selecting *Individual Patches* and *HP-UX Patches* will allow you to select a specific release of HP-UX, such as 11.11 for HP-UX 11i, and select the patch(es) you wish to load on your system. You can then add the patches to your *Shopping Cart* and download them.

When you search *IT Resource Center* for a patch, you will be told if the patch has been replaced by a more recent patch. After selecting the patch you wish to view, a lot of useful information related to the patch will be available, including:

Patch Files

Dependencies

Supersedes

Size

Critical

Symptoms

Defect Description

Installation Instructions

I always view the installation instructions for a patch while on the web site to see if any special work is required to load the patch. The following are the installation instructions for installing a patch we'll download and install in our example

```
1. Back up your system before installing a patch.

                  2. Login as root.

                  3. Copy the patch to the /tmp directory.

                  4. Move to the /tmp directory and unshar the patch:

                  cd /tmp
                  sh PHCO_21187

                  5. Run swinstall to install the patch:

                  swinstall -x autoreboot=true -x patch_match_target=true \
                    -s /tmp/PHCO_21187.depot

                  By default swinstall will archive the original software in
                  /var/adm/sw/save/PHCO_21187.  If you do not wish to retain a
                  copy of the original software, use the patch_save_files option:

                  swinstall -x autoreboot=true -x patch_match_target=true \
                    -x patch_save_files=false -s /tmp/PHCO_21187.depot

                  WARNING: If patch_save_files is false when a patch is installed,
```

```
                the patch cannot be deinstalled.  Please be careful
                when using this feature.

              For future reference, the contents of the PHCO_21187.text file is
              available in the product readme:

              swlist -l product -a readme -d @ /tmp/PHCO_21187.depot

              To put this patch on a magnetic tape and install from the
              tape drive, use the command:

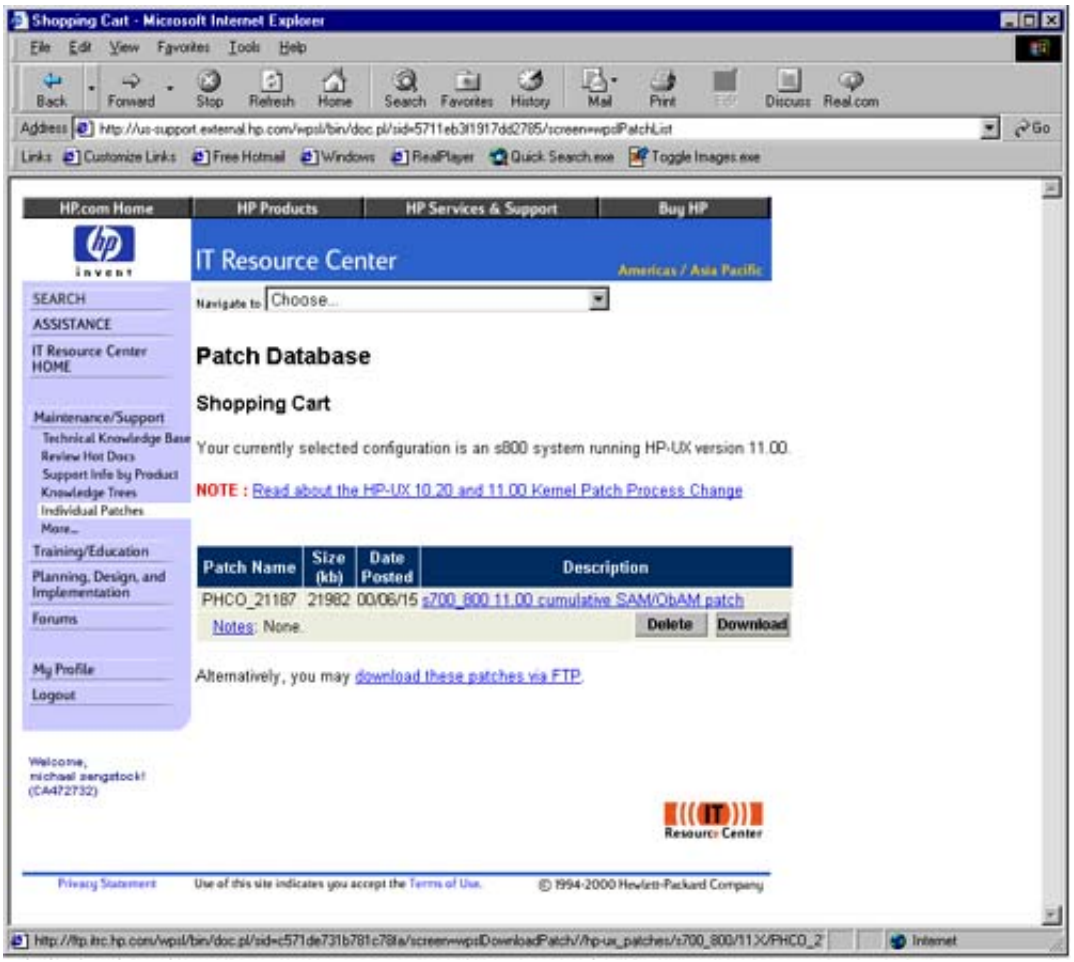
              dd if=/tmp/PHCO_21187.depot of=/dev/rmt/0m bs=2k

Special Installation Instructions: None
```

In the installation procedure for many patches, a reboot is required. This patch has the *autoreboot* option equal to *true,* meaning that a reboot will automatically take place if indeed a reboot is required. With HP-UX 11i, there are many more *Dynamic Patches* being introduced, which means that a reboot is not required when the patch is installed. You will see more and more patches for which a reboot is not required. Combined with *Dynamically Loadable Kernel Modules* and *Dynamically Tunable Kernel Parameters,* there will be fewer and fewer reboots required of 11i systems.

After reviewing the information related to the patch, we'll download it using a browser by selecting the patch and *Add to Cart*. Figure 2-17 shows a patch that has been placed in the *Shopping Cart* and is ready for download:

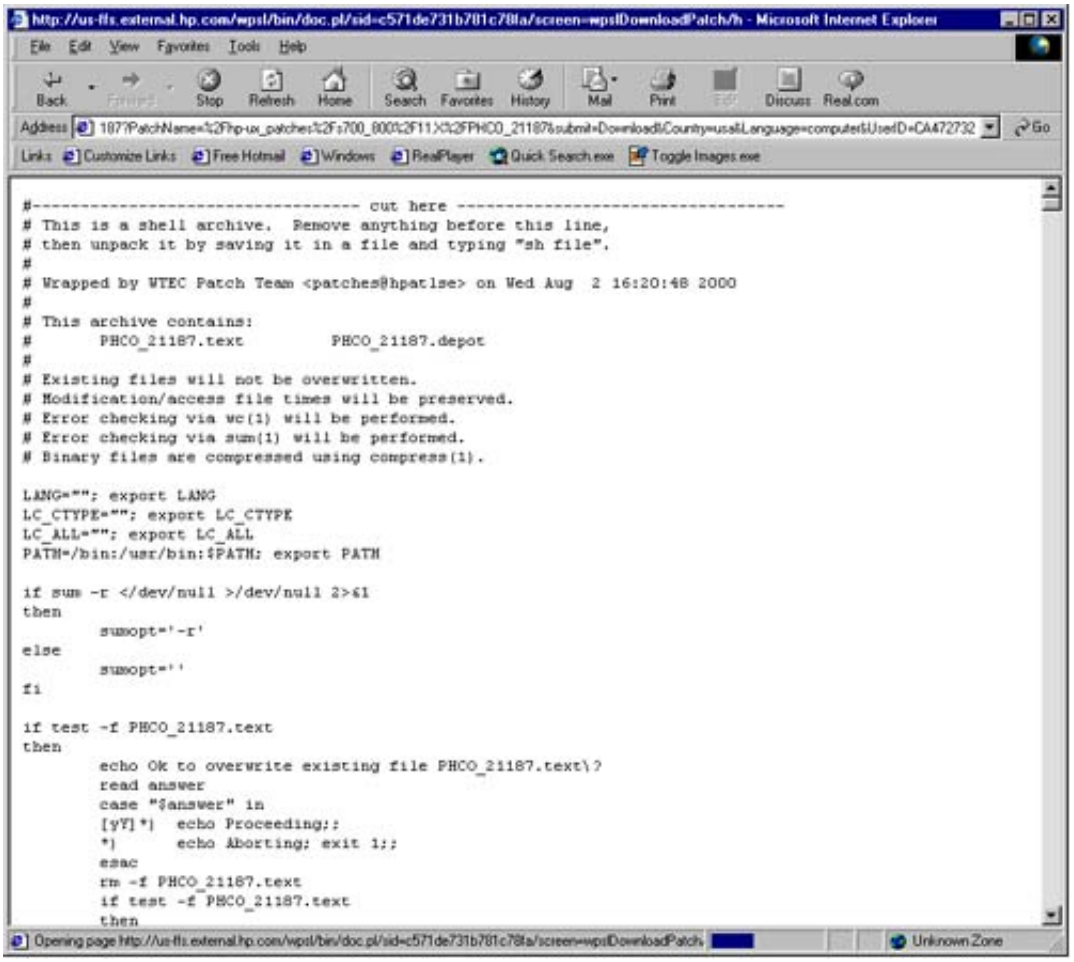
Figure 2-17. Patch in *Shopping Cart* Ready to Download



Note at the bottom of the figure that you can select to download the patch(es) with FTP instead of through the browser.

Figure 2-18 shows a patch that is in the process of being downloaded through the browser:

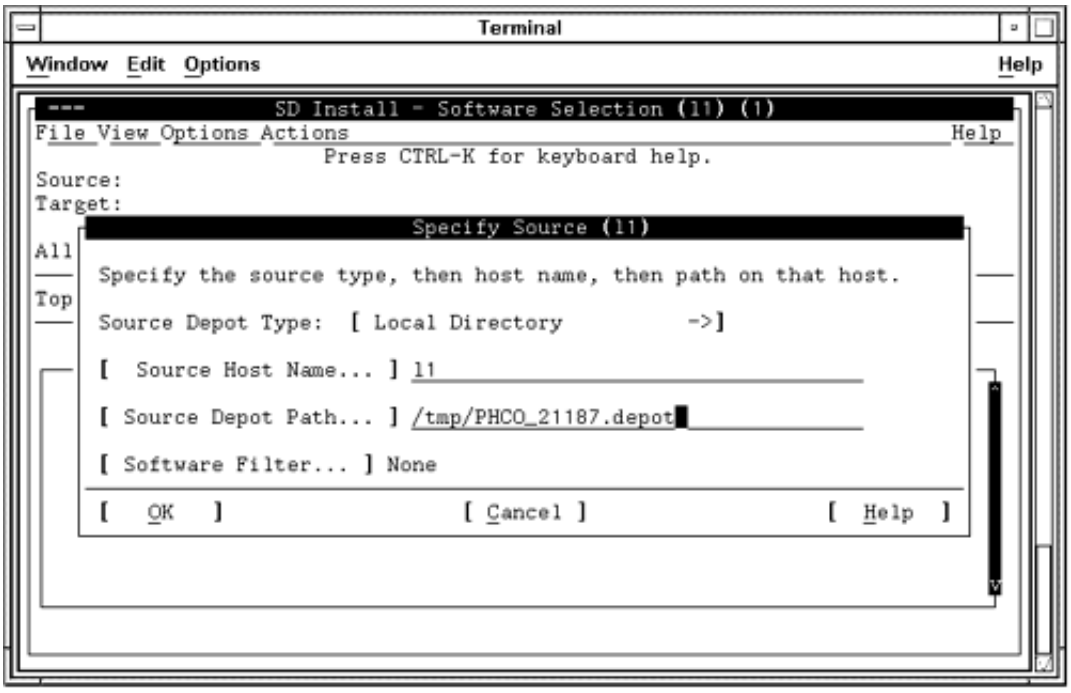
Figure 2-18. Patch Download



Once the patch is downloaded, you simply follow the instructions for installing the patch, such as those that were shown earlier for the patch in our example. You normally download a patch into the **/tmp** directory and run **sh** against the downloaded file, which produces *patchname***.depot**. You can then run *Software Distributor* commands on the command line as shown in the instructions.

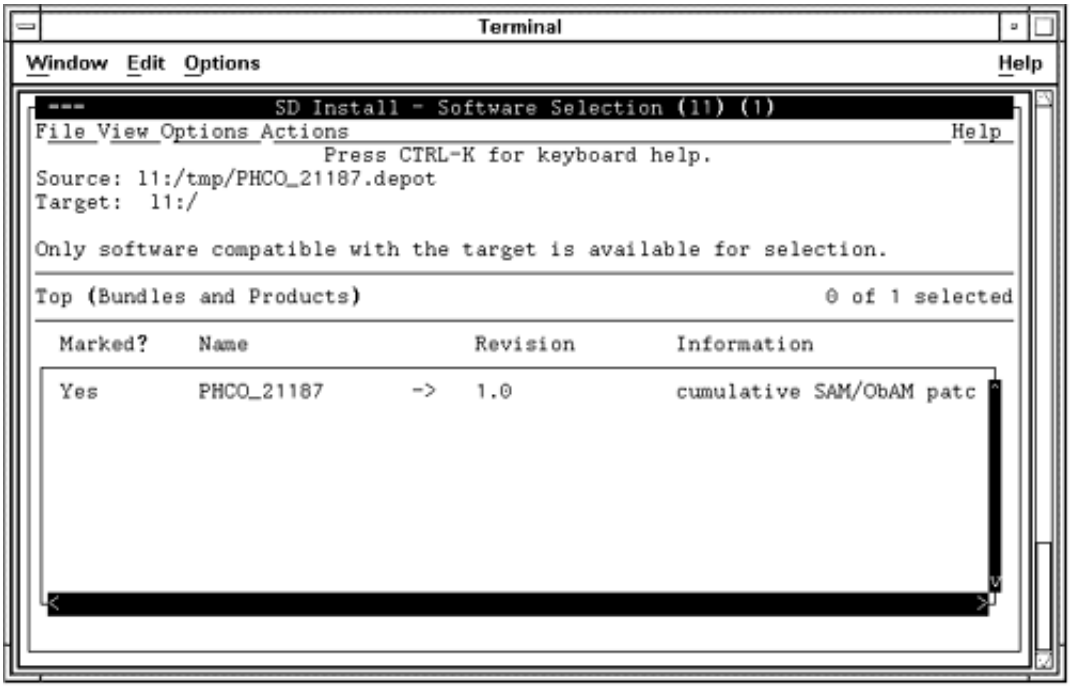
For the patch in our example, I've run **sh** against the file downloaded, which produced a **.depot** file. We'll run **swinstall** to invoke the user interface. Figure 2-19 shows specifying the source as a *Local Directory* rather than a *Network Directory/CDROM,* as in the earlier example in this chapter, and the *Source Depot Path* of our **.depot** file in the **/tmp** directory:

Figure 2-19. *Specify Source* of Patch



We would then mark the patch for installation as shown in Figure 2-20:

Figure 2-20. Patch Marked for Installation





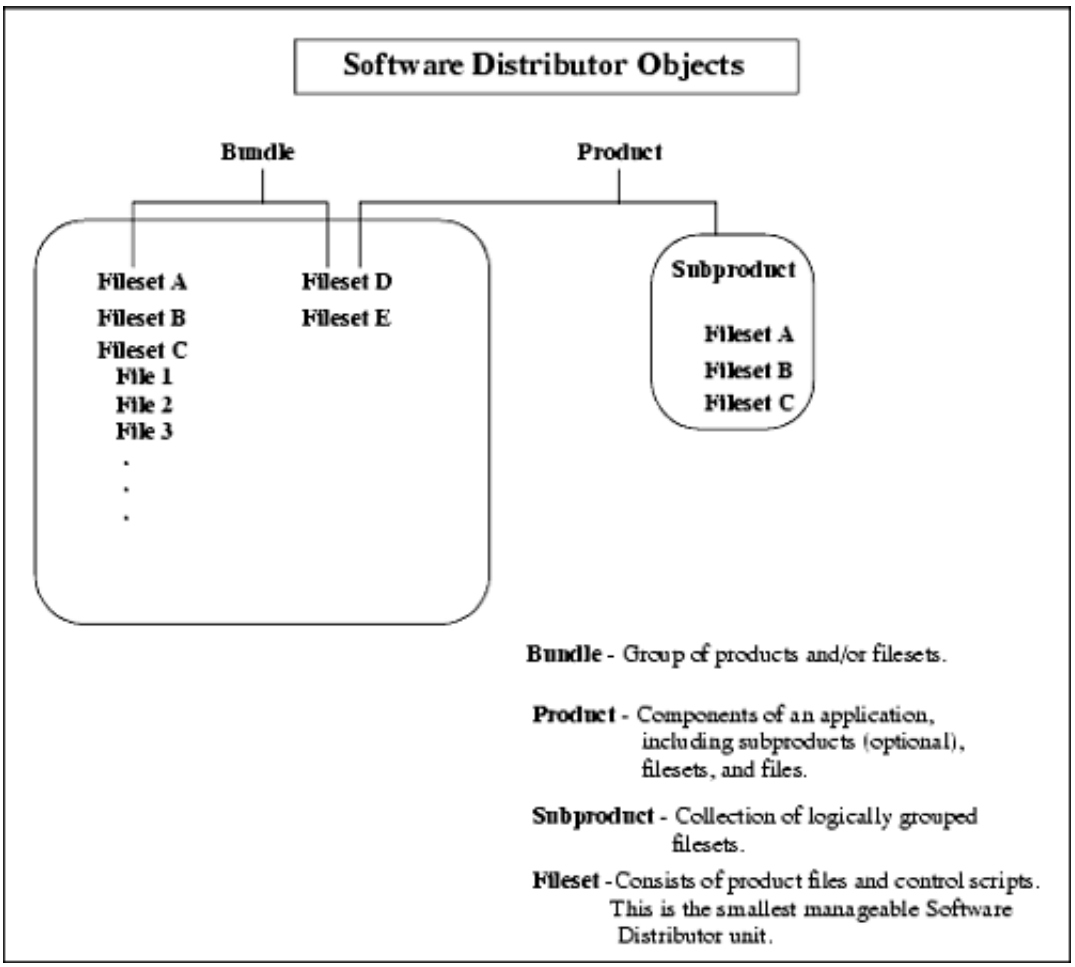
You can load patches using **swinstall** interactively as I have done in this example or by following the instructions embedded in the patch and running **swinstall** from the command line.

Software Distributor Background



You need to have some background on the way software is organized in Software Distributor. There are the four following types of objects into which software is grouped in Software Distributor; bundle, product, subproduct, and fileset. Figure 2-21 shows the hierarchy of Software Distributor objects.

Figure 2-21. Software Distributor Objects



You can look at the bundle in Figure 2-21 as a group of software. This can be products, subproducts, and filesets, as shown in the diagram. The concept here is to organize software in such a way that it is easy to manage. The diagram shows that some filesets are shared between different bundles and products. This hierarchical organization and sharing makes managing software with Software Distributor flexible.

Here are some of the common software management-related tasks you can perform with Software Distributor.

Installing and Updating Software (Command Line or GUI)



The **swinstall** command is used to install and update software. The source of the software you are loading can come from a variety of places, including, CD-ROM, magnetic tape, or a "depot" directory from which software can be distributed. Using the depot, you can load software into a directory and then install and update software on other nodes from this directory. Software loaded from CD-ROM with Software Distributor must be loaded onto the local system; this technique is used in the upcoming example. You have a lot of flexibility with SD-OV when selecting the target system onto which you want to load software and the source from which you will load the software. You can, for instance, load software from a depot that is on another system on your network. This command can be run at the command line or with the Graphical User Interface.

Copying Software to a Depot (Command Line or GUI)

The **swcopy** command is used to copy software from one depot to another. The depot used in the upcoming examples is a CD-ROM. By setting up depots, you can quickly install or update software to other nodes simultaneously with SD-OV. This command can be run at the command line or with the Graphical User Interface.

Removing Software from a System (Command Line or GUI)



The **swremove** command is used to remove software from a system that has had software loaded with Software Distributor. This includes removing installed and configured software from a system or removing software from a depot. This command can be run at the command line or with the Graphical User Interface.

List Information about Installation Software

The **swlist** command provides information about the depots that exist on a system, the contents of a depot, or information about installed software. Examples of using this command are provided shortly. This command can be run at the command line or with the Graphical User Interface.

Configure Installed Software

The **swconfig** command configures or unconfigures installed software. Configuration of software normally takes place as part of **swinstall,** but configuration can be deferred until a later time.

Verify Software

The **swverify** command confirms the integrity of installed software or software stored in a depot.

Package Software That Can Later Be Installed (Local Sys Only)

You may want to produce "packages" of software that you can later put on tape or in a depot with the **swpackage** command. This packaged software can then be used as a source for **swinstall** and be managed by other Software Distributor commands.

Control Access to Software Distributor Objects

You may want to apply restricted access to Software Distributor objects such as packaged software. Using the **swacl** command, you can view and change the Access Control List (ACL) for objects.

Modify Information about Loaded Software (Local System Only)

The Installed Products Database (IPD) and associated files are used to maintain information about software products you have loaded. **swmodify** can be run at the command line to modify these files.

Register or Unregister a Depot

A software depot can be registered or unregistered with **swreg**. This means you don't have to remove a depot; if you temporarily don't want it used, you can unregister it.

Manage Jobs (Command Line or GUI, this is SD-OV only)



Software Distributor jobs can be viewed and removed with **swjob**. The graphical user interface version of this command can be invoked with **sd** or **swjob -i**.

Listing Software

Although I like the Graphical User Interface of **swinstall,** you can also issue Software Distributor commands at the command line. One example is the **swlist** command. The **swlist** command is useful for viewing the software you have loaded on your system, viewing the software you have loaded in a depot, or producing a list of depots. A Graphical User Interface to the **swlist** command can be invoked with the ***-i*** option and is also available in SAM. With the **swlist** command, you perform many functions, including the following:

- List the software you have at the specified level with the **-l** option. I will show several examples shortly. The levels you can specify are:

    root

    depot

    bundle

    product

    subproduct

    fileset

    file

Levels are delineated by "." so you will see *bundle.[product].[subproduct].[fileset]*. You can get all kinds of useful information out of **swlist** and use this for other purposes. Some of the things you can do with **swlist** are:

- Display the table of contents from a software source.
- Specify which attributes you wish to see for a level of software such as name, size, revision, etc.

- 

Create a list of products that can be used as input to other Software Distributor commands, such as **swinstall** and **swremove**.

When you run **swlist** with no options, you get a list of the software products installed on your system. Let's try a few **swlist** commands with the **-l** option to view software installed on a system (by default, **swlist** will list installed products; you can use the **-s** option to specify a software depot or other source). The following example shows listing software at the *bundle* level.

```
$ swlist -l bundle

# various header information
#             .
B2491BA      B.11.00 MirrorDisk/UX
B3701AA_TRY  B 11.00.31 Trial HP GlancePlus/UX Pak s800
B3929BA      B11.00  HP   OnLineJFS (Advanced VxFS)
B3947BA      B.11.00 HP Process Resource Manager
B5725AA      B.1.4   HP-UX Installation Utilities (Ignite-UX)
HPUXEng32RT  B 11.00 English HP-UX 32-bit Runtime Environment
```

This system has the HP-UX runtime environment, GlancePlus/UX trial software, HP OnLineJFS, and MirrorDisk/UX.

If we run **swlist** at the product level, the following is produced for GlancePlus/UX trial software:

```
$ swlist -l product B3701AA_TRY

# various header information
#             .
#             .
#             .
B3701AA_TRY                 B.11.00.31  Trial HP GlancePlus/UX Pak for s800 11.00
B3701AA_TRY.MeasurementInt  B.11.00.31 HP-UX Measurement Interface for 11.00
B3701AA_TRY.MeasureWare     B.11.00.31  MeasureWare Software/UX
B3701AA_TRY.Glance          B.11.00.31  HP GlancePlus/UX

 (bundle)  (product)
```

GlancePlus/UX is comprised of the two products shown in this example. Are there any subproducts of which GlancePlus/UX is comprised? The following example will help us determine the answer.

```
$ swlist -l subproduct B3701AA_TRY

# various header information
#             .
#             .
#             .
B3701AA_TRY                 B.11.00.31  Trial HP GlancePlus/UX Pak for s800 11.00
B3701AA_TRY.MeasurementInt  B.11.00.31 HP-UX Measurement Interface for 11.00
B3701AA_TRY.MeasureWare     B.11.00.31 MeasureWare Software/UX
B3701AA_TRY.Glance          B.11.00.31 HP GlancePlus/UX

(bundle)   (product)
```

The output of the products and subproducts levels is the same; therefore, there are no subproducts in GlancePlus/UX. We can go one step further and take this to the fileset level, as shown in the following example:

```
$ swlist -l fileset B3701AA_TRY

# various header information
#             .
#             .
#             .
B3701AA_TRY                       B.11.00.31  Trial HP GlancePlus/UX Pak for s800 11.00
B3701AA_TRY.MeasurementInt          B.11.00.31  HP-UX Measurement Interface for 11.00
B3701AA_TRY.MeasurementInt.ARM      B.11.00.31  HP-UX Application Response Measurement
for 11.00
B3701AA_TRY.MeasurementInt.MI       B.11.00.31 HP-UX Measurement Interface for 11.00
```

```
B3701AA_TRY.MeasureWare            B.11.00.31 MeasureWare Software/UX
B3701AA_TRY.MeasureWare.MWA        B.11.00.31 MeasureWare Software files
B3701AA_TRY.MeasureWare.MWANO      B.11.00.31 MeasureWare NOS Connectivity Module
Software files
B3701AA_TRY.MeasureWare.PERFDSI    B.11.00.31 HP PCS Data Source Integration
B3701AA_TRY.Glance                 B.11.00.31 HP GlancePlus/UX
B3701AA_TRY.Glance.GLANC           B.11.00.31 HP GlancePlus files
B3701AA_TRY.Glance.GPM             B.11.00.31     HP GlancePlus Motif interface files

(bundle)   (product) (fileset)
```



With the **swlist** command and the *-l* option, we have worked our way down the hierarchy of HP GlancePlus/UX. Going down to the file level with the *-l file* option produces a long list of files associated with this product.

Table 2-4 shows some of the *-l* options to **swlist** that I use:

Table 2-4. List of Some swlist -l Options

Command	Explanation
swlist -l root	Shows the root level.
swlist -l shroot	Shows the shared roots.
swlist -l prroot	Shows the private roots.
swlist -l bundle	Shows bundles only.
swlist -l product	Shows products only.
swlist -l subproduct	Shows both products and subproducts.
swlist -l fileset	Shows products, subproducts, and filesets.
swlist -l file	Shows products, subproducts, filesets, files and numbers.
swlist -l category	Shows all categories of available patches if they have category in their definition.
swlist -l patch	Shows all applied patches.
swlist -l depot	Shows all depots on the local host.
swlist -l depot @ sys	Shows all depots on *sys.*



I also like to use the *-a* option with **swlist.** *-a* specifies that you would like to see a specific attribute associated with the software you are listing. You can look at the **sd** manual page on your HP-UX system to get a complete list of attributes. One attribute I often look at is *size*. To get a list of the *subproducts* in *NETWORKING* and their *size* in KBytes you would issue the following command:

```
$ swlist -l subproduct -a size NETWORKING
```

Another attribute I often view is *revision,* which you can view with the following command:

```
$ swlist -l subproduct -a revision NETWORKING
```

Sometimes, the brief descriptions of filesets that are given are insufficient to really understand the fileset. The *title* attribute provides a descriptive title, which you can see with the following command for the fileset level:

```
$ swlist -l fileset -a title NETWORKING
```

Table 2-5 is a list of some attributes that you may find of interest.

Table 2-5. List of Some Attributes of Interest

Attribute	Explanation
architecture	Shows the target systems supported by the software.
`category`	Shows the type of software.
`description`	Shows more detailed description of software.
`title`	Shows the official name of the software.
`owner`	Shows the owner of the file.
`path`	Shows the full pathname of the file.
`revision`	Shows the revision number of the software object.
`size`	Shows the size of all filesets.
`state`	Shows the state of the fileset.

The other Software Distributor commands listed earlier can also be issued at the command line. You may want to look at the manual pages for these commands as you prepare to do more advanced Software Distributor work than loading software from DVD, CD-ROM, or tape.



To system administrators familiar with HP-UX 9.x, there is a different organization of software in HP-UX 10.x and 11.x, but the Graphical User Interface of **swinstall** combined with the better organization of Software Distributor makes this an advantage of HP-UX 11.x.

Manual Pages for Commands Used in Chapter 2

There are several manual pages from commands used in this chapter in Appendix A. The following list of commands from this chapter appears in Appendix A:

"**sw**" command summaries



Chapter 3. Logical Volume Manager and Veritas Volume Manager

Introduction

Veritas Volume Manager

Logical Volume Manager Background

Commonly Used LVM Procedures

Adding Disks

Reconfiguring Some Disks - An Example of Using Some Logical Volume Commands

Optional Procedure to Exchange Dump Priorities

HP VERITAS Volume Manager

Some Additional File System-Related Commands



Introduction

Since this is a "how to" book I'll focus on some commonly performed Logical Volume Manager (LVM) procedures in the first part of this chapter. In the Veritas Volume Manager (VxVM) section I'll provide an overview of VxVM focusing on its fine graphical interface. The following is a summary of what I'll cover:

- Veritas Volume Manager overview
- Logical Volume Manager (LVM) - You will probably be using LVM to manage the data on your system. I provide LVM background in this section.
- Example of adding XP 256 disks to your system using LVM.
- Example of Using LVM to Reconfigure Disks - I include an example of a complex disk reconfiguration performed on a real system to show how many LVM commands are used. Although this disk reconfiguration may not be required on your system(s), it is a good example of using LVM commands.
- Some Additional File System Commands.

Veritas Volume Manager

New to HP-UX 11i is the Veritas Volume Manager, which is bundled with the operating system.

Unlike Logical Volume Manager, most of the work you perform with Veritas Volume Manager is performed through the Java-based graphical interface. The examples in this section all use the graphical interface to introduce you to Veritas Volume Manager.

Logical Volume Manager Background

Logical Volume Manager is a disk management subsystem that allows you to manage physical disks as logical volumes. This means that a file system can span multiple physical disks. You can view Logical Volume Manager as a flexible way of defining boundaries of disk space that are independent of one another. Not only can you specify the size of a logical volume, but you can also change its size if the need arises. This possibility is a great advancement over dedicating a disk to a filesystem or having fixed-size partitions on a disk. Logical volumes can hold filesystems, raw data, or swap space. You can now specify a logical volume to be any size you wish, have logical volumes that span multiple physical disks, and then change the size of the logical volume if you need to do so!

So what do you need to know in order to set up Logical Volume Manager and realize all these great benefits? First, you need to know the terminology, and second, you need to know Logical Volume Manager commands. As with many other system administration tasks, you can use graphical management tools covered in Chapter 10. But, as usual, I recommend that you read this overview and at least understand the basics of Logical Volume Manager.

For use with the Journaled Filesystem (JFS), Hewlett-Packard has an add-on product called HP OnLineJFS. This product allows you to perform many of the LVM functions without going into single-user mode. For example, when a filesystem needs to be expanded, the logical volume on which it resides needs to be unmounted before the expansion takes place. Normally, that unmounting would mean shutting the system down into single-user mode so that no user or process could access the volume and it could then be unmounted. With OnLineJFS, the logical volumes and file systems are simply expanded with the system up and running and no interruption to users or processes.

With both JFS and OnlineJFS an *intent log*, or *journal*, is used to keep track of metadata (structural information) that would be written synchronously on a traditional HFS-based system. The journal information is used to complete an operation if a crash occurs, thereby making system recovery with **fsck** much faster.

With both the base JFS product and OnlineJFS product, you would specify a file system as *vxfs* rather than *hfs*. With the **newfs** or **mount** commands, for instance, you would specify *-F vxfs*. There are many options that you could specify with *-o* when working with JFS. One of the most common is *-o largefiles,* which allows files larger than two GBytes in size.

Logical Volume Manager Terms

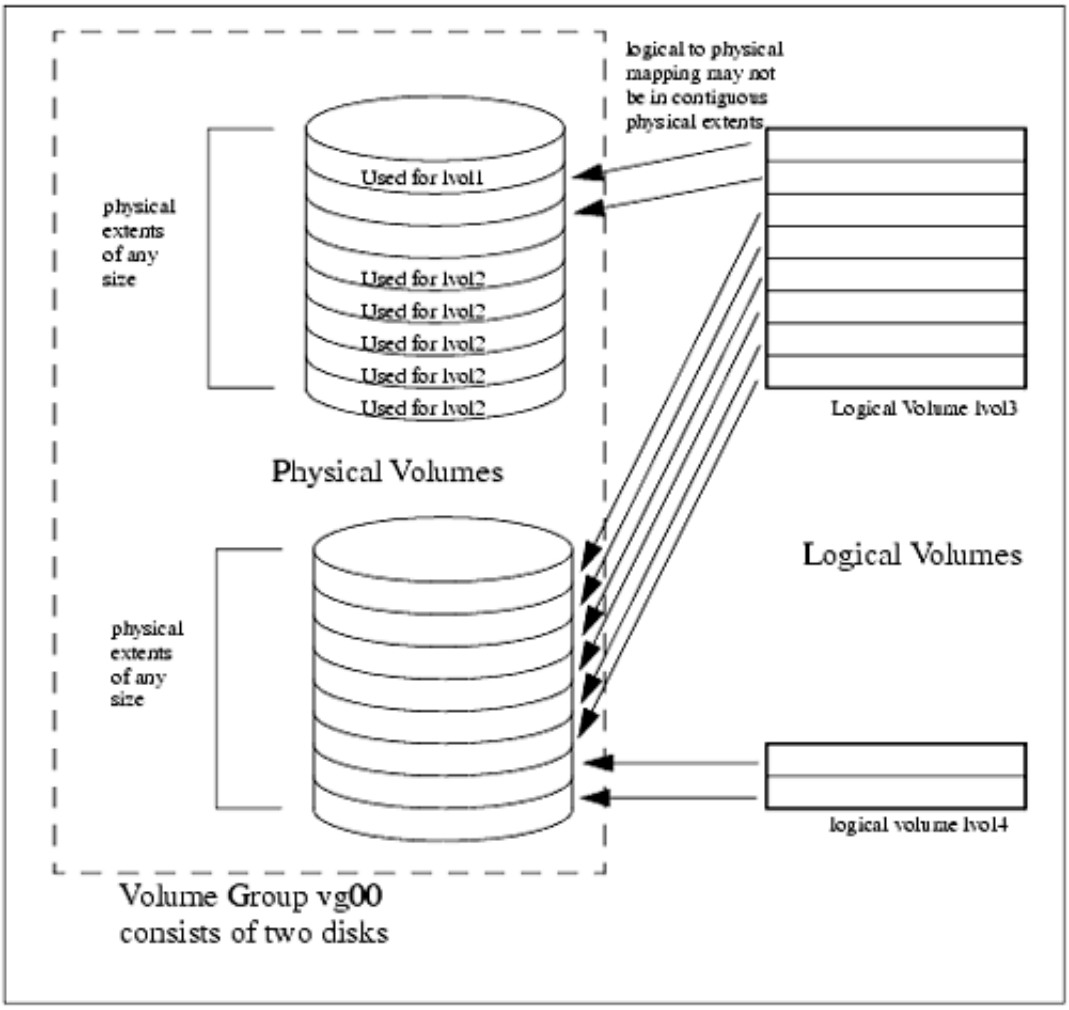
The following terms are used when working with Logical Volume Manager. They are only some of the terminology associated with Logical Volume Manager, but they are enough for you to get started with Logical Volume Manager. It is a good idea to read the following brief overview of these terms.

Volume	A volume is a device used for a filesystem, swap, or raw data. Without Logical Volume Manager, a volume would be either a disk partition or an entire disk drive.
Physical Volume	A physical volume is a disk that has been not been initialized for use by Logical Volume Manager. An entire disk must be initialized if it is to be used by Logical Volume Manager; that is, you can't initialize only part of a disk for Logical Volume Manager use and the rest for fixed partitioning.
Volume Group	A volume group is a collection of logical volumes that are managed by Logical Volume Manager. You would typically define which disks on your system are going to be used by Logical Volume Manager and then define how you wish to group these into volume groups. Each individual disk may be a volume group, or more than one disk may form a volume group. At this point, you have created a pool of disk space called a *volume group*. A disk can belong to only one volume group. A volume group may span multiple physical disks.
Logical Volume	This is space that is defined within a volume group. A volume group is divided up into logical volumes. This is like a disk partition, which is of a fixed size, but you have the flexibility to change its size. A logical volume is contained within a volume group, but the

	volume group may span multiple physical disks. You can have a logical volume that is bigger than a single disk.
Physical Extent	A physical extent is a set of contiguous disk blocks on a physical volume. If you define a disk to be a physical volume, then the contiguous blocks within that disk form a physical extent. Logical Volume Manager uses the physical extent as the unit for allocating disk space to logical volumes. If you use a small physical extent size, such as 1 MByte, then you have a fine granularity for defining logical volumes. If you use a large physical extent size such as 256 MBytes, then you have a coarse granularity for defining logical volumes. The default size is 4 MBytes.
Logical Extents	A logical volume is a set of logical extents. Logical extents and physical extents are the same size within a volume group. Although logical and physical extents are the same size, this doesn't mean that two logical extents will map to two contiguous physical extents. It may be that you have two logical extents that end up being mapped to physical extents on different disks!
/etc/lvmtab	This file has in it the device file associated with each disk in a volume group. **/sbin/lvmrc** starts each volume group by reading the contents of this file at boot time. This file can be rebuilt with **vgscan**. This is not an ascii file, so **strings /etc/lvmtab** is used to see its contents.
PV Links	Physical Volume Links (PV Links) provide dual SCSI or FL links to the same disk. If one of the links were to fail, the other link would automatically take over routing I/O to the disk.

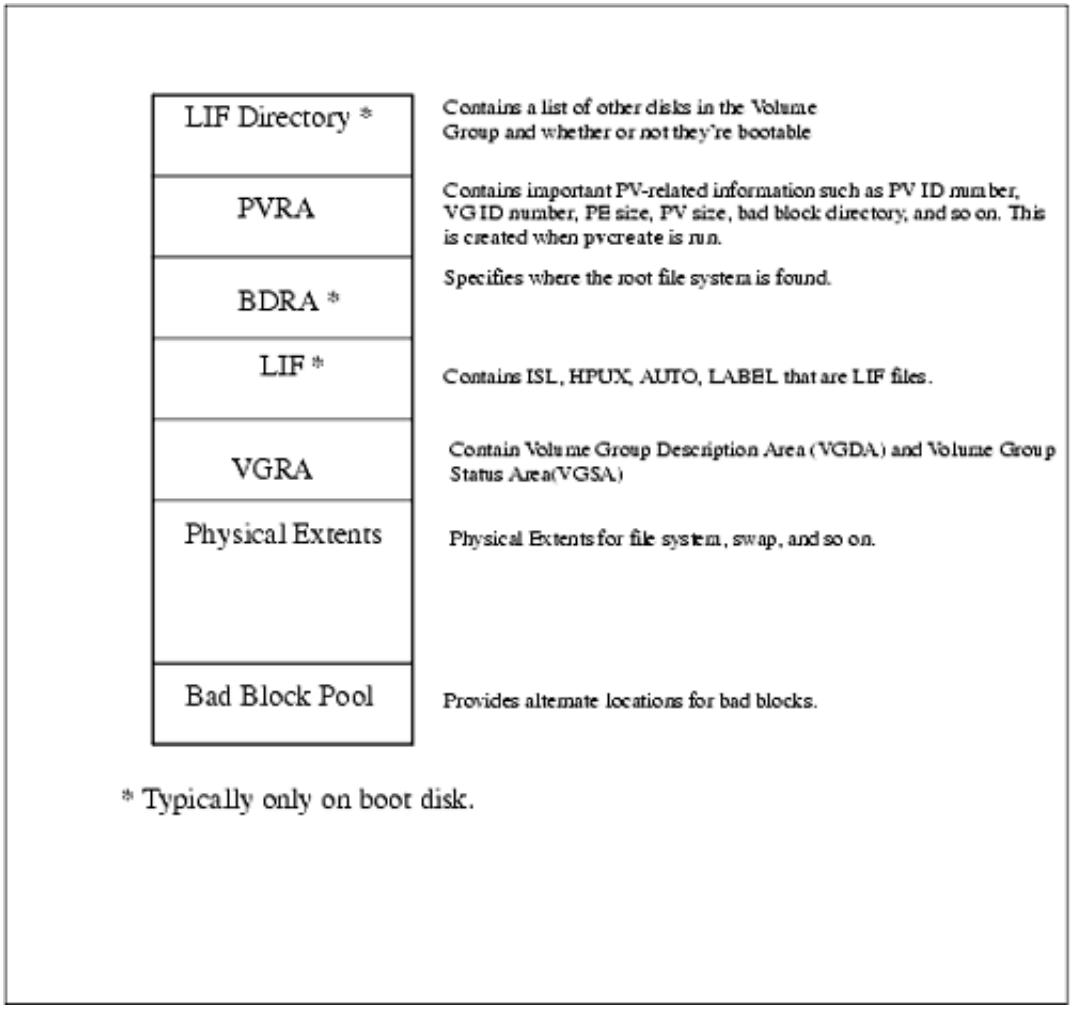
Figure 3-1 graphically depicts some of the logical volume terms we just covered. In this diagram you can see clearly that logical extents are not mapped to contiguous physical extents, because some of the physical extents are not used.

Figure 3-1. Logical Volume Manager Partial Logical to Physical Mapping



Another diagram that is helpful in understanding LVM is Figure 3-2 which shows some of the important components of which an LVM boot disk is comprised.

Figure 3-2. Boot Disk Components



Some of these components exist only on a boot disk, as noted, and others exist on all LVM disks. The components in *LIF* are put there when the **mkboot** and **lvlnboot** commands are run.

Disk Mirroring

Logical volumes can be mirrored one or more times, creating an identical image of the logical volume. This means that a logical extent can map to more than one physical extent if mirrored.

Mirroring is done with the *MirrorDisk/UX* product. Mirroring is done on a logical volume basis, not a disk basis. You must specify the logical volumes you want mirrored. This is demonstrated in several examples later in this chapter. Recall Figure 3-1, which showed logical extents mapped to physical extents. With mirroring you map each logical extent to two or more physical extents, meaning that your data physically resides in two or more places.

You may have an environment where you wish to mirror some or all of the logical volumes. You can configure your mirroring manually (as shown in some examples later in this chapter) or graphical management tools covered in Chapter 10 can be used to set up disk mirroring for you. You must first, however, decide the characteristics of your mirroring. There is a mirroring policy called "strict." You define one of the following three strict policies when you create the logical volume using the following options:

n No, this is not a strict allocation policy, meaning that mirrored copies of a logical extent can share the same physical volume. This means that your original data and mirrored data may indeed be on the same physical disk. If you encounter a disk mechanism problem of some type, you may lose both your original and mirrored data.

y Yes, this is a strict allocation policy, meaning that mirrored copies of a logical extent may not share the

same physical volume. This is safer than allowing mirrored copies of data to share the same physical volume. If you have a problem with a disk in this scenario, you are guaranteed that your original data is on a different physical disk from your mirrored data. Original data and mirrored data are always part of the same volume group even if you want them on different physical volumes.

g Mirrored data will not be on the same Physical Volume Group (PVG) as the original data. This policy is called a PVG-strict allocation policy.

The strict allocation policy depends on your environment. Most installations that employ mirroring buy sufficient disk drives to mirror all data. In an environment such as this, I would create two volume groups, one for the original data and one for the mirrored data, and use the "strict -g" option when creating logical volumes so that the original data is on one volume group and the mirrored data on the other.

Logical Volume Manager Commands

The following are definitions of some of the more common Logical Volume Commands. I am giving a description of these commands here so that when you see them, you'll have an idea of each command's use. Although these are not all of the Logical Volume Manager commands, these are the ones I use most often and are the commands you should have knowledge of when using Logical Volume Manager. The commands are grouped by physical volume (pv) commands, volume group (vg) commands, and logical volume (lv) commands. There are *usage* summaries of some of the commands included with the descriptions, and details can be found in the online manual pages. Some of the commands, such as **vgdisplay, pvdisplay**, and **lvdisplay** were issued so that you could see examples of these. The following output of **bdf** will be helpful to you when you view the output of Logical Volume Manager commands that are issued. The output of **bdf** shows several logical volumes mounted (**lvol1, lvol3, lvol4, lvol5, lvol6, lvol7, lvol8**), all of which are in volume group **vg00** (see the **bdf** command overview later in this chapter).

$ **bdf**

Filesystem	kbytes	used	avail	%used	Mounted on
/dev/vg00/lvol3	47829	18428	24618	43%	/
/dev/vg00/lvol1	67733	24736	36223	41	/stand
/dev/vg00/lvol8	34541	8673	22413	28%	/var
/dev/vg00/lvol7	299157	149449	119792	56%	/usr
/dev/vg00/lvol4	23013	48	20663	0%	/tmp
/dev/vg00/lvol6	99669	32514	57188	36%	/opt
/dev/vg00/lvol5	19861	9	17865	0%	/home
/dev/dsk/c0t6d0	802212	552120	169870	76%	/mnt/9.x

Physical Volume Commands

pvchange This command is used to change a physical volume in some way. For example, you may wish to allow additional physical extents to be added to a physical volume if they are not permitted, or prohibit additional physical extents from being added to a physical volume if, indeed, they are allowed.

```
/usr/sbin/pvchange [-A autobackup] -s pv_path

/usr/sbin/pvchange [-A autobackup] -S autoswitch pv_path

/usr/sbin/pvchange [-A autobackup] -x extensibility pv_path

/usr/sbin/pvchange [-A autobackup] -t IO_timeout pv_path

/usr/sbin/pvchange [-A autobackup] -z sparepv pv_path
```

pvcreate This command is used to create a physical volume that will be part of a volume group. Remember that a volume group may consist of several physical volumes. The physical volumes are the disks on your system. The example below shows creating a physical volume.

```
/usr/sbin/pvcreate [-b] [-B] [-d soft_defects] [-s disk_size]
                   [-f] [-t disk_type] pv_path
```

pvdisplay This command shows information about the physical volumes you specify. You can get a lot of information about the logical to physical mapping with this command if you use the verbose (*-v*) option. With *-v* **pvdisplay** will show you the mapping of logical to physical extents for the physical volumes specified.

```
/usr/sbin/pvdisplay [-v] [-b BlockList] pv_path ...
```

You get a lot of other useful data from this command, such as the name of the physical volume; the name of the volume group to which the physical volume belongs; the status of the physical volume; the size of physical extents on the physical volume; the total number of physical extents; and the number of free physical extents.

pvmove You can move physical extents from one physical volume to other physical volumes with this command. By specifying the source physical volume and one or more destination physical volumes, you can spread data around to the various physical volumes you wish with this command. In the example below, if the lvol had been left off the entire disk would have been moved.

```
/usr/sbin/pvmove [-A autobackup] [-n lv_path] source_pv_path
                 [dest_pv_path ... | dest_pvg_name ...]
```

```
# pvmove -n /dev/vg01/lvol1 /dev/dsk/c0t2d0 /dev/dsk/c0t4d0
```

mknod Although this is not an LVM command exclusively, it is used often when creating volume groups, as shown in the example below:

```
# cd /dev
# mkdir vg01
# cd vg01
# mknod group c 64 0x010000
```

Volume Group Commands

vgcfgbackup This command is used to save the configuration information for a volume group. Remember that a volume group is made up of one or more physical volumes.

```
/usr/sbin/vgcfgbackup [-f vg_conf_path] [-u] vg_name
```

vgcfgrestore This command is used to restore the configuration information for a volume group.

```
/usr/sbin/vgcfgrestore -n vg_name -l
```

```
/usr/sbin/vgcfgrestore [-R] [-F] -n vg_name [-o old_pv_path]
                       pv_path
```

```
/usr/sbin/vgcfgrestore -f vg_conf_path -l
```

```
/usr/sbin/vgcfgrestore [-R] [-F] -f vg_conf_path
                       [-o old_pv_path] pv_path
```

vgchange This command makes a volume group active or inactive. With the *-a* option, you can deactivate (*-a n*) a volume group or activate (*-a y*) a volume group.

```
/usr/sbin/vgchange -a availability [-l] [-p] [-q quorum] [-s]
                   [-P resync_daemon_count] [vg_name...]
```

vgcreate You can create a volume group and specify all of its parameters with this command. You specify a volume group name and all of the associated parameters for the volume group when creating it. You can specify many physical volume block devices on the same command line if you wish. *ex2* below shows creating a PV link. The second path in the **vgcreate** command is the *alternate* link.

```
/usr/sbin/vgcreate [-f] [-A autobackup] [-x extensibility]
                   [-e max_pe] [-l max_lv] [-p max_pv]
                   [-s pe_size] [-g pvg_name] vg_name
                   pv_path ...
```

ex1: **vgcreate /dev/vg01 /dev/dsk/c0t2d0**

ex2: **pvcreate /dev/rdsk/c0t1d0**
vgcreate /dev/vg01 /dev/dsk/c0t1d0 /dev/dsk/c2t1d0

vgdisplay This displays all information related to the volume group if you use the verbose (*-v*) option, including the volume group name; the status of the volume group; the maximum, current, and open logical volumes in the volume group; the maximum, current, and active physical volumes in the volume group; and physical extent-related information.

```
/usr/sbin/vgdisplay [-v] [vg_name ...]
```

vgexport This command removes a logical volume group from the system, but does not modify the logical volume information on the physical volumes. These physical volumes can then be imported to another system using **vgimport.** Use *-s* for sharable option in ServiceGuard environments.

```
/usr/sbin/vgexport [-m mapfile] [-p] [-v] [-f outfile] vg_name

 /usr/sbin/vgexport -m mapfile -s -p -v vg_name
```

```
# vgchange -a n /dev/vg01
# vgexport -v -m /etc/lvmconf/vg01.map /dev/vg01
```

vgextend Physical volumes can be added to a volume group with this command by specifying the physical volume to be added to the volume group. After the summary of the command is an example:

```
/usr/sbin/vgextend [-f] [-A autobackup] [-g pvg_name]
     [-x extensibility] [-z sparepv] vg_name pv_path ...
```

```
# vgextend /dev/rdsk/c0t2d0
# vgextend /dev/vg01 /dev/dsk/c0t2d0
```

vgimport This command can be used to import a physical volume to another system. Note in the example below that several disks could have been specified on the **vgimport** line.

```
/usr/sbin/vgimport [-m mapfile] [-p] [-v] [-f infile]
                   vg_name pv_path ...

/usr/sbin/vgimport -m mapfile -s -v vg_name
```

```
# mkdir /dev/vg01
# mknod /dev/vg01/group c 64 0x010000
# vgimport -v -m /dev/lvmconf/vg01.map /dev/vg01 /dev/dsk/c0t2d0
# vgchange -a y /dev/vg01
# vgcfgbackup vg01
```

vgreduce The size of a volume group can be reduced with this command, by specifying which physical volume(s) to remove from a volume group. Make sure that the physical volume to be removed has no data on it before doing this.

```
/usr/sbin/vgreduce [-A autobackup] vg_name pv_path ...

/usr/sbin/vgreduce [-A autobackup] [-l] vg_name pv_path

/usr/sbin/vgreduce [-A autobackup] [-f] vg_name
```

vgremove A volume group definition can be completely removed from the system with this command. The entry for the volume group in **/dev** is not removed. An example of removing a volume

group is shown below also:

```
/usr/sbin/vgremove vg_name ...
```

```
# vgchange -a n /dev/vg01
# lvremove /dev/vg01/lvol1          ;run for all lvols in vg
# vgremove /dev/vg01
```

vgscan In the event of a catastrophe of some type, you can use this command to scan your system in an effort to rebuild the **/etc/lvmtab** file. The *-p* option performs a *preview* of **vgscan**.

```
/usr/sbin/vgscan [-a] [-p] [-v]
```

vgsync There are times when mirrored data in a volume group becomes "stale" or out-of-date. **vgsync** is used to synchronize the physical extents in each mirrored logical volume in a volume group.

```
/usr/sbin/vgsync vg_name ...
```

Logical Volume Commands

lvcreate This command is used to create a new logical volume. A logical volume is created within a volume group. A logical volume may span multiple disks, but must exist within a volume group. Many options exist for this command, and two that you would often use are *-L* to define the size of the logical volume and *-n* to define the name of the logical volume.

```
/usr/sbin/lvcreate [-A autobackup] [-c mirror_consistency]
     [-C contiguous] [-d schedule] [-D distributed]
     [-i stripes -I stripe_size] [-l le_number | -L lv_size]
     [-m mirror_copies] [-M mirror_write_cache] [-n lv_name]
     [-p permission] [-r relocate] [-s strict] vg_name
```

An interesting nuance to **lvcreate** is that you can't specify a physical volume on which to create the logical volume. If you want a logical volume and its mirror on specific disks you first run **lvcreate** specifying no size, then two **lvextend** commands for the original and mirror copies as shown below:

```
# lvcreate /dev/vg03/lvol3
# lvextend -L 400 /dev/vg03/lvol3 /dev/dsk/c0t2d0
# lvextend -m 1 /dev/vg03/lvol3 /dev/dsk/c2t2d0
```

lvchange This command is used to change the logical volume in some way. For example, you may wish to change the permission on a logical volume to read-write (w) or read (r) with the *-p* option. The *-C* is used to specify contiguous. Or, you may want to change the strict policy (described under Disk Mirroring) to strict (y), not strict (n), or PVG strict (g).

```
/usr/sbin/lvchange [-a availability] [-A autobackup]
     [-c mirror_consistency] [-C contiguous]
     [-d schedule] [-D distributed] [-M mirror_write_cache]
     [-p permission] [-r relocate] [-s strict]
     [-t IO_timeout] lv_path
```

lvdisplay This command shows the status and characteristics of every logical volume that you specify. If you use the verbose (*-v*) option of this command, you get a lot of useful data in many categories, including:

1. Information about the way in which the logical volumes are set up, such as the physical volume on which the logical extents appear; the number of local extents on a physical volume; and the number of physical extents on the physical volume.
2. Detailed information for logical extents, including the logical extent number and some information about the physical volume and physical extent for the logical extent.

```
/usr/sbin/lvdisplay [-k] [-v] lv_path  ...
```

lvextend This command is used to increase the number of physical extents allocated to a logical

volume. We sometimes underestimate the size required for a logical volume, and with this command you can easily correct this problem. You may want to extend a logical volume to increase the number of mirrored copies (using the *-m* option), to increase the size of the logical volume (using the *-L* option), or to increase the number of logical extents (using the *-l* option). You can specify the disk on which you want the logical volume extended and the *size* is the new size, not an amount by which you want the logical volume extended:

```
/usr/sbin/lvextend [-A autobackup] {-l le_number | -L lv_size
     | -m mirror_copies} lv_path [pv_path ... | pvg_name ...]
```

```
# lvextend -L 300 /dev/vg01/lvol1
```

extendfs Use this command after **lvextend**. Whereas the **lvextend** command expands the logical volume, **extendfs** expands the filesystem within the logical volume. You would use **fsadm** if you have OnlineJFS installed rather than the procedure that appears below:

```
/usr/sbin/extendfs [-F FStype] [-q] [-v] [-s size] special
```

```
# umount /dev/vg01/lvol1              ;use fsadm if online
# extendfs -f vxfs /dev/vg01/rlvol1    ;JFS installed
# mount /dev/vg01/lvol1 /backup
```

mkboot Places boot utilities in boot area.

```
/usr/sbin/mkboot  [-b boot_file_path]   [-c [-u] | -f | -h | -u]
   [-i included_lif_file]  [-p preserved_lif_file]
   [-l | -H | -W] [-v] device

/usr/sbin/mkboot [-a auto_file_string] [-v] device

/usr/sbin/rmboot device
```

lvlnboot Use this to set up a logical volume to be a root, boot, primary swap, or dump volume (this can be undone with **lvrmboot**). Issuing the **lvlnboot** command with the *-v* option gives the current settings.

```
/usr/sbin/lvlnboot [[-A autobackup { -b boot_lv |
  -d dump_lv | -r root_lv | -R | -s swap_lv }] [-v] [vg_name]
```

lvsplit and **lvmerge**

These commands are used to split and merge mirrored logical volumes, respectively. If you have a mirrored logical volume, **lvsplit** will split this into two logical volumes. **lvmerge** merges two logical volumes of the same size, increasing the number of mirrored copies. In the example below, two logical volumes are split and the name *backup* is appended to each. **fsck** is run on the split volumes and they are mounted under separate names. **lvsplit** performs an *atomic* split meaning that the split takes place at the same time for all logical volumes that appear on the command line. In the **lvmerge** example the primary, or master copy, is specified first and the logical volume to be synchronized with the primary is specified second:

```
/usr/sbin/lvsplit [-A autobackup] [-s suffix]
                  [-g PhysicalVolumeGroup] lv_path ...
```

```
# lvsplit -s backup /dev/vg01/lvol1 /dev/vg01/lvol4
# fsck -f vxfs -p /dev/vg01/rlvol1backup
# fsck -f vxfs -p /dev/vg01/rlvol4backup
# mount /dev/vg01/lvol1 backuplvol1
# mount /dev/vg01/rlvol4 backuplvol4
```

```
/usr/sbin/lvmerge [-A autobackup] dest_lv_path src_lv_path
```

```
# lvmerge /dev/vg00/lvol1backup /dev/vg00/lvol1
```

lvmmigrate This command prepares a root file system in a disk partition for migration to a logical volume. You would use this if you had a partition to convert into a logical volume.

lvreduce Use this to decrease the number of physical extents allocated to a logical volume. When creating logical volumes, we sometimes overestimate the size of the logical volume. This command can be used to set the number of mirrored copies (with the *-m* option), decrease the number of logical extents (with the *-l* option), or decrease the size of the logical volume (with the *-L* option). Be careful when decreasing the size of a logical volume. You may make it smaller than the data in it. If you choose to do this, make sure that you have a good backup of your data.

```
/usr/sbin/lvreduce [-A autobackup] [-f] -l le_number lv_path

/usr/sbin/lvreduce [-A autobackup] [-f] -L lv_size lv_path

/usr/sbin/lvreduce [-A autobackup] -m mirror_copies lv_path
[pv_path ...]

/usr/sbin/lvreduce [-A autobackup] -k -m mirror_copies lv_path
                   pv_key ...
```

lvremove After emptying a logical volume, you can use this command to remove logical volumes from a volume group.

```
/usr/sbin/lvremove [-A autobackup] [-f] lv_path ...
```

lvrmboot Use this if you don't want a logical volume to be root, boot, primary swap, or a dump device (this is the converse of the **lvlnboot** command). However, unless you have a disk partition to boot from, don't leave the system without a root or boot device designated with the **lvlnboot** command, or the system won't know where to boot from.

```
/usr/sbin/lvrmboot [-A autobackup] [-d dump_lv] [-r] [-s]
                   [-v] vg_name
```

lvsync There are times when mirrored data in a logical volume becomes "stale" or out-of-date. **lvsync** is used to synchronize the physical extents in a logical volume.

```
/usr/sbin/lvsync lv_path ...
```

newfs Although this is not exclusively an LVM command, it is often used to place a file system on a newly created logical volume. The following example shows placing a file system on new logical volume and mounting it:

```
/usr/sbin/newfs [-F FStype] [-o specific_options] [-V] special
```

```
# newfs -F vxfs -o largefiles /dev/vg01/lvol1
# mount /dev/vg01/lvol1 /backup
```

JFS and OnlineJFS Commands

fsck Because of the intent log you can run **fsck** much faster with JFS than without it. If you specify *-F vxfs* below your **fsck** will run faster and there are many additional options with *-o* that you can specify.

```
/usr/sbin/fsck [-F FSType] [-m] [-V] [special ...]

/usr/sbin/fsck [-F FSType] [-o FSspecific-options] [-V]
               [special ...]
```

fsadm **fsadm** performs a variety of operations when using OnlineJFS, such as defragmentation, resizing, online backup, and so on.

```
/usr/sbin/fsadm [-F FStype] [-V] [-o specific_options] special
```

```
for defragmentation:
fsadm -F vxfs [-d][-D][-d][-E] /mount_point

to increase OnlineJFS file system:
fsadm -F vxfs -b new_size /mount_point
```

```
# lvextend -L 400 /dev/vg01/lvol3
# fsadm -F vxfs 400 /backup
```

to reduce OnlineJFS file system:
fsadm -f vxfs -b *new_size /mount_point*

```
# fsadm -F vxfs -d -D -e -E /backup
# fsadm -f vxvs -b 300 /backup
```

[Team LiB]

PREVIOUS NEXT



Commonly Used LVM Procedures

The best way to learn how to use LVM commands is to evaluate some commonly used procedures. System administrators tend to be careful before running any intrusive (there is a chance something LVM-related will be changed) LVM commands. Always back up your system and create a bootable recovery tape with Ignite-UX before you run any intrusive LVM commands. The following are some procedures for tasks I have encountered on a regular basis. Please modify them to suit your needs. Don't run the commands as shown. You will need to prepare your system, substitute the names of your volumes, and perform additional steps. These commands, however, serve as good examples for ways in which the tasks shown can be performed.

The first task we'll perform is to mirror the root disk to a second disk. This is one of the most commonly run procedures. Most HP 9000s are ordered with a minimum of two internal disks for root and swap. After loading the first disk with HP-UX, selecting logical volume sizes, file system types, and so on, we're ready to mirror the disk (please see Chapter 2 on installing HP-UX 11i to see this procedure.)

Figure 3-3 shows some typical steps to mirror root and swap:

Figure 3-3 Mirror root and swap Disk

```
# pvcreate -B /dev/rdsk/cxtxdx                ; create physical volume for disk to be used as mirror

# vgextend /dev/vg00 /dev/dsk/cxtxdx          ; extend vg00 to include disk to be used as mirror

# mkboot /dev/rdsk/cxtxdx                     ; mkboot places boot utilities in boot area

# mkboot -a "hpux -lq" /dev/rdsk/cxtxdx       ; add the AUTO file and specify low quorum

# mkboot -a "hpux -lq" /dev/rdsk/primary      ; update the AUTO file on primary disk low quorum

# lvextend -m 1 /dev/vg00/lvol1 /dev/dsk/cxdxtx ; mirror lvol1 which is /stand
# lvextend -m 1 /dev/vg00/lvol2 /dev/dsk/cxtxdx ; mirror lvol2 which is root
# lvextend -m 1 /dev/vg00/lvol3 /dev/dsk/cxtxdx ; mirror lvol3 which is swap
# lvextend all others on primary


# setboot -a 8/8.6.0              ; have alternate boot path point to H/W path of new mirror disk
```

This is the one of the most common procedures performed in LVM because you like to have the root and swap disk mirrored whenever possible.

Since this is such an important procedure to know, let's go through an example of running this procedure.

Let's first see what volume groups we have on the system by running **strings /etc/lvmtab** and then run **ioscan** to see what disks exist on the system:

```
# strings /etc/lvmtab
/dev/vg00
/dev/dsk/c1t15d0
# ioscan -funC disk
Class     I  H/W Path        Driver   S/W State   H/W Type     Description
==========================================================================
disk      0  0/0/1/0.1.0     sdisk    CLAIMED     DEVICE       HP      DVD-ROM 305
                            /dev/dsk/c0t1d0   /dev/rdsk/c0t1d0
disk      1  0/0/1/1.15.0    sdisk    CLAIMED     DEVICE       SEAGATE ST318404LC
                            /dev/dsk/c1t15d0   /dev/rdsk/c1t15d0
disk      2  0/0/2/1.15.0    sdisk    CLAIMED     DEVICE       SEAGATE ST318404LC
                            /dev/dsk/c3t15d0   /dev/rdsk/c3t15d0
disk      3  0/4/0/0/4/0.8.0   sdisk    CLAIMED     DEVICE       HP 18.2GST318406LC
                            /dev/dsk/c4t8d0   /dev/rdsk/c4t8d0
disk      4  0/4/0/0/4/0.10.0  sdisk    CLAIMED     DEVICE       HP 18.2GST318406LC
#                           /dev/dsk/c4t10d0   /dev/rdsk/c4t10d0
```

The disk at *3* is on a different bus than the internal disk and would therefore be an ideal disk on which to mirror the root disk. We'll take the disk at *3*, get some information on it with **diskinfo** and run a **vgscan** to ensure that there is now only one disk in the volume group.

```
# diskinfo /dev/rdsk/c4t8d0
SCSI describe of /dev/rdsk/c4t8d0:
             vendor: HP 18.2G
         product id: ST318406LC
               type: direct access
               size: 17783240 Kbytes
   bytes per sector: 512
# vgdisplay -v /dev/vg00
--- Volume groups ---
VG Name                     /dev/vg00
VG Write Access             read/write
VG Status                   available
Max LV                      255
```

```
Cur LV                     8
Open LV                    8
Max PV                     16
Cur PV                     1
Act PV                     1
Max PE per PV              4350
VGDA                       2
PE Size (Mbytes)           4
Total PE                   4340
Alloc PE                   1965
Free PE                    2375
Total PVG                  0
Total Spare PVs            0
Total Spare PVs in use     0

   --- Logical volumes ---
   LV Name                     /dev/vg00/lvol1
   LV Status                   available/syncd
   LV Size (Mbytes)            300
   Current LE                  75
   Allocated PE                75
   Used PV                     1

   LV Name                     /dev/vg00/lvol2
   LV Status                   available/syncd
   LV Size (Mbytes)            2048
   Current LE                  512
   Allocated PE                512
   Used PV                     1

   LV Name                     /dev/vg00/lvol3
   LV Status                   available/syncd
   LV Size (Mbytes)            200
   Current LE                  50
   Allocated PE                50
   Used PV                     1

   LV Name                     /dev/vg00/lvol4
   LV Status                   available/syncd
   LV Size (Mbytes)            500
   Current LE                  125
   Allocated PE                125
   Used PV                     1

   LV Name                     /dev/vg00/lvol5
   LV Status                   available/syncd
   LV Size (Mbytes)            200
   Current LE                  50
   Allocated PE                50
   Used PV                     1

   LV Name                     /dev/vg00/lvol6
   LV Status                   available/syncd
   LV Size (Mbytes)            1032
   Current LE                  258
   Allocated PE                258
   Used PV                     1

   LV Name                     /dev/vg00/lvol7
   LV Status                   available/syncd
   LV Size (Mbytes)            1032
   Current LE                  258
   Allocated PE                258
   Used PV                     1

   LV Name                     /dev/vg00/lvol8
   LV Status                   available/syncd
   LV Size (Mbytes)            2548
   Current LE                  637
   Allocated PE                637
   Used PV                     1

   --- Physical volumes ---
   PV Name                     /dev/dsk/c1t15d0
   PV Status                   available
   Total PE                    4340
   Free PE                     2375
   Autoswitch                  On
```

Now that we're sure of the way in which **vg00** is set up and we have all of the information on the disk, we wish to act as a mirror for the root disk. Let's run all of the commands necessary to create the mirror:

```
# pvcreate -B /dev/rdsk/c4t8d0
Physical volume "/dev/rdsk/c4t8d0" has been successfully created.
# vgextend /dev/vg00 /dev/dsk/c4t8d0
Volume group "/dev/vg00" has been successfully extended.
Volume Group configuration for /dev/vg00 has been saved in /etc/lvmconf/vg00.conf
```

```
# mkboot /dev/rdsk/c4t8d0
# mkboot -a "hpux -lq" /dev/rdsk/c4t8d0
# mkboot -a "hpux -lq" /dev/rdsk/c1t15d0
# lvextend -m 1 /dev/vg00/lvol1 /dev/dsk/c4t8d0
The newly allocated mirrors are now being synchronized. This operation will
take some time. Please wait ....
Logical volume "/dev/vg00/lvol1" has been successfully extended.
Volume Group configuration for /dev/vg00 has been saved in /etc/lvmconf/vg00.conf
#
# lvextend -m 1 /dev/vg00/lvol2 /dev/dsk/c4t8d0
The newly allocated mirrors are now being synchronized. This operation will
take some time. Please wait ....
Logical volume "/dev/vg00/lvol2" has been successfully extended.
Volume Group configuration for /dev/vg00 has been saved in /etc/lvmconf/vg00.conf
# lvextend -m 1 /dev/vg00/lvol3 /dev/dsk/c4t8d0
The newly allocated mirrors are now being synchronized. This operation will
take some time. Please wait ....
Logical volume "/dev/vg00/lvol3" has been successfully extended.
Volume Group configuration for /dev/vg00 has been saved in /etc/lvmconf/vg00.conf
# lvextend -m 1 /dev/vg00/lvol4 /dev/dsk/c4t8d0
The newly allocated mirrors are now being synchronized. This operation will
take some time. Please wait ....
Logical volume "/dev/vg00/lvol4" has been successfully extended.
Volume Group configuration for /dev/vg00 has been saved in /etc/lvmconf/vg00.conf
# lvextend -m 1 /dev/vg00/lvol5 /dev/dsk/c4t8d0
The newly allocated mirrors are now being synchronized. This operation will
take some time. Please wait ....
Logical volume "/dev/vg00/lvol5" has been successfully extended.
Volume Group configuration for /dev/vg00 has been saved in /etc/lvmconf/vg00.conf
# lvextend -m 1 /dev/vg00/lvol6 /dev/dsk/c4t8d0
The newly allocated mirrors are now being synchronized. This operation will
take some time. Please wait ....
Logical volume "/dev/vg00/lvol6" has been successfully extended.
Volume Group configuration for /dev/vg00 has been saved in /etc/lvmconf/vg00.conf
# lvextend -m 1 /dev/vg00/lvol7 /dev/dsk/c4t8d0
The newly allocated mirrors are now being synchronized. This operation will
take some time. Please wait ....
Logical volume "/dev/vg00/lvol7" has been successfully extended.
Volume Group configuration for /dev/vg00 has been saved in /etc/lvmconf/vg00.conf
# lvextend -m 1 /dev/vg00/lvol8 /dev/dsk/c4t8d0
The newly allocated mirrors are now being synchronized. This operation will
take some time. Please wait ....
Logical volume "/dev/vg00/lvol8" has been successfully extended.
Volume Group configuration for /dev/vg00 has been saved in /etc/lvmconf/vg00.conf
# setboot -a 0/4/0/0/4/0.8.0
```

The setup of the mirrored volume should now be complete. Let's run **strings /etc/lvmtab** and **vgdisplay** again to ensure that our disk has been added to the volume group:

```
# strings /etc/lvmtab
/dev/vg00
/dev/dsk/c1t15d0
/dev/dsk/c4t8d0
# vgdisplay -v /dev/vg00
--- Volume groups ---
VG Name                     /dev/vg00
VG Write Access             read/write
VG Status                   available
Max LV                      255
Cur LV                      8
Open LV                     8
Max PV                      16
Cur PV                      2
Act PV                      2
Max PE per PV               4350
VGDA                        4
PE Size (Mbytes)            4
Total PE                    8680
Alloc PE                    3930
Free PE                     4750
Total PVG                   0
Total Spare PVs             0
Total Spare PVs in use      0

   --- Logical volumes ---
   LV Name                      /dev/vg00/lvol1
   LV Status                    available/syncd
   LV Size (Mbytes)             300
   Current LE                   75
   Allocated PE                 150
   Used PV                      2

   LV Name                      /dev/vg00/lvol2
   LV Status                    available/syncd
   LV Size (Mbytes)             2048
   Current LE                   512
   Allocated PE                 1024
```

```
   Used PV                     2

   LV Name                     /dev/vg00/lvol3
   LV Status                   available/syncd
   LV Size (Mbytes)            200
   Current LE                  50
   Allocated PE                100
   Used PV                     2

   LV Name                     /dev/vg00/lvol4
   LV Status                   available/syncd
   LV Size (Mbytes)            500
   Current LE                  125
   Allocated PE                250
   Used PV                     2

   LV Name                     /dev/vg00/lvol5
   LV Status                   available/syncd
   LV Size (Mbytes)            200
   Current LE                  50
   Allocated PE                100
   Used PV                     2

   LV Name                     /dev/vg00/lvol6
   LV Status                   available/syncd
   LV Size (Mbytes)            1032
   Current LE                  258
   Allocated PE                516
   Used PV                     2

   LV Name                     /dev/vg00/lvol7
   LV Status                   available/syncd
   LV Size (Mbytes)            1032
   Current LE                  258
   Allocated PE                516
   Used PV                     2

   LV Name                     /dev/vg00/lvol8
   LV Status                   available/syncd
   LV Size (Mbytes)            2548
   Current LE                  637
   Allocated PE                1274
   Used PV                     2

   --- Physical volumes ---
   PV Name                     /dev/dsk/c1t15d0
   PV Status                   available
   Total PE                    4340
   Free PE                     2375
   Autoswitch                  On

   PV Name                     /dev/dsk/c4t8d0
   PV Status                   available
   Total PE                    4340
   Free PE                     2375
   Autoswitch                  On
```

These outputs look great in that both disks are in **vg00**. The end of the **vgdisplay** output (*Physical Volumes)* shows both disks with identical *Total PE* and *Free PE*.

Let's now run **lvdisplay** to see that the physical extents are mirrored across the two disks, **lifls** on both disks to see the boot information, and **setboot** to ensure that we have the *primary* and *alternate* paths set up properly:

```
# lvdisplay -v /dev/vg00/lvol1
--- Logical volumes ---
LV Name                     /dev/vg00/lvol1
VG Name                     /dev/vg00
LV Permission               read/write
LV Status                   available/syncd
Mirror copies               1
Consistency Recovery        MWC
Schedule                    parallel
LV Size (Mbytes)            300
Current LE                  75
Allocated PE                150
Stripes                     0
Stripe Size (Kbytes)        0
Bad block                   off
Allocation                  strict/contiguous
IO Timeout (Seconds)        default

   --- Distribution of logical volume ---
   PV Name            LE on PV  PE on PV
   /dev/dsk/c1t15d0   75        75
   /dev/dsk/c4t8d0    75        75

   --- Logical extents ---
```

```
   LE    PV1                 PE1   Status 1 PV2                 PE2   Status 2
   00000 /dev/dsk/c1t15d0    00000 current  /dev/dsk/c4t8d0     00000 current
   00001 /dev/dsk/c1t15d0    00001 current  /dev/dsk/c4t8d0     00001 current
   00002 /dev/dsk/c1t15d0    00002 current  /dev/dsk/c4t8d0     00002 current
   00003 /dev/dsk/c1t15d0    00003 current  /dev/dsk/c4t8d0     00003 current
   00004 /dev/dsk/c1t15d0    00004 current  /dev/dsk/c4t8d0     00004 current
   00005 /dev/dsk/c1t15d0    00005 current  /dev/dsk/c4t8d0     00005 current
   00006 /dev/dsk/c1t15d0    00006 current  /dev/dsk/c4t8d0     00006 current
   00007 /dev/dsk/c1t15d0    00007 current  /dev/dsk/c4t8d0     00007 current
   00008 /dev/dsk/c1t15d0    00008 current  /dev/dsk/c4t8d0     00008 current
   00009 /dev/dsk/c1t15d0    00009 current  /dev/dsk/c4t8d0     00009 current
   00010 /dev/dsk/c1t15d0    00010 current  /dev/dsk/c4t8d0     00010 current
   00011 /dev/dsk/c1t15d0    00011 current  /dev/dsk/c4t8d0     00011 current
                  -
                  -
                  -
   00071 /dev/dsk/c1t15d0    00071 current  /dev/dsk/c4t8d0     00071 current
   00072 /dev/dsk/c1t15d0    00072 current  /dev/dsk/c4t8d0     00072 current
   00073 /dev/dsk/c1t15d0    00073 current  /dev/dsk/c4t8d0     00073 current
   00074 /dev/dsk/c1t15d0    00074 current  /dev/dsk/c4t8d0     00074 current

# lifls /dev/rdsk/c1t15d0
ODE          MAPFILE      SYSLIB       CONFIGDATA   SLMOD2
SLDEV2       SLDRV2       SLSCSI2      MAPPER2      IOTEST2
PERFVER2     PVCU         SSINFO       ISL          AUTO
HPUX         LABEL
# lifls /dev/rdsk/c4t8d0
ISL          AUTO         HPUX         PAD          LABEL

# setboot
Primary bootpath : 0/0/1/1.15.0
Alternate bootpath : 0/4/0/0/4/0.8.0

Autoboot is ON (enabled)
Autosearch is ON (enabled)

#
```

It looks as though this setup of a mirrored volume is complete in that the data is mirrored across two different disks on two different buses and that the mirrored volume is also bootable.

Another important procedure that is helpful to know is to create a new volume group and add a disk to it. We'll also put a logical volume on the disk and mount it. Figure 3-4 shows the steps involved in this procedure.

Figure 3-4 Create a New Volume Group, Add a Disk to it with a Logical Volume

[View full width]

```
# pvcreate /dev/rdsk/cxtxdx                        ; create physical volume for disk to be used in VG

# mkdir /dev/vg01                                 ; make directory for new vg01

# mknod /dev/vg01/group c 64 0x010000               ; mknod of vg01 ( minor 01 corresponds
➥ to vgnum)

# vgcreate /dev/vg01 /dev/dsk/cxtxdx                ; create vg01

# lvcreate -L 2000 /dev/vg01                       ; create default lvol1 in vg01 of 2 GBytes

# newfs -F vxfs -o largefiles /dev/vg01/rlvol1      ; create a new filesystem on lvol1

# mkdir /backup                                   ; create a directory on which to mount
➥ lvol1

# mount /dev/vg01/lvol1 /backup                     ; mount lvol1

# bdf                                             ; to see mounted filesystem
```

Since this is such an important procedure to know, let's go through an example of running this procedure.

In the previous example we performed a **diskinfo** to see all of the disks on our system. We'll use the disk at *4* in this example as the basis for creating a new volume group.

We'll run the steps as they appear in Figure 3-4 to create **vg01** and mount one of its logical volumes on **/backup**:

```
# pvcreate -B -f /dev/rdsk/c3t15d0
Physical volume "/dev/rdsk/c3t15d0" has been successfully created.
# mkdir /dev/vg01
# mknod /dev/vg01/group c 64 0x010000
# vgcreate /dev/vg01 /dev/dsk/c3t15d0
Increased the number of physical extents per physical volume to 4341.
Volume group "/dev/vg01" has been successfully created.
Volume Group configuration for /dev/vg01 has been saved in /etc/lvmconf/vg01.conf
```

```
# lvcreate -L 2000 /dev/vg01
Logical volume "/dev/vg01/lvol1" has been successfully created with
character device "/dev/vg01/rlvol1".
Logical volume "/dev/vg01/lvol1" has been successfully extended.
Volume Group configuration for /dev/vg01 has been saved in /etc/lvmconf/vg01.conf
# newfs -F vxfs -o largefiles /dev/vg01/rlvol1
    version 4 layout
   2048000 sectors, 2048000 blocks of size 1024, log size 1024 blocks
    unlimited inodes, largefiles supported
    2048000 data blocks, 2046392 free data blocks
    63 allocation units of 32768 blocks, 32768 data blocks
    last allocation unit has 16384 data blocks
# mkdir /backup
# mount /dev/vg01/lvol1 /backup
# bdf
Filesystem          kbytes    used   avail %used Mounted on
/dev/vg00/lvol3     204800   67065  129166   34% /
/dev/vg00/lvol1     299157   26222  243019   10% /stand
/dev/vg00/lvol8    2609152  138568 2316618    6% /var
/dev/vg00/lvol7    1056768  767077  271632   74% /usr
/dev/vg00/lvol5     204800    1338  190810    1% /tmp
/dev/vg00/lvol6    1056768  691085  342889   67% /opt
/dev/vg00/lvol4     512000    1234  478849    0% /home
/dev/vg01/lvol1    2048000    1606 1918502    0% /backup
# strings /etc/lvmtab
/dev/vg00
/dev/dsk/c1t15d0
/dev/dsk/c4t8d0
/dev/vg01
/dev/dsk/c3t15d0
#
```

We successfully created a new volume group in this example of size two GBytes. We'll later increase this size of this logical volume without unmounting **/backup**. We did not have to put boot-related files on this physical volume since we don't need to boot off of it.

The next series of figures give tips on performing some additional frequently performed LVM procedures.

Figure 3-5 Replace a Non-Mirrored Disk

```
# vgcfgrestore - /dev/vgXX /dev/rdsk/cxtxdx   ; volume group configuration restore

# vgchange -a y /dev/vgXX                    ; change volume group to available (-a y)

# newfs -F fstype /dev/vgXX/rlvolx            ; create filesystem for every lvol on physical volume

# mount /mountpointname                      ; mount every new filesystem


Notes:

Confirm you have /etc/lvmconf/vgXX.conf

vgcfgbackup is run automatically

Defective disk was not mirrored before it failed

Additional steps may be required on your system
```

In Figure 3-5, we had a non-mirrored and non-root disk that was defective and had to be replaced. After replacing the disk, the volume group information was restored for the specific disk with **vgcfgrestore**. We then changed the volume group to available, ran a **newfs** on the disk, and mounted the filesystems that used the disk.

In Figure 3-6 we will again replace a defective non-root disk, but this time, the disk will be mirrored.

Figure 3-6 Replace a Mirrored Disk

```
# vgcfgrestore -n /dev/vgXX /dev/rdsk/cxtxdx   ; volume group configuration restore

# vgchange -a y /dev/vgXX                     ; change volume group to available (-a y)

# vgsync /dev/vgXX                           ; resync logical volumes in volume group


Notes:

Defective disk was mirrored before it failed

Additional steps may be required on your system
```

In this example, we performed the same first two steps of restoring the volume group configuration and changing the volume

group to available. Because the disk was mirrored, we only have to synchronize the data on the new disk with that on its mirror with **vgsync**.

In Figure 3-7 we will again replace a mirrored disk, but this time the disk will have a boot area on it.

Figure 3-7 Replace Mirrored Disk Boot Disk

```
# vgcfgrestore -n /dev/vgXX /dev/rdsk/cxtxdx  ; volume group configuration restore

# vgchange -a y /dev/vgXX                    ; change volume group to available (-a y)

# vgsync /dev/vgXX                          ; resync logical volumes in volume group

# mkboot /dev/rdsk/cxtxdx                     ; create bootarea on disk

# mkboot -a "hpux lq" /dev/rdsk/cxtxdx       ; specify low quorum in boot area

# shutdown <desired options>                  ; reboot system to take effect

vgcfgbackup is run automatically

Notes:

Confirm you have /etc/lvmconf/vgXX.conf

Defective disk was mirrored before it failed

Addtional steps may be required on your system
```

In this example, we performed the same first three steps as the previous example, but we also have to create a boot area on the disk. The system has to be rebooted in order for this to take effect.

In Figure 3-8a we want to move a volume group onto a different system.

Figure 3-8a Exporting (Removing) and Importing a Volume Group Onto Another System

```
   On system 1:

# vgchange -a n /dev/vg01                   ; change volume group to unavailable (-a n) on 1st sys

# vgexport -v -m /tmp/mapfile -s /dev/vg01   ; make volume group unavailable on 1st system

                                          ; use -p -s -m options if you need shared for HA

# rcp /tmp/mapfile system2:/tmp/mapfile       ; copy mapfile to target or second system.

   On system 2:

# mkdir /dev/vg01                           ; make directory on 2nd system for volume group

# mknod /dev/vg01/group c 64 0xyy0000        ; create special file for vol group (yy=group number
                                            such as 00 for vg00, 01 for vg01, and so on)

# vgimport -v -m mapfile -s /dev/vg01        ; import volume group using mapfile copied from
                                            1st system or specify device filenames
                                            (use -s and -m in HA cluster)

# vgchange -a y /dev/vg01                   ; make volume group available

# mkdir /backup

# mount /dev/vg01/backup /backup             ;    add to /etc/fstab if you want this mounted at boot


Notes:
vgcfgbackup is run automatically

Addtional steps may be required on your system

The -s -p -m -v options are used in HA cluster environments to create sharable mapfile with
vgexport and -s -m on vgmport
```

In this example we performed the first three steps on the system from which we're moving the volume group. We make the volume group unavailable and then export the volume group. The mapfile from the first system needs to be copied to the second system. You don't have to specify the mapfile name if standard names such as **/dev/vg01** are used. In any event, the mapfile has to be copied to the system to which the volume group is being migrated.

On the second system we create a directory and device special file for the new volume group. We then import the mapfile for the volume group and make the volume group active. We then make the volume group available and perform the appropriate mounts.

There are many comments regarding a High Availability Cluster and the options used. In a ServiceGuard environment you don't have to make the volume group unavailable and you use some different options for sharing. Figure 3-8b shows the ServiceGuard steps:

Figure 3-8b Exporting (Removing) and Importing a Volume Group Onto Another System in a ServiceGuard Environment

```
   On system 1:

# vgexport -p -s -m /tmp/mapfile /dev/vg01   ; you'll get a volume group active warning

# rcp /tmp/mapfile system2:/tmp/mapfile        ; copy mapfile to target or second system.

   On system 2:

# mkdir /dev/vg01                            ; make directory on 2nd system for volume group

# mknod /dev/vg01/group c 64 0xyy0000          ; create special file for vol group (yy=group number
                                               such as 00 for vg00, 01 for vg01, and so on)

# vgimport -s -m mapfile /dev/vg01             ; import volume group using mapfile copied from
                                                1st system or specify device filenames

# vgchange -a y /dev/vg01                     ; make volume group available

# mkdir /backup

# mount /dev/vg01/backup /backup               ; add to /etc/fstab if you want this mounted at boot


Notes:
vgcfgbackup is run automatically

Addtional steps may be required on your system

The -s -p -m -v options are used in HA cluster environments to create sharable mapfile with
vgexport and -s -m on vgmport
```

These steps are very similar to those in Figure 3-8a. The options to **vgexport** and **vgimport** are different and you may want to look at those manual pages.

Figure 3-9 shows extending a VxFS filesystem by 800 MBytes.

Figure 3-9 Extend VxFS File System Using OnlineJFS

```
# lvextend -L 800 /dev/vgXX/lvolY           ; extend logical volume rlvolY by 800 MBytes

# fsadm -F vxfs -b 800M /mountpointname     ; use fsadm to extend or extendfs on non-JFS


Notes:

This is a VxFS filesystem and OnlineJFS is installed on system

Additional steps may be required on your system
```

This is a commonly run procedure because OnlineJFS is used in most high availability environments. Let's extend the logical volume in **vg01** that we earlier created from two GBytes to four GBytes using the steps in the following example:

```
# bdf
Filesystem          kbytes    used   avail %used Mounted on
/dev/vg00/lvol3     204800   67067  129164   34% /
/dev/vg00/lvol1     299157   26222  243019   10% /stand
/dev/vg00/lvol8    2609152  138519 2316664    6% /var
/dev/vg00/lvol7    1056768  767085  271624   74% /usr
/dev/vg00/lvol5     204800    1338  190810    1% /tmp
/dev/vg00/lvol6    1056768  691085  342889   67% /opt
/dev/vg00/lvol4     512000    1234  478849    0% /home
/dev/vg01/lvol1    2048000    1606 1918502    0% /backup
# lvextend -L 4000 /dev/vg01/lvol1
Logical volume "/dev/vg01/lvol1" has been successfully extended.
Volume Group configuration for /dev/vg01 has been saved in /etc/lvmconf/vg01.conf
# fsadm -F vxfs -b 4000M /backup
vxfs fsadm: /dev/vg01/rlvol1 is currently 2048000 sectors - size will be increased
# bdf
Filesystem          kbytes    used   avail %used Mounted on
/dev/vg00/lvol3     204800   67067  129164   34% /
/dev/vg00/lvol1     299157   26222  243019   10% /stand
/dev/vg00/lvol8    2609152  138521 2316663    6% /var
/dev/vg00/lvol7    1056768  767085  271624   74% /usr
```

```
/dev/vg00/lvol5      204800    1338  190810    1% /tmp
/dev/vg00/lvol6     1056768  691085  342889   67% /opt
/dev/vg00/lvol4      512000    1234  478849    0% /home
/dev/vg01/lvol1     4096000    2104 3838035    0% /backup
#
```

Note that this change from two GBytes to four GBytes took place while **/backup** was mounted. The ability to resize logical volumes while they're mounted is one of the advanced features of OnlineJFS.

If this directory were to be mounted at boot time, we would add an entry to **/etc/fstab**.

The Figure 3-10 shows changing a VxFS filesystem to support large files.

Figure 3-10 Change a Logical Volume to Support Large Files

```
# fsadm -F vxfs -o largefiles /dev/vgXX/rlvolY   ; use fsadm and specify largefile option


Notes:

This is VxFS filesystem, you can substitute hfs for vxfs in above example for an hfs filesystem and use
      lvolY in place of rlvolY
Unmount filesystem before running

Additional steps may be required on your system
```

Using **fsadm**, we specify the raw logical volume on which we want large files supported.

This change would be unnecessary had the logical volume been created to support large files.

[Team LiB]



Adding Disks

In this section, we're going to do some work with XP 256 disks. We want to create ten volume groups with five primary disks and five alternate disks per volume group. The XP 256 has been replaced by newer models but all of the same principles apply to newer models.

Let's take a look at the file that contains the physical disks to be used for the primary and alternate paths on the XP 256. The XP 256 is an advanced storage device that has in it the capability to fail over to an alternate controller should the primary controller fail. The same set of disks are connected to the primary and alternate controllers but the disks are given two different sets of device files. One set is for the disks when connected to the primary controller, and the second set is for the same disks when connected to the alternate controller. This is the same concept that you may have come across if you are a ServiceGuard user. There are a set of disks connected through two different paths, so you must define the disks with different names depending on whether they are connected through the primary or alternate path. The following is a listing of the file **pri**, containing the primary disks in groups of five:

```
c9t0d0 c9t0d1 c9t0d2 c8t0d0  c8t0d1
c7t0d0 c7t0d1 c7t0d2 c10t0d0 c10t0d1
c9t0d3 c9t0d4 c9t0d5 c8t0d3  c8t0d4
c7t0d3 c7t0d4 c7t0d5 c10t0d3 c10t0d4
c9t0d6 c9t0d7 c9t1d0 c8t0d6  c8t0d7
c7t0d6 c7t0d7 c7t1d0 c10t0d6 c10t0d7
c9t1d1 c9t1d2 c9t1d3 c8t1d1  c8t1d2
c7t1d1 c7t1d2 c7t1d3 c10t1d1 c10t1d2
c9t1d4 c9t1d5 c9t1d6 c8t1d4  c8t1d5
c7t1d4 c7t1d5 c7t1d6 c10t1d4 c10t1d5
```

Notice that in this listing, the disks have been grouped in fives. Each group of five disks will constitute a volume group. There are a total of *10* groups of five disks that will be placed in volume groups *vgu01* through *vgu10*.

There will also be an alternate group of five disks. The alternate disks will be used in the event of a disk controller failover as described earlier. The following is a listing of the file **alt**, which contains a list of alternate disks in groups of five:

```
c8t8d0  c8t8d1  c8t8d2  c9t8d0 c9t8d1
c10t8d0 c10t8d1 c10t8d2 c7t8d0 c7t8d1
c8t8d3  c8t8d4  c8t8d5  c9t8d3 c9t8d4
c10t8d3 c10t8d4 c10t8d5 c7t8d3 c7t8d4
c8t8d6  c8t8d7  c8t9d0  c9t8d6 c9t8d7
c10t8d6 c10t8d7 c10t9d0 c7t8d6 c7t8d7
c8t9d1  c8t9d2  c8t9d3  c9t9d1 c9t9d2
c10t9d1 c10t9d2 c10t9d3 c7t9d1 c7t9d2
c8t9d4  c8t9d5  c8t9d6  c9t9d4 c9t9d5
c10t9d4 c10t9d5 c10t9d6 c7t9d4 c7t9d5
```

There are a total of *10* groups of alternate disks shown in this listing that will also be part of volume groups *vgu01* through *vgu10*. Using these primary and alternate disks that have been set up on the XP 256, we'll set up the appropriate volumes on the host system. In this example, the host system is a V-Class system.

Let's now cover the steps to create one of these volume groups. First, we'll create a physical volume for each of the disks in the volume group with the **pvcreate** command. Next, we'll create a directory for the volume group with **mkdir**, then we'll create a device special file for the volume group within the directory with **mknod**. This will set up the directory and special file required for the first of the *10* volume groups. Then we'll create the volume group in which the five primary disks will be contained using **vgcreate**. We'll specify the first disk when we create the volume group and then include the other disks in the volume group with **vgextend**. Then we extend the volume group with **vgextend** to include the five alternate disks. The final step is to create a single logical volume for the entire volume group. You might want to create several logical volumes within a volume group, but in our example, we need only one logical volume that consumes the entire capacity of the volume group, which is 8755 physical extents. The following procedure is the list of manual steps to create the first volume group:

```
# pvcreate /dev/rdsk/c9t0d0    # run for each of the 5 pri disks
# mkdir /dev/vgu01            # make dir for first vol group

# mknod /dev/vgu01/group -c 64 0x01000
                              # create special file with major
```

```
                                  and minor numbers shown

# vgcreate /dev/vgu01 /dev/dsk/c9t0d0
                                # place first disk in volume group

# vgextend /dev/vgu01 /dev/dsk/c9t0d1
                                # extend volume with remaining four
                                   primary disks disks

# vgextend /dev/vgu01 /dev/dsk/c8t8d0
                               # extend volume group to include five
                                 alternate disks

# lvcreate -l 8755 /dev/vgu01  # creates lvol1 (lvol1 by default)
                                  that consumes all 8755 extents
```

We completed the procedure for only one disk, and there are nine additional disks in this volume group. In addition, there are another nine volume groups for which this procedure must be completed. That is a total of an additional 99 disks for which various commands must be run. There is a lot of room for error with that much typing involved, so this is an ideal process to automate.

-- Read this only if you wish to see how to automate the procedure --

Since there are a primary set of disks and an alternate set of disks we'll write a short program to automate each procedure. The following program performs all of the steps required to create a physical volume for each disk, create a volume group, and include the primary disks:

```
#!/bin/ksh
set -x         ;set tracing on

  vgnum=$1     ;first item on each line is the volume group no.
  shift          ;shift to get to first disk
for i in $*    ;run pvcreate for every disk name in first line
do
   pvcreate /dev/rdsk/$i
done


reada            ;pause program to view what has been run


mkdir /dev/vgu$vgnum                        ;mkdir for volume group
mknod /dev/vgu$vgnum/group c 0x$(vgnum)0000  ;mknod for volume group
vgcreate /dev/vgu$vgnum /dev/dsk/$1          ;vgcreate 1st disk in vg


shift            ;shift over to second disk

for i in $*      ;extend volume group to include remaining four disks
do
   vgextend /dev/vgu$vgnum /dev/dsk/$i
done


lvcreate -l 8755 /dev/vgu$vgnum   ;create single log vol for entire vg
```

I use **set -x** in this file to turn on execution tracing. I always do this when first debugging a shell program so I can see the lines in the shell program as they are executed. The line being executed will appear with a "+" in front of it, followed by what you would normally see when the program is run. The **read a** is a way of pausing the program to wait for input so I can review what has been run to that point of the program.

In order for this program to run, we must slightly modify the file containing the primary disk devices and add the volume group number to the beginning of each line. In addition, I decided to call the program from the file that has the primary disks in it and operate on one line of disks at a time. The following listing shows the updated file containing the name of the shell program in the previous listing (**vg.sh**), followed by the volume group number and then the list of five primary disks names for each volume group:

```
#vg.sh 01 c9t0d0 c9t0d1 c9t0d2 c8t0d0 c8t0d1
#read a
#vg.sh 02 c7t0d0 c7t0d1 c7t0d2 c10t0d0 c10t0d1
#read a
#vg.sh 03 c9t0d3 c9t0d4 c9t0d5 c8t0d3 c8t0d4
#read a
#vg.sh 04 c7t0d3 c7t0d4 c7t0d5 c10t0d3 c10t0d4
```

```
#read a
#vg.sh 05 c9t0d6 c9t0d7 c9t1d0 c8t0d6 c8t0d7
#read a
#vg.sh 06 c7t0d6 c7t0d7 c7t1d0 c10t0d6 c10t0d7
#read a
#vg.sh 07 c9t1d1 c9t1d2 c9t1d3 c8t1d1 c8t1d2
#read a
#vg.sh 08 c7t1d1 c7t1d2 c7t1d3 c10t1d1 c10t1d2
#read a
#vg.sh 09 c9t1d4 c9t1d5 c9t1d6 c8t1d4 c8t1d5
#read a
#vg.sh 10 c7t1d4 c7t1d5 c7t1d6 c10t1d4 c10t1d5
```

The **read a** between lines of this file will pause and wait for you to enter a *Return* before the next line will be executed. I did this in case I decided to run several lines and I wanted to check the results between the execution of lines.

We can now uncomment the first line of the file and type the file name **pri**, which will call **vg.sh** and run the program (you must give appropriate permissions to the files and make sure both **vg.sh** and **pri** are executable). I like to run such files one line at a time and check the volume groups as they are created. The script is written to run one line at a time but is easily modifiable to run all ten lines.

We need to do much less work with the alternate disk names. The physical volumes have already been created, and the volume group and single logical volume have already been set up in **vg.sh**. We'll create another script called **vga.sh**, "a" for alternate), in which we'll extend the volume group to include the alternate name for each disk. This script is shown in the listing below:

```
#!/bin/ksh
set -x              ;set tracing on

  vgnum=$1          ;first item on each line is the volume group number
  shift             ;shift to get to first disk


  for i in $*       ;extend vol group to include all five disks on line

    vgextend /dev/vgu$vgnum /dev/dsk/$i

done
```

This script performs only the task of extending the volume group *vgnum* to include all five disks that appear on the line. Much like the file **pri**, the file **alt** will call the script **vga.alt** as shown in the following listing:

```
#vga.sh 01 c8t8d0  c8t8d1  c8t8d2  c9t8d0 c9t8d1
#vga.sh 02 c10t8d0 c10t8d1 c10t8d2 c7t8d0 c7t8d1
#vga.sh 03 c8t8d3  c8t8d4  c8t8d5  c9t8d3 c9t8d4
#vga.sh 04 c10t8d3 c10t8d4 c10t8d5 c7t8d3 c7t8d4
#vga.sh 05 c8t8d6  c8t8d7  c8t9d0  c9t8d6 c9t8d7
#vga.sh 06 c10t8d6 c10t8d7 c10t9d0 c7t8d6 c7t8d7
#vga.sh 07 c8t9d1  c8t9d2  c8t9d3  c9t9d1 c9t9d2
#vga.sh 08 c10t9d1 c10t9d2 c10t9d3 c7t9d1 c7t9d2
#vga.sh 09 c8t9d4  c8t9d5  c8t9d6  c9t9d4 c9t9d5
#vga.sh 10 c10t9d4 c10t9d5 c10t9d6 c7t9d4 c7t9d5
```

You would uncomment the line for which you wanted to run the script. Again, you could run all ten lines, but I like to check what has taken place after each line has been run. You could add **read a** between the lines of this file if you wanted to run several lines and have a pause between them to check the results.

These two scripts automate a lot of typing. There are 100 disks for which commands must be run as well as other Logical Volume Manager commands. This is the type of HP-UX system administration task that is ideally suited to shell programming.

---------- End of Automated Procedure ----------

I completed the steps that had to be run for the additional disks to complete the work, such as the **vgcreate** for the additional four disks and the **vgextend** for the additional nine disk devices. I included only the first disk in the examples so that you could see the initial step that had to be run.

We don't have to set up any RAID levels within the primary or alternate volume because this is being done internally to the XP 256.

The following **vgdisplay** listing shows the disks we set up for volume group *vgu01* with both the group of five primary and alternate disks:

```
# vgdisplay /dev/vgu01 -v


VG Name                     /dev/vgu01
VG Write Access             read/write
VG Status                   available
Max LV                      255
Cur LV                      1
Open LV                     1
Max PV                      16
Cur PV                      5
Act PV                      5
Max PE per PV               1751
VGDA                        10
PE Size (Mbytes)            4
Total PE                    8755
Alloc PE                    8755
Free PE                     0
Total PVG                   0
Total Spare PVs             0
Total Spare PVs in use      0

   --- Logical volumes ---
   LV Name                      /dev/vgu01/lvol1
   LV Status                    available/syncd
   LV Size (Mbytes)             35020
   Current LE                   8755
   Allocated PE                 8755
   Used PV                      5


   --- Physical volumes ---
   PV Name                      /dev/dsk/c9t0d0
   PV Name                      /dev/dsk/c8t8d0 Alternate Link
   PV Status                    available
   Total PE                     1751
   Free PE                      0

   PV Name                      /dev/dsk/c9t0d1
   PV Name                      /dev/dsk/c8t8d1 Alternate Link
   PV Status                    available
   Total PE                     1751
   Free PE                      0

   PV Name                      /dev/dsk/c9t0d2
   PV Name                      /dev/dsk/c8t8d2 Alternate Link
   PV Status                    available
   Total PE                     1751
   Free PE                      0

   PV Name                      /dev/dsk/c8t0d0
   PV Name                      /dev/dsk/c9t8d0 Alternate Link
   PV Status                    available
   Total PE                     1751
   Free PE                      0

   PV Name                      /dev/dsk/c8t0d1
   PV Name                      /dev/dsk/c9t8d1 Alternate Link
   PV Status                    available
   Total PE                     1751
   Free PE                      0
```

There are some points of interest to cover in this **vgdisplay**. The first is that there is a primary and alternate path to the same disk because we defined them earlier. For instance, the first disk in the volume group has a primary pathname of *c9t0d0* and an alliterate path name of *c8t8d0*. Next, both the volume group *vgu01* and the only logical volume in it, *lvol1,* consist of a total of 8755 PE or physical extents (the size of volume group is PE x PE size, or 8755 x 4MB in our case).

We should also check one logical volume on *vgu01,* called *lvol1*. We can check the parameters of this logical volume with the **lvdisplay** command as shown in the following example:

```
# lvdisplay -v /dev/vgu01/l*
```

```
--- Logical volumes ---
LV Name                     /dev/vgu01/lvol1
VG Name                     /dev/vgu01
LV Permission               read/write
LV Status                   available/syncd
Mirror copies               0
Consistency Recovery        MWC
Schedule                    parallel
LV Size (Mbytes)            35020
Current LE                  8755
Allocated PE                8755
Stripes                     0
Stripe Size (Kbytes)        0
Bad block                   on
Allocation                  strict
IO Timeout (Seconds)        default

   --- Distribution of logical volume ---
   PV Name            LE on PV  PE on PV
   /dev/dsk/c9t0d0    1751      1751
   /dev/dsk/c9t0d1    1751      1751
   /dev/dsk/c9t0d2    1751      1751
   /dev/dsk/c8t0d0    1751      1751
   /dev/dsk/c8t0d1    1751      1751

   --- Logical extents ---
   LE    PV1                 PE1   Status 1
   00000 /dev/dsk/c9t0d0     00000 current
   00001 /dev/dsk/c9t0d0     00001 current
   00002 /dev/dsk/c9t0d0     00002 current
   00003 /dev/dsk/c9t0d0     00003 current
   00004 /dev/dsk/c9t0d0     00004 current
   00005 /dev/dsk/c9t0d0     00005 current
   00006 /dev/dsk/c9t0d0     00006 current
   00007 /dev/dsk/c9t0d0     00007 current
   00008 /dev/dsk/c9t0d0     00008 current
   00009 /dev/dsk/c9t0d0     00009 current
   00010 /dev/dsk/c9t0d0     00010 current
   00011 /dev/dsk/c9t0d0     00011 current
   00012 /dev/dsk/c9t0d0     00012 current
   00013 /dev/dsk/c9t0d0     00013 current
   00014 /dev/dsk/c9t0d0     00014 current
   00015 /dev/dsk/c9t0d0     00015 current
   00016 /dev/dsk/c9t0d0     00016 current
   00017 /dev/dsk/c9t0d0     00017 current
   00018 /dev/dsk/c9t0d0     00018 current
   00019 /dev/dsk/c9t0d0     00019 current
   00020 /dev/dsk/c9t0d0     00020 current
   00021 /dev/dsk/c9t0d0     00021 current
   00022 /dev/dsk/c9t0d0     00022 current
   00023 /dev/dsk/c9t0d0     00023 current
   00024 /dev/dsk/c9t0d0     00024 current
   00025 /dev/dsk/c9t0d0     00025 current
   00026 /dev/dsk/c9t0d0     00026 current
   00027 /dev/dsk/c9t0d0     00027 current
   00028 /dev/dsk/c9t0d0     00028 current
   00029 /dev/dsk/c9t0d0     00029 current
   00030 /dev/dsk/c9t0d0     00030 current
   00031 /dev/dsk/c9t0d0     00031 current
   00032 /dev/dsk/c9t0d0     00032 current
              -
              -

              -
   08733 /dev/dsk/c8t0d1     01729 current
   08734 /dev/dsk/c8t0d1     01730 current
   08735 /dev/dsk/c8t0d1     01731 current
   08736 /dev/dsk/c8t0d1     01732 current
   08737 /dev/dsk/c8t0d1     01733 current
   08738 /dev/dsk/c8t0d1     01734 current
   08739 /dev/dsk/c8t0d1     01735 current
   08740 /dev/dsk/c8t0d1     01736 current
   08741 /dev/dsk/c8t0d1     01737 current
   08742 /dev/dsk/c8t0d1     01738 current
   08743 /dev/dsk/c8t0d1     01739 current
```

```
   08744 /dev/dsk/c8t0d1    01740 current
   08745 /dev/dsk/c8t0d1    01741 current
   08746 /dev/dsk/c8t0d1    01742 current
   08747 /dev/dsk/c8t0d1    01743 current
   08748 /dev/dsk/c8t0d1    01744 current
   08749 /dev/dsk/c8t0d1    01745 current
   08750 /dev/dsk/c8t0d1    01746 current
   08751 /dev/dsk/c8t0d1    01747 current
   08752 /dev/dsk/c8t0d1    01748 current
   08753 /dev/dsk/c8t0d1    01749 current
   08754 /dev/dsk/c8t0d1    01750 current
```

This listing has been abbreviated where the three dots are shown. Only the beginning of the first disk and end of the last disk are shown. The **lvdisplay** does indeed show the five primary disks of which the logical volume is comprised.

We would create a file system on each logical volume using **newfs** and enable large files. We would then add these to /**etc/fstab** because we want them to be mounted both now and at the time of system boot, as shown in the following listing:

```
# System /etc/fstab file.  Static information about the file systems
# See fstab(4) and sam(1M) for further details on configuring devices.
/dev/vg00/lvol3 / vxfs delaylog 0 1
/dev/vg00/lvol1 /stand hfs defaults 0 1
/dev/vg00/lvol4 /opt vxfs delaylog 0 2
/dev/vg00/lvol5 /tmp vxfs delaylog 0 2
/dev/vg00/lvol6 /usr vxfs delaylog 0 2
/dev/vg00/lvol7 /var vxfs delaylog 0 2
/dev/vg00/lvol8 /home vxfs delaylog 0 2
/dev/vg00/lvol9 ... swap pri=0 0 0
/dev/vgapp/lvol1 /app vxfs rw,suid,largefiles,delaylog,datainlog 0 2
/dev/vgu01/lvol1 /u01 vxfs rw,suid,largefiles,delaylog,datainlog 0 2
/dev/vgu02/lvol1 /u02 vxfs rw,suid,largefiles,delaylog,datainlog 0 2
/dev/vgu03/lvol1 /u03 vxfs rw,suid,largefiles,delaylog,datainlog 0 2
/dev/vgu04/lvol1 /u04 vxfs rw,suid,largefiles,delaylog,datainlog 0 2
/dev/vgu05/lvol1 /u05 vxfs rw,suid,largefiles,delaylog,datainlog 0 2
/dev/vgu06/lvol1 /u06 vxfs rw,suid,largefiles,delaylog,datainlog 0 2
/dev/vgu07/lvol1 /u07 vxfs rw,suid,largefiles,delaylog,datainlog 0 2
/dev/vgu08/lvol1 /u08 vxfs rw,suid,largefiles,delaylog,datainlog 0 2
/dev/vgu09/lvol1 /u09 vxfs rw,suid,largefiles,delaylog,datainlog 0 2
/dev/vgu10/lvol1 /u10 vxfs rw,suid,largefiles,delaylog,datainlog 0 2
```

This listing shows our ten logical volumes with the parameters we set up such as VxFS and support for large files as the last ten entries.

The next section provides tips on various procedures you may want to perform using logical volume manager. These are procedures I have used many times and they may help you when you need to perform similar tasks.

Reconfiguring Some Disks - An Example of Using Some Logical Volume Commands

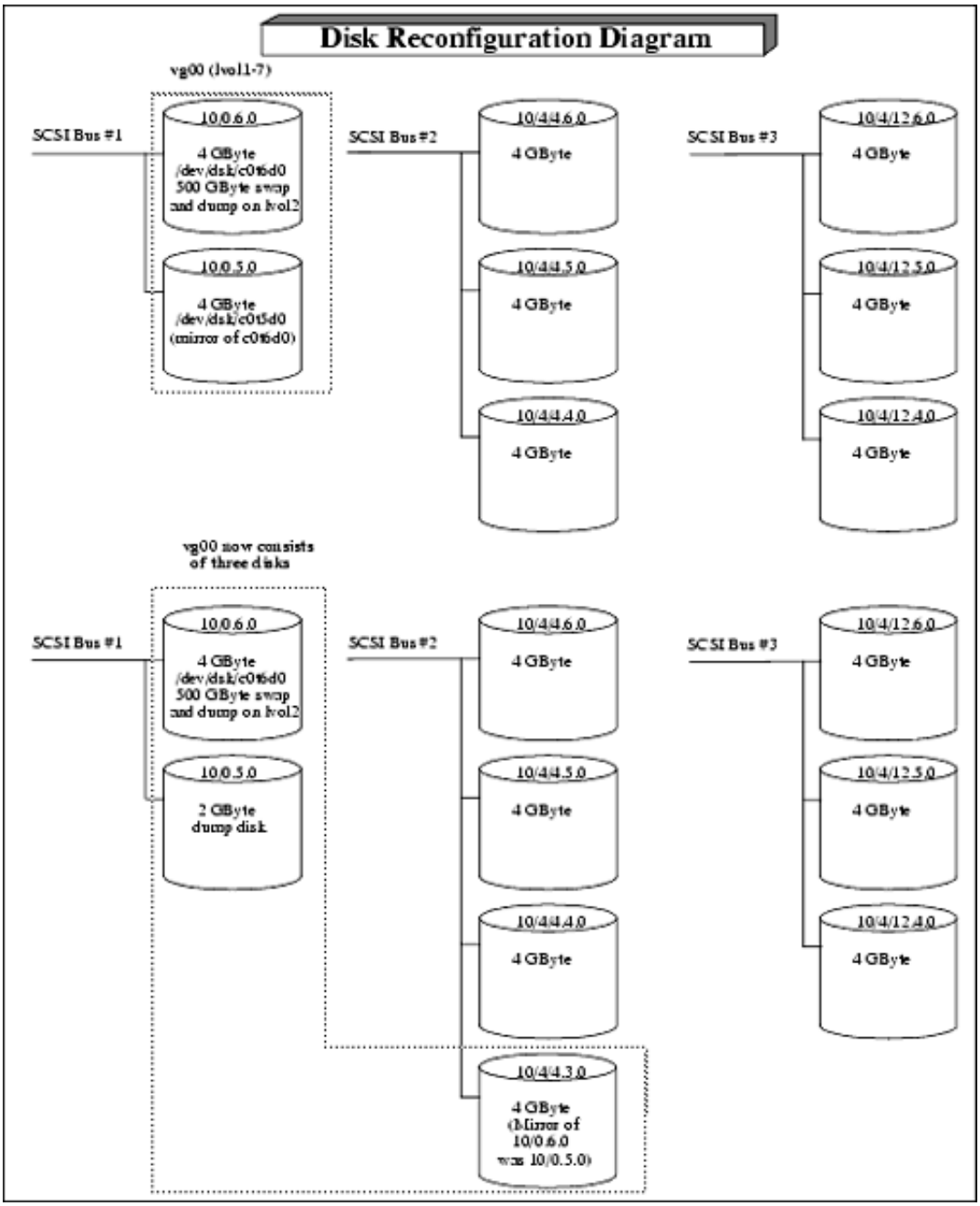
I have always advised in my books and articles to take great care when you first set up disks on your HP-UX systems to make sure the disk layout you select is one you can live with for a long time. No matter how careful you are, however, you often need to perform some logical volume reconfiguration. It is much more difficult to make changes to an existing logical volume layout than it is to set up your system correctly when it is first installed. This section describes the steps performed to make some changes to the dump and mirror on an existing system.

This is not a procedure you should follow. It is an example of some advanced Logical Volume Manager (LVM) commands used to reconfigure some disks on a specific system. It is a good procedure for illustrating how several LVM commands can be used.

Why Change?

Figure 3-11 shows the original configuration of disks on a system and the updated configuration we wish to implement.

Figure 3-11. Disk Reconfiguration Diagram



The overall objective here is to move the 4 GByte disk used as the mirror of the root disk to a different SCSI channel and to install a 2 GByte dump device on the same SCSI channel as the root disk.

The procedure consists of several parts. The first is to obtain a snapshot of the system before any reconfigurations. This serves two purposes. The first is to have documentation of the original system configuration that can be included in the system administration notebook. Should any questions arise in the future as to the original configuration and changes made to it, the original configuration will be in the system administration notebook. The second purpose of having this information is to have all of the relevant information about the configuration available as you proceed with the reconfiguration process.

The second part of the procedure is to shut down the system, install the new 2 GByte disk, and move the 4 GByte disk.

The last part of the procedure is to perform the system administration reconfiguration of the dump and mirror.

Figures 3-12, 3-13, and 3-14 show a flowchart depicting the procedure we'll follow throughout this section.

The step numbers in the upcoming procedure correspond to the step numbers shown in these figures. Let's now proceed beginning with the snapshot of the system.

Figure 3-12. Disk Reconfiguration Flow Diagram - Part 1 (Rearrange Disks)

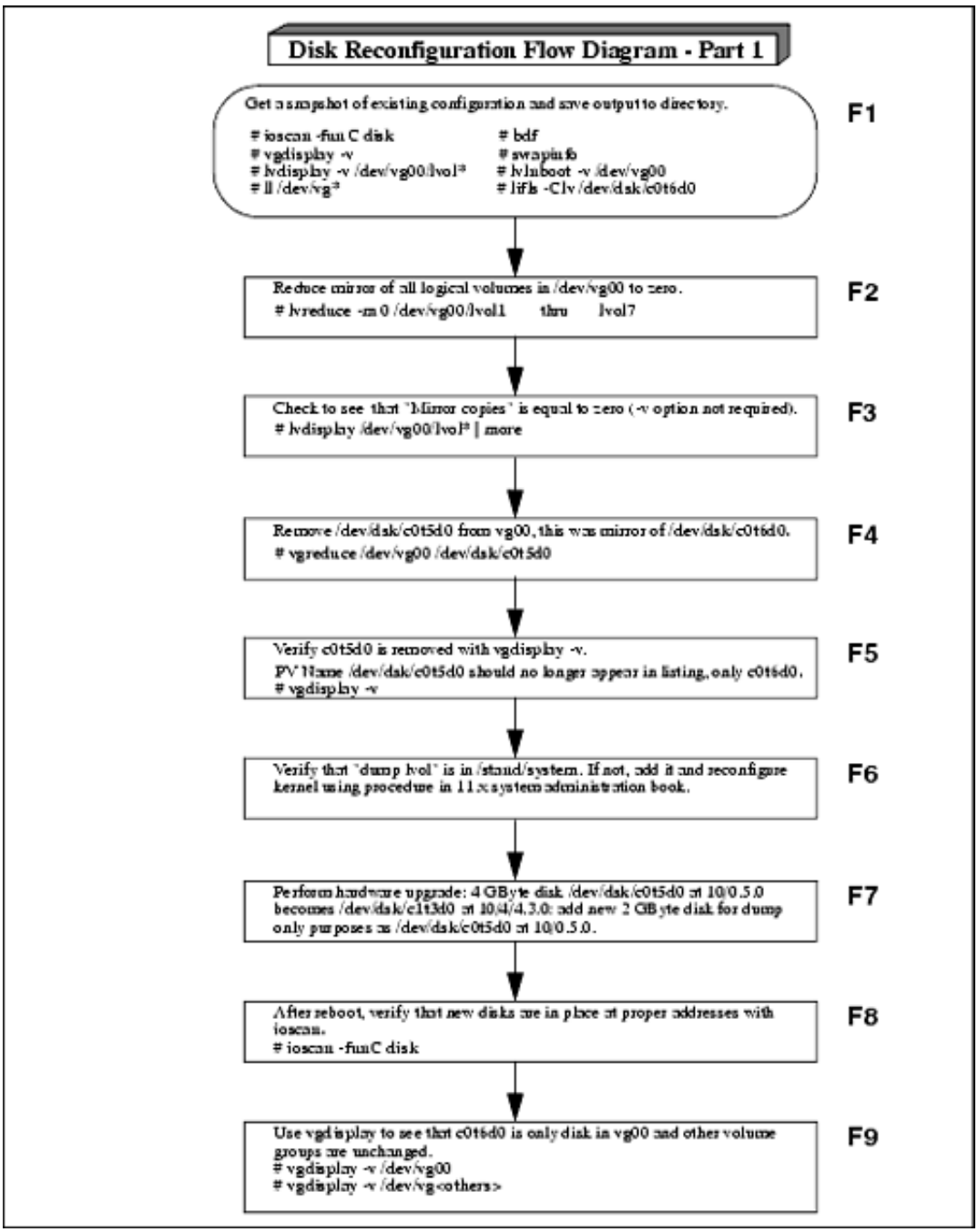


Figure 3-13. Disk Reconfiguration Flow Diagram - Part 2 (Set Up Dump)

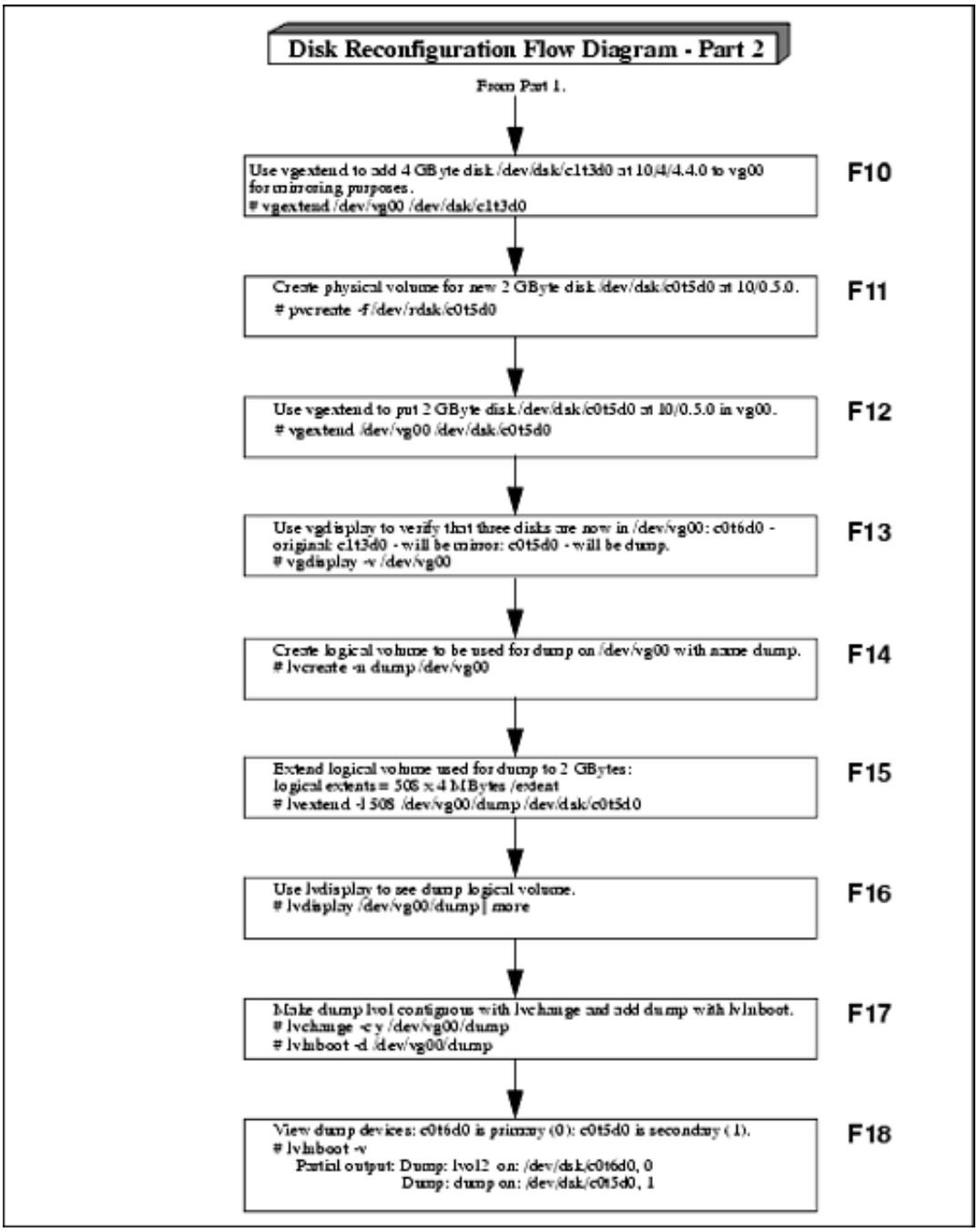
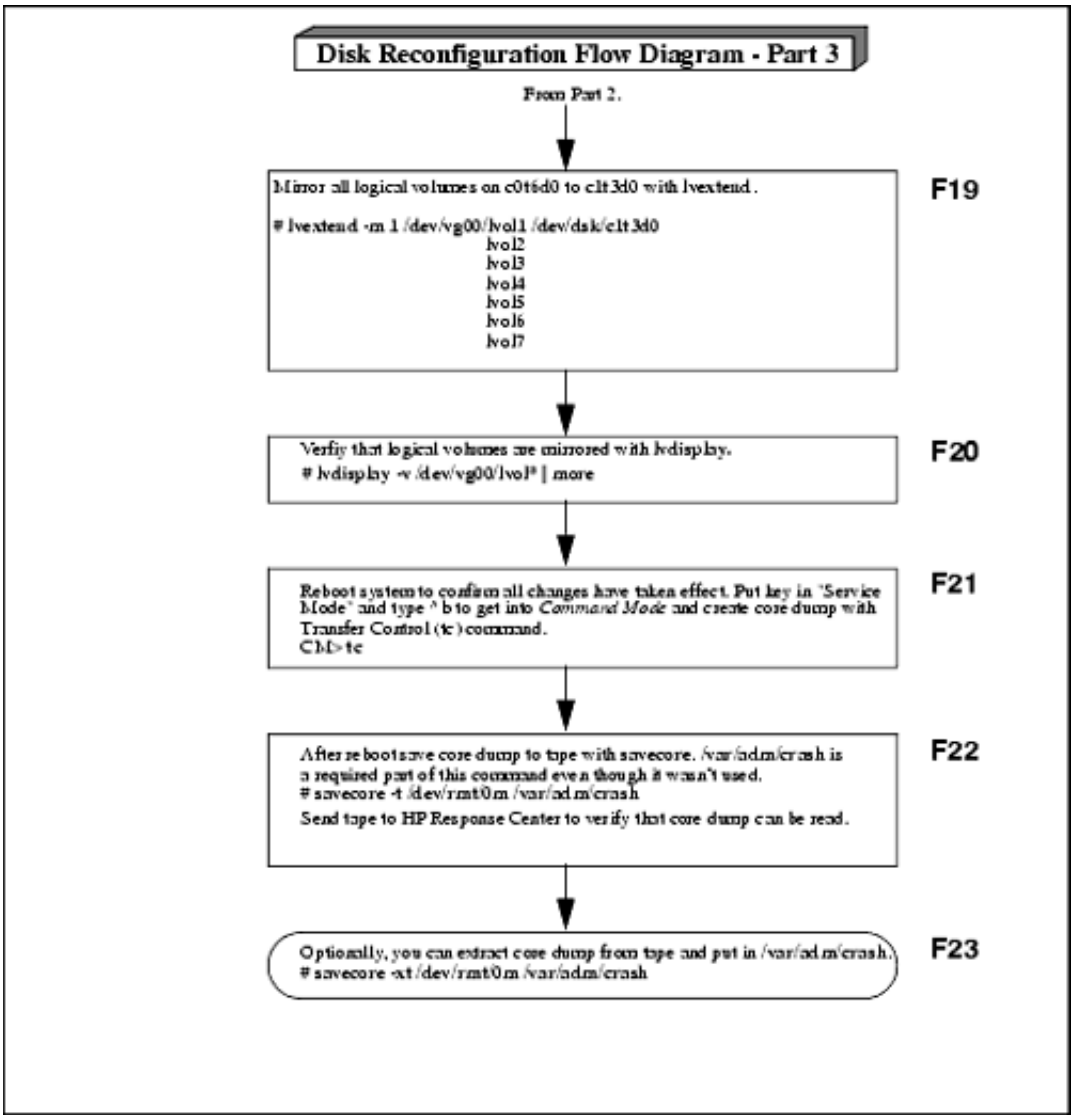


Figure 3-14. Disk Reconfiguration Flow Diagram - Part 3 (Reinstate Mirrors)



F1 - to scan

First let's run **ioscan** to see the disks on the system.

```
$ /usr/sbin/ioscan -funC disk

Class     I  H/W Path      Driver      S/W State H/W Type  Description
===================================================================
disk      2  10/0.5.0      sdisk       CLAIMED   DEVICE    SEAGATE ST15150W
                         /dev/dsk/c0t5d0   /dev/rdsk/c0t5d0
disk      3  10/0.6.0      sdisk       CLAIMED   DEVICE    SEAGATE ST15150W
                         /dev/dsk/c0t6d0   /dev/rdsk/c0t6d0
disk      6  10/4/4.4.0    sdisk       CLAIMED   DEVICE    SEAGATE ST15150W
                         /dev/dsk/c1t4d0   /dev/rdsk/c1t4d0
disk      7  10/4/4.5.0    sdisk       CLAIMED   DEVICE    SEAGATE ST15150W
                         /dev/dsk/c1t5d0   /dev/rdsk/c1t5d0
disk      8  10/4/4.6.0    sdisk       CLAIMED   DEVICE    SEAGATE ST15150W
                         /dev/dsk/c1t6d0   /dev/rdsk/c1t6d0
disk      9  10/4/12.4.0   sdisk       CLAIMED   DEVICE    SEAGATE ST15150W
                         /dev/dsk/c2t4d0   /dev/rdsk/c2t4d0
disk      10 10/4/12.5.0   sdisk       CLAIMED   DEVICE    SEAGATE ST15150W
                         /dev/dsk/c2t5d0   /dev/rdsk/c2t5d0
disk      11 10/4/12.6.0   sdisk       CLAIMED   DEVICE    SEAGATE ST15150W
                         /dev/dsk/c2t6d0   /dev/rdsk/c2t6d0
disk      5  10/12/5.2.0   sdisk       CLAIMED   DEVICE    TOSHIBA CD-ROM XM-5401TA
```

```
                    /dev/dsk/c3t2d0   /dev/rdsk/c3t2d0
```

Note that the disks in this configuration correspond to those on the top of Figure 3-12 (Disk Reconfiguration Diagram.) We haven't yet looked at the logical volume information related to these disks, only their physical addresses.

F1 - vgdisplay

Next run **vgdisplay** to see the volume groups. **lvol2** on **vg00** is the dump logical volume we are going to move to a separate 2 GByte disk. We don't yet know if **lvol1-7** on **vg00** are all mirrored.

```
# vgdisplay -v
--- Volume groups ---
VG Name                 /dev/vg00
VG Write Access         read/write
VG Status               available
Max LV                  255
Cur LV                  7
Open LV                 7
Max PV                  16
Cur PV                  2
Act PV                  2
Max PE per PV           1023
VGDA                    4
PE Size (Mbytes)        4
Total PE                2046
Alloc PE                688
Free PE                 1358
Total PVG               0

--- Logical volumes ---
LV Name                 /dev/vg00/lvol1
LV Status               available/syncd
LV Size (Mbytes)        92
Current LE              23
Allocated PE            46
Used PV                 2

LV Name                 /dev/vg00/lvol2
LV Status               available/syncd
LV Size (Mbytes)        500
Current LE              125
Allocated PE            250
Used PV                 2

LV Name                 /dev/vg00/lvol3
LV Status               available/syncd
LV Size (Mbytes)        20
Current LE              5
Allocated PE            10
Used PV                 2

LV Name                 /dev/vg00/lvol4
LV Status               available/syncd
LV Size (Mbytes)        252
Current LE              63
Allocated PE            126
Used PV                 2

LV Name                 /dev/vg00/lvol5
LV Status               available/syncd
LV Size (Mbytes)        32
Current LE              8
Allocated PE            16
Used PV                 2

LV Name                 /dev/vg00/lvol6
LV Status               available/syncd
LV Size (Mbytes)        320
Current LE              80
Allocated PE            160
Used PV                 2
LV Name                 /dev/vg00/lvol7
```

```
LV Status              available/syncd
LV Size (Mbytes)       160
Current LE             40
Allocated PE           80
Used PV                2

   --- Physical volumes ---
PV Name                /dev/dsk/c0t6d0
PV Status              available
Total PE               1023
Free PE                679

PV Name                /dev/dsk/c0t5d0
PV Status              available
Total PE               1023
Free PE                679
```

F1 - lvdisplay

View detailed logical volume information with **lvdisplay**. Note that all of these logical volumes are mirrored and that each has "current" status. Only **lvol1** and **lvol2** are shown in the listing. **lvol3** through **lvol7** are not shown.

```
# lvdisplay -v /dev/vg00/lvol*
--- Logical volumes ---
LV Name                /dev/vg00/lvol1
VG Name                /dev/vg00
LV Permission          read/write
LV Status              available/syncd
Mirror copies          1
Consistency Recovery   MWC
Schedule               parallel
LV Size (Mbytes)       92
Current LE             23
Allocated PE           46
Stripes                0
Stripe Size (Kbytes)   0
Bad block              off
Allocation             strict/contiguous

--- Distribution of logical volume ---
PV Name            LE on PV  PE on PV
/dev/dsk/c0t6d0    23        23
/dev/dsk/c0t5d0    23        23

--- Logical extents ---
LE   PV1                PE1  Status 1 PV2                PE2  Status 2
0000 /dev/dsk/c0t6d0    0000 current  /dev/dsk/c0t5d0    0000 current
0001 /dev/dsk/c0t6d0    0001 current  /dev/dsk/c0t5d0    0001 current
0002 /dev/dsk/c0t6d0    0002 current  /dev/dsk/c0t5d0    0002 current
0003 /dev/dsk/c0t6d0    0003 current  /dev/dsk/c0t5d0    0003 current
0004 /dev/dsk/c0t6d0    0004 current  /dev/dsk/c0t5d0    0004 current
0005 /dev/dsk/c0t6d0    0005 current  /dev/dsk/c0t5d0    0005 current
0006 /dev/dsk/c0t6d0    0006 current  /dev/dsk/c0t5d0    0006 current
0007 /dev/dsk/c0t6d0    0007 current  /dev/dsk/c0t5d0    0007 current
0008 /dev/dsk/c0t6d0    0008 current  /dev/dsk/c0t5d0    0008 current
0009 /dev/dsk/c0t6d0    0009 current  /dev/dsk/c0t5d0    0009 current
0010 /dev/dsk/c0t6d0    0010 current  /dev/dsk/c0t5d0    0010 current
0011 /dev/dsk/c0t6d0    0011 current  /dev/dsk/c0t5d0    0011 current
0012 /dev/dsk/c0t6d0    0012 current  /dev/dsk/c0t5d0    0012 current
0013 /dev/dsk/c0t6d0    0013 current  /dev/dsk/c0t5d0    0013 current
0014 /dev/dsk/c0t6d0    0014 current  /dev/dsk/c0t5d0    0014 current
0015 /dev/dsk/c0t6d0    0015 current  /dev/dsk/c0t5d0    0015 current
0016 /dev/dsk/c0t6d0    0016 current  /dev/dsk/c0t5d0    0016 current
0017 /dev/dsk/c0t6d0    0017 current  /dev/dsk/c0t5d0    0017 current
0018 /dev/dsk/c0t6d0    0018 current  /dev/dsk/c0t5d0    0018 current
0019 /dev/dsk/c0t6d0    0019 current  /dev/dsk/c0t5d0    0019 current
0020 /dev/dsk/c0t6d0    0020 current  /dev/dsk/c0t5d0    0020 current
0021 /dev/dsk/c0t6d0    0021 current  /dev/dsk/c0t5d0    0021 current
0022 /dev/dsk/c0t6d0    0022 current  /dev/dsk/c0t5d0    0022 current

LV Name                /dev/vg00/lvol2
VG Name                /dev/vg00
```

```
LV Permission           read/write
LV Status               available/syncd
Mirror copies           1
Consistency Recovery    MWC
Schedule                parallel
LV Size (Mbytes)        500
Current LE              125
Allocated PE            250
Stripes                 0
Stripe Size (Kbytes)    0
Bad block               off
Allocation              strict/contiguous

--- Distribution of logical volume ---

PV Name            LE on PV  PE on PV
/dev/dsk/c0t6d0    125       125
/dev/dsk/c0t5d0    125       125

--- Logical extents ---
LE   PV1                PE1  Status 1 PV2                PE2  Status 2
0000 /dev/dsk/c0t6d0    0023 current  /dev/dsk/c0t5d0    0023 current
0001 /dev/dsk/c0t6d0    0024 current  /dev/dsk/c0t5d0    0024 current
0002 /dev/dsk/c0t6d0    0025 current  /dev/dsk/c0t5d0    0025 current
0003 /dev/dsk/c0t6d0    0026 current  /dev/dsk/c0t5d0    0026 current
0004 /dev/dsk/c0t6d0    0027 current  /dev/dsk/c0t5d0    0027 current
0005 /dev/dsk/c0t6d0    0028 current  /dev/dsk/c0t5d0    0028 current
0006 /dev/dsk/c0t6d0    0029 current  /dev/dsk/c0t5d0    0029 current
0007 /dev/dsk/c0t6d0    0030 current  /dev/dsk/c0t5d0    0030 current
0008 /dev/dsk/c0t6d0    0031 current  /dev/dsk/c0t5d0    0031 current
0009 /dev/dsk/c0t6d0    0032 current  /dev/dsk/c0t5d0    0032 current
0010 /dev/dsk/c0t6d0    0033 current  /dev/dsk/c0t5d0    0033 current
0011 /dev/dsk/c0t6d0    0034 current  /dev/dsk/c0t5d0    0034 current
0012 /dev/dsk/c0t6d0    0035 current  /dev/dsk/c0t5d0    0035 current
0013 /dev/dsk/c0t6d0    0036 current  /dev/dsk/c0t5d0    0036 current
                  .
                  .
                  .
0111 /dev/dsk/c0t6d0    0134 current  /dev/dsk/c0t5d0    0134 current
0112 /dev/dsk/c0t6d0    0135 current  /dev/dsk/c0t5d0    0135 current
0113 /dev/dsk/c0t6d0    0136 current  /dev/dsk/c0t5d0    0136 current
0114 /dev/dsk/c0t6d0    0137 current  /dev/dsk/c0t5d0    0137 current
0115 /dev/dsk/c0t6d0    0138 current  /dev/dsk/c0t5d0    0138 current
0116 /dev/dsk/c0t6d0    0139 current  /dev/dsk/c0t5d0    0139 current
0117 /dev/dsk/c0t6d0    0140 current  /dev/dsk/c0t5d0    0140 current
0118 /dev/dsk/c0t6d0    0141 current  /dev/dsk/c0t5d0    0141 current
0119 /dev/dsk/c0t6d0    0142 current  /dev/dsk/c0t5d0    0142 current
0120 /dev/dsk/c0t6d0    0143 current  /dev/dsk/c0t5d0    0143 current
0121 /dev/dsk/c0t6d0    0144 current  /dev/dsk/c0t5d0    0144 current
0122 /dev/dsk/c0t6d0    0145 current  /dev/dsk/c0t5d0    0145 current
0123 /dev/dsk/c0t6d0    0146 current  /dev/dsk/c0t5d0    0146 current
0124 /dev/dsk/c0t6d0    0147 current  /dev/dsk/c0t5d0    0147 current
```

F1 - ll /dev/vg00

Next, view **/dev/vg00** to have a record of the logical volumes.

```
# ll /dev/vg00

/dev/vg00:
total 0
crw-r--r--   1 root     sys       64 0x000000 May 29 04:44 group
brw-r-----   1 root     sys       64 0x000001 May 29 04:44 lvol1
brw-r-----   1 root     sys       64 0x000002 Jul  9 17:10 lvol2
brw-r-----   1 root     sys       64 0x000003 May 29 04:44 lvol3
brw-r-----   1 root     sys       64 0x000004 May 29 04:44 lvol4
brw-r-----   1 root     sys       64 0x000005 May 29 04:44 lvol5
brw-r-----   1 root     sys       64 0x000006 May 29 04:44 lvol6
brw-r-----   1 root     sys       64 0x000007 May 29 04:44 lvol7
crw-r-----   1 root     sys       64 0x000001 May 29 04:44 rlvol1
crw-r-----   1 root     sys       64 0x000002 Jul  9 17:10 rlvol2
crw-r-----   1 root     sys       64 0x000003 May 29 04:44 rlvol3
crw-r-----   1 root     sys       64 0x000004 May 29 04:44 rlvol4
```

```
crw-r-----   1 root     sys       64 0x000005 May 29 04:44 rlvol5
crw-r-----   1 root     sys       64 0x000006 May 29 04:44 rlvol6
crw-r-----   1 root     sys       64 0x000007 May 29 04:44 rlvol7
```

F1 - ll /dev/vg_nw

Next, view **/dev/vg_nw** and any other volume groups.

```
# ll /dev/vg_nw

/dev/vg_nw:
total 0
crw-rw-rw-   1 root     sys       64 0x010000 Jul  9 12:03 group
brw-r-----   1 root     sys       64 0x010003 Jul  9 13:01 lv_nwbackup
brw-r-----   1 root     sys       64 0x010004 Jul  9 13:01 lv_nwlog
brw-r-----   1 root     sys       64 0x010002 Jul  9 12:54 lv_nwsys
brw-r-----   1 root     sys       64 0x010001 Jul  9 12:53 lv_nwtext
crw-r-----   1 root     sys       64 0x010003 Jul  9 13:01 rlv_nwbackup
crw-r-----   1 root     sys       64 0x010004 Jul  9 13:01 rlv_nwlog
crw-r-----   1 root     sys       64 0x010002 Jul  9 12:55 rlv_nwsys
crw-r-----   1 root     sys       64 0x010001 Jul  9 12:54 rlv_nwtext
```

F1 - bdf

Next, view the file systems with **bdf**. Notice that **lvol2** is not shown because this is a swap and dump device.

```
# bdf
Filesystem               kbytes    used   avail %used Mounted on
/dev/vg00/lvol1           91669   31889   50613   39% /
/dev/vg00/lvol7          159509   83630   59928   58% /var
/dev/vg00/lvol6          319125  197912   89300   69%
/usr /dev/vg00/lvol5      31829   11323   17323   40% /tmp
/dev/vg00/lvol4          251285   67854  158302   30% /opt
/dev/vg_nw/lv_nwtext    4099465 2070905 1618613   56% /nwtext
/dev/vg_nw/lv_nwsys     4099465 1063909 2625609   29% /nwsys
/dev/vg_nw/lv_nwlog       99669   17313   72389   19% /nwlog
/dev/vg_nw/lv_nwbackup  2552537  377388 1919895   16% /nwbackup
/dev/vg00/lvol3           19861    2191   15683   12% /home
```

F1 - swapinfo

Next, run **swapinfo** to see that **lvol2** is the only swap device.

```
# swapinfo
             Kb       Kb      Kb   PCT  START/      Kb
TYPE      AVAIL    USED    FREE  USED   LIMIT RESERVE  PRI  NAME
dev      512000       0  512000    0%       0       -    1  /dev/vg00/lvol2
reserve       -  512000 -512000
memory  1670828 1474704  196124   88%
```

F1 - lvlnboot

Next, look at the boot information with **lvlnboot. lvol2** on **vg00** is the dump device.

```
 # lvlnboot -v /dev/vg00
Boot Definitions for Volume Group /dev/vg00:
Physical Volumes belonging in Root Volume Group:
         /dev/dsk/c0t6d0 (10/0.6.0) -- Boot Disk
         /dev/dsk/c0t5d0 (10/0.5.0) -- Boot Disk
Root: lvol1        on: /dev/dsk/c0t6d0
                       /dev/dsk/c0t5d0
Swap: lvol2        on: /dev/dsk/c0t6d0
                       /dev/dsk/c0t5d0
Dump: lvol2        on: /dev/dsk/c0t6d0, 0
```

F1 - lifls

Look at the boot area with **lifls**.

```
#lifls -Clv /dev/dsk/c0t6d0

volume ISL10 data size 7984 directory size 8 94/11/04 15:46:53
filename   type   start   size     implement  created
================================================================
ODE        -12960 584     496      0          95/05/19 13:36:50
MAPFILE    -12277 1080    32       0          95/05/19 13:36:50
SYSLIB     -12280 1112    224      0          95/05/19 13:36:50
CONFIGDATA -12278 1336    62       0          95/05/19 13:36:50
SLMOD      -12276 1400    70       0          95/05/19 13:36:50
SLDEV      -12276 1472    68       0          95/05/19 13:36:50
SLDRIVERS  -12276 1544    244      0          95/05/19 13:36:50
MAPPER     -12279 1792    93       0          95/05/19 13:36:51
IOTEST     -12279 1888    150      0          95/05/19 13:36:51
PERFVER    -12279 2040    80       0          95/05/19 13:36:51
PVCU       -12801 2120    64       0          95/05/19 13:36:51
SSINFO     -12286 2184    1        0          96/09/16 09:04:01
ISL        -12800 2192    240      0          94/11/04 15:46:53
AUTO       -12289 2432    1        0          94/11/04 15:46:53
HPUX       -12928 2440    800      0          94/11/04 15:46:54
LABEL      BIN    3240    8        0          96/05/29 01:49:55
```

F2

After all the appropriate information has been saved for the existing configuration, we can begin the reconfiguration. First, we break the mirror with **lvreduce** and the **-m** option.

```
# lvreduce -m 0 /dev/vg00/lvol1
Logical volume "/dev/vg00/lvol1" has been successfully reduced.
Volume Group configuration for /dev/vg00 has been saved in /etc/lvmconf/vg00.conf
# lvreduce -m 0 /dev/vg00/lvol2
Logical volume "/dev/vg00/lvol2" has been successfully reduced.
Volume Group configuration for /dev/vg00 has been saved in /etc/lvmconf/vg00.conf
# lvreduce -m 0 /dev/vg00/lvol3
Logical volume "/dev/vg00/lvol3" has been successfully reduced.
Volume Group configuration for /dev/vg00 has been saved in /etc/lvmconf/vg00.conf
# lvreduce -m 0 /dev/vg00/lvol4
Logical volume "/dev/vg00/lvol4" has been successfully reduced.
Volume Group configuration for /dev/vg00 has been saved in /etc/lvmconf/vg00.conf
# lvreduce -m 0 /dev/vg00/lvol5
Logical volume "/dev/vg00/lvol5" has been successfully reduced.
Volume Group configuration for /dev/vg00 has been saved in /etc/lvmconf/vg00.conf
# lvreduce -m 0 /dev/vg00/lvol6
Logical volume "/dev/vg00/lvol6" has been successfully reduced.
Volume Group configuration for /dev/vg00 has been saved in /etc/lvmconf/vg00.conf
# lvreduce -m 0 /dev/vg00/lvol7
Logical volume "/dev/vg00/lvol7" has been successfully reduced.
Volume Group configuration for /dev/vg00 has been saved in /etc/lvmconf/vg00.conf
```

You can type each command or make a file with the **lvreduce** commands in it and run the file. You can call the file **/tmp/reduce** with the following entries:

```
lvreduce -m 0 /dev/vg00/lvol1
lvreduce -m 0 /dev/vg00/lvol2
lvreduce -m 0 /dev/vg00/lvol3
lvreduce -m 0 /dev/vg00/lvol4
lvreduce -m 0 /dev/vg00/lvol5
lvreduce -m 0 /dev/vg00/lvol6
lvreduce -m 0 /dev/vg00/lvol7
```

After you create this file, change it to executable and then run with the following two commands.

```
# chmod 555 /tmp/reduce
# /tmp/reduce
```

You will then see all the output of having run the **lvreduce** commands.

F3

Check to see that mirroring of **lvol1-7** has been reduced with **lvdisplay**. Look to see that mirrored copies are equal to 0. Only **lvol1** through **lvol3** are shown in this listing.

```
# lvdisplay -v /dev/vg00/lvol* | more

--- Logical volumes ---
LV Name                /dev/vg00/lvol1
VG Name                /dev/vg00
LV Permission          read/write
LV Status              available/syncd
Mirror copies          0
Consistency Recovery   MWC
Schedule               parallel
LV Size (Mbytes)       92
Current LE             23
Allocated PE           23
Stripes                0
Stripe Size (Kbytes)   0
Bad block              off
Allocation             strict/contiguous

LV Name                /dev/vg00/lvol2
VG Name                /dev/vg00
LV Permission          read/write
LV Status              available/syncd
Mirror copies          0
Consistency Recovery   MWC
Schedule               parallel
LV Size (Mbytes)       500
Current LE             125
Allocated PE           125
Stripes                0
Stripe Size (Kbytes)   0
Bad block              off
Allocation             strict/contiguous

LV Name                /dev/vg00/lvol3
VG Name                /dev/vg00
LV Permission          read/write
LV Status              available/syncd
Mirror copies          0
Consistency Recovery   MWC
Schedule               parallel
LV Size (Mbytes)       20
Current LE             5
Allocated PE           5
Stripes                0
Stripe Size (Kbytes)   0
Bad block              on
Allocation             strict
```

F4

Now remove **c0t5d0** from **vg00** with **vgreduce**. Since there is no mirroring in place, this approach will work. This disk will be put on a different SCSI controller and again used for mirroring later in the procedure.

```
# vgreduce /dev/vg00 /dev/dsk/c0t5d0
Volume group "/dev/vg00" has been successfully reduced.
Volume Group configuration for /dev/vg00 has been saved in /etc/lvmconf/vg00.conf
```

F5

At this point **c0t5d0** is no longer in **vg00**. Verify that "PV Name" **c0t5d0** is no longer in **vg00** with **vgdisplay**.

```
# vgdisplay -v
```

There should be no **c0t5d0** in **vg00**.

F6

Verify that "dump lvol" is in **/stand/system**. If not, add "dump vol" and reconfigure the kernel. See kernel the rebuild procedure in Chapter 1.

F7

Now the hardware upgrade takes place. The system is shut down, disk drives are added and moved, and the system is rebooted. The 4 GByte disk **/dev/dsk/c0t5d0** becomes **/dev/dsk/c1t3d0** at address 10/4/4.3.0, and a new 2 GByte disk is introduced as 10/0.5.0 with the device name **/dev/dsk/c0t5d0**. The second half of Figure 3-12 (Disk Reconfiguration Diagram) depicts this change.

F8

The first activity to perform after the hardware upgrade is to view the new disks with **ioscan**. There is now a 2 GByte disk at 10/0.5.0 and a 4 GByte disk at 10/4/4.3.0.

```
# ioscan -funC disk
Class     I  H/W Path     Driver      S/W State H/W Type  Description
===================================================================
disk      2  10/0.5.0   sdisk      CLAIMED   DEVICE SEAGATE    ST32550W
                         /dev/dsk/c0t5d0   /dev/rdsk/c0t5d0
disk      3  10/0.6.0     sdisk       CLAIMED   DEVICE    SEAGATE ST15150W
                         /dev/dsk/c0t6d0   /dev/rdsk/c0t6d0
disk     12  10/4/4.3.0   disc3       CLAIMED   DEVICE    SEAGATE ST15150W
                         /dev/dsk/c1t3d0       /dev/rdsk/c1t3d0
                         /dev/floppy/c1t3d0    /dev/rfloppy/c1t3d0
disk      6  10/4/4.4.0   disc3       CLAIMED   DEVICE    SEAGATE ST15150W
                         /dev/dsk/c1t4d0      /dev/rdsk/c1t4d0
                         /dev/floppy/c1t4d0    /dev/rfloppy/c1t4d0
disk      7  10/4/4.5.0   disc3       CLAIMED   DEVICE    SEAGATE ST15150W
                         /dev/dsk/c1t5d0       /dev/rdsk/c1t5d0
                         /dev/floppy/c1t5d0    /dev/rfloppy/c1t5d0
disk      8  10/4/4.6.0   disc3       CLAIMED   DEVICE    SEAGATE ST15150W
                         /dev/dsk/c1t6d0       /dev/rdsk/c1t6d0
                         /dev/floppy/c1t6d0    /dev/rfloppy/c1t6d0
disk      9  10/4/12.4.0  disc3       CLAIMED   DEVICE    SEAGATE ST15150W
                         /dev/dsk/c2t4d0       /dev/rdsk/c2t4d0
                         /dev/floppy/c2t4d0    /dev/rfloppy/c2t4d0
disk     10  10/4/12.5.0  disc3       CLAIMED   DEVICE    SEAGATE ST15150W
                         /dev/dsk/c2t5d0       /dev/rdsk/c2t5d0
                         /dev/floppy/c2t5d0    /dev/rfloppy/c2t5d0
disk     11  10/4/12.6.0  disc3       CLAIMED   DEVICE    SEAGATE ST15150W
                         /dev/dsk/c2t6d0       /dev/rdsk/c2t6d0
                         /dev/floppy/c2t6d0    /dev/rfloppy/c2t6d0
disk      5  10/12/5.2.0  sdisk       CLAIMED   DEVICE    TOSHIBA CD-ROM XM-5401TA
                         /dev/dsk/c3t2d0   /dev/rdsk/c3t2d0
```

F9

Now we run **vgdisplay** to see new volume group information. Only **c0t6d0** is in **vg00** and no mirroring is yet configured. The other volume groups have remained the same. Only **lvol1** through **lvol3** are shown in our example.

```
# vgdisplay -v /dev/vg00

--- Volume groups ---
VG Name                 /dev/vg00
VG Write Access         read/write
VG Status               available
Max LV                  255
Cur LV                  7
Open LV                 7
Max PV                  16
Cur PV                  1
Act PV                  1
```

```
Max PE per PV          1023
VGDA                   2
PE Size (Mbytes)       4
Total PE               1023
Alloc PE               344
Free PE                679
Total PVG              0

--- Logical volumes ---
LV Name                /dev/vg00/lvol1
LV Status              available/syncd
LV Size (Mbytes)       92
Current LE             23
Allocated PE           23
Used PV                1

LV Name                /dev/vg00/lvol2
LV Status              available/syncd
LV Size (Mbytes)       500
Current LE             125
Allocated PE           125
Used PV                1

LV Name                /dev/vg00/lvol3
LV Status              available/syncd
LV Size (Mbytes)       20
Current LE             5
Allocated PE           5
Used PV                1
```

(F9 continued)

Only the first three logical volumes in **/dev/vg_nw** are shown.

```
# vgdisplay -v /dev/vg_nw
VG Name                /dev/vg_nw
VG Write Access        read/write
VG Status              available
Max LV                 255
Cur LV                 4
Open LV                4
Max PV                 16
Cur PV                 6
Act PV                 6
Max PE per PV          1023
VGDA                   12
PE Size (Mbytes)       4
Total PE               6138
Alloc PE               5416
Free PE                722
Total PVG              2
   --- Logical volumes ---
LV Name                /dev/vg_nw/lv_nwtext
LV Status              available/syncd
LV Size (Mbytes)       4092
Current LE             1023
Allocated PE           2046
Used PV                2

LV Name                /dev/vg_nw/lv_nwsys
LV Status              available/syncd
LV Size (Mbytes)       4092
Current LE             1023
Allocated PE           2046
Used PV                2
LV Name                /dev/vg_nw/lv_nwbackup
LV Status              available/syncd
LV Size (Mbytes)       2548
Current LE             637
Allocated PE           1274
Used PV                2
```

F10

Use **vgextend** to add the 4 GByte disk to **vg00** for mirroring (you may also have to run **pvcreate** here, too).

```
# vgextend /dev/vg00 /dev/dsk/c1t3d0
Volume group "/dev/vg00" has been successfully extended. Volume Group configuration for
/dev/vg00 has been saved in /etc/lvmconf/vg00.conf
```

F11

Now we can create the new 2 GByte disk and add it to **vg00** using the two following commands: **pvcreate** (F11) to create the physical volume and **vgextend** (F12) to extend the volume group.

```
# pvcreate -f /dev/rdsk/c0t5d0
Physical volume "/dev/rdsk/c0t5d0" has been successfully created.
```

F12

```
# vgextend /dev/vg00 /dev/dsk/c0t5d0
Volume group "/dev/vg00" has been successfully extended.
Volume Group configuration for /dev/vg00 has been saved in /etc/lvmconf/vg00.conf
```

F13

We can check to see that these two disks have indeed been added to **vg00** with **vgdisplay**. Only **lvol1** through **lvol3** are shown in our example. The end of the display is the significant part of the listing showing three physical volumes.

```
# vgdisplay -v /dev/vg00

--- Volume groups ---
VG Name                     /dev/vg00
VG Write Access             read/write
VG Status                   available
Max LV                      255
Cur LV                      7
Open LV                     7
Max PV                      16
Cur PV                      3
Act PV                      3
Max PE per PV               1023
VGDA                        6
PE Size (Mbytes)            4
Total PE                    2554
Alloc PE                    344
Free PE                     2210
Total PVG                   0

--- Logical volumes ---

LV Name                     /dev/vg00/lvol1
LV Status                   available/syncd
LV Size (Mbytes)            92
Current LE                  23
Allocated PE                23
Used PV                     1

LV Name                     /dev/vg00/lvol2
LV Status                   available/syncd
LV Size (Mbytes)            500
Current LE                  125
Allocated PE                125
Used PV                     1

LV Name                     /dev/vg00/lvol3
LV Status                   available/syncd
LV Size (Mbytes)            20
Current LE                  5
```

```
Allocated PE            5
Used PV                 1

                  -
                  -
                  -

--- Physical volumes ---

PV Name                 /dev/dsk/c0t6d0
PV Status               available
Total PE                1023
Free PE                 679

PV Name                 /dev/dsk/c1t3d0
PV Status               available
Total PE                1023
Free PE                 1023

PV Name                 /dev/dsk/c0t5d0
PV Status               available
Total PE                508
Free PE                 508
```

F14

We can now create the dump logical volume in **vg00** with **lvcreate** (F14), extend it to 2 GBytes with **lvextend** (F15), and view it with **lvdisplay** (F16).

```
# lvcreate -n dump  /dev/vg00
Logical volume "/dev/vg00/dump" has been successfully created with character device
"/dev/vg00/rdump".
Volume Group configuration for /dev/vg00 has been saved in /etc/lvmconf/vg00.conf
```

F15

```
# lvextend -l 508 /dev/vg00/dump /dev/dsk/c0t5d0
Logical volume "/dev/vg00/dump" has been successfully extended.
Volume Group configuration for /dev/vg00 has been saved in /etc/lvmconf/vg00.conf
```

F16

```
# lvdisplay /dev/vg00/dump | more

--- Logical volumes ---
LV Name                 /dev/vg00/dump
VG Name                 /dev/vg00
LV Permission           read/write
LV Status               available/syncd
Mirror copies           0
Consistency Recovery    MWC
Schedule                parallel
LV Size (Mbytes)        2032
Current LE              508
Allocated PE            508
Stripes                 0
Stripe Size (Kbytes)    0
Bad block               on
Allocation              strict

--- Distribution of logical volume ---
PV Name            LE on PV  PE on PV
/dev/dsk/c0t5d0    508       508
                   -
                   -
                   -
```

F17

In order to make **/dev/vg00/dump** the dump device, we must first make it contiguous with **lvchange** and then make it a dump device with **lvlnboot**.

```
# lvchange -C y /dev/vg00/dump


# lvlnboot -d /dev/vg00/dump
```

F18

View dump devices.

```
# lvlnboot -v | more
Boot Definitions for Volume Group /dev/vg00:
Physical Volumes belonging in Root Volume Group:
             /dev/dsk/c0t6d0 (10/0.6.0) -- Boot Disk
             /dev/dsk/c1t3d0 (10/4/4.3.0) -- Boot Disk
             /dev/dsk/c0t5d0 (10/0.5.0)
Root: lvol1     on:     /dev/dsk/c0t6d0
Swap: lvol2     on:     /dev/dsk/c0t6d0
Dump: lvol2     on:     /dev/dsk/c0t6d0, 0
Dump: dump      on:     /dev/dsk/c0t5d0, 1
```

This may not be what we want. The primary dump device, as indicated by the "0" is **/dev/dsk/c0t6d0** and the secondary dump device, indicated by the "1," is **/dev/dsk/c0t5d0**. We can, optionally, redo this. Let's proceed with mirroring the lvols on **/dev/vg00** and come back to dump devices.

F19

Let's now extend all the volumes in **vg00** for one mirror using **lvextend**.

```
# lvextend -m 1 /dev/vg00/lvol1 /dev/dsk/c1t3d0
The newly allocated mirrors are now being synchronized.
This operation will take some time. Please wait ....
Logical volume "/dev/vg00/lvol1" has been successfully extended.
Volume Group configuration for /dev/vg00 has been saved in /etc/lvmconf/vg00.conf
```

Put the following in **/tmp/mirror** and run. **lvol1** was extended earlier; **lvol2** is swap and doesn't need to be extended:

```
lvextend -m 1 /dev/vg00/lvol3 /dev/dsk/c1t3d0
lvextend -m 1 /dev/vg00/lvol4 /dev/dsk/c1t3d0
lvextend -m 1 /dev/vg00/lvol5 /dev/dsk/c1t3d0
lvextend -m 1 /dev/vg00/lvol6 /dev/dsk/c1t3d0
lvextend -m 1 /dev/vg00/lvol7 /dev/dsk/c1t3d0


The newly allocated mirrors are now being synchronized.
This operation will take some time.
Please wait .... Logical volume "/dev/vg00/lvol2" has been successfully extended.
Volume Group configuration for /dev/vg00 has been saved in /etc/lvmconf/vg00.conf
                  -
                  -
                  -
```

F20

Let's now verify that the mirroring is in place with **lvdisplay** (only **lvol1** and **lvol2** are shown).

```
# lvdsisplay -v /dev/vg00/lvol* | more

--- Logical volumes ---
LV Name                /dev/vg00/lvol1
VG Name                /dev/vg00
LV Permission          read/write
LV Status              available/syncd
```

```
Mirror copies          1
Consistency Recovery   MWC
Schedule               parallel
LV Size (Mbytes)       92
Current LE             23
Allocated PE           46
Stripes                0
Stripe Size (Kbytes)   0
Bad block              off
Allocation             strict/contiguous

--- Distribution of logical volume ---
PV Name            LE on PV  PE on PV
/dev/dsk/c0t6d0    23        23
/dev/dsk/c1t3d0    23        23

--- Logical extents ---
LE   PV1                PE1  Status 1 PV2                PE2  Status 2
0000 /dev/dsk/c0t6d0    0000 current  /dev/dsk/c1t3d0    0000 current
0001 /dev/dsk/c0t6d0    0001 current  /dev/dsk/c1t3d0    0001 current
0002 /dev/dsk/c0t6d0    0002 current  /dev/dsk/c1t3d0    0002 current
0003 /dev/dsk/c0t6d0    0003 current  /dev/dsk/c1t3d0    0003 current
                  .
                  .
                  .
```

You can see from this listing that **c0t6d0** is mirrored on **c1t3d0**.

F21

Reboot the system to confirm that all changes have taken effect.

After reboot, do the following to create a dump. The key must be in the "Service" position for **^b** to work (you must be on a server and at the system console for this to work.)

Use **^b** to get the **CM>** prompt.

Use the **tc** command at the **CM>** prompt to create core dump

---

F22

The system will automatically reboot after a core dump. Use the following command to save the core dump to tape. The **/var/adm/crash** file name is required even though the core dump is in the dump logical volume and not in the **/var/adm/crash** directory.

```
# savecore -t /dev/rmt/0m /var/adm/crash
```

F23

Then use **savecore -xt** and the directory name to the extract core dump. If you do not have room for the core dump, or you want a more thorough check, you can place a call and ask the HP Response Center to verify the **savecore** to tape has worked.

```
# savecore -xt /dev/rmt/0m /var/adm/crash
```

The core dump space requirement is calculated from the end of dump back toward the front. For this reason about roughly 1.5 GBytes are written to the **dump** logical volume and then roughly 600 MBytes are written to **lvol2**.

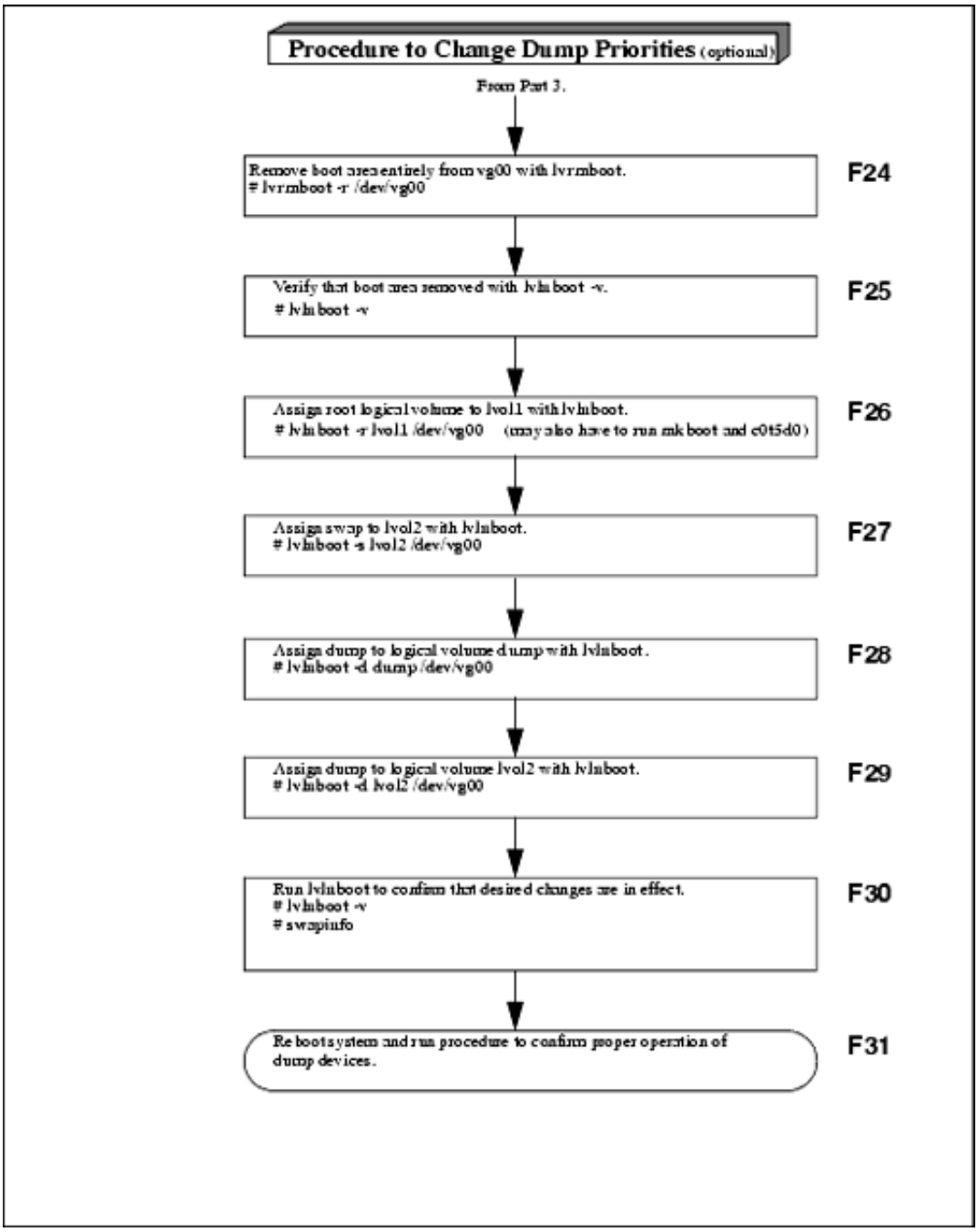


Optional Procedure to Exchange Dump Priorities

This procedure removes all boot definitions, including swap and dump, from **/dev/vg00** with **lvrmboot** and recreates them with **lvlnboot**. This needs to be done because **lvol2** is the primary dump logical volume (0) and dump is the secondary dump logical volume (1).

You must reboot in order for these changes to take effect. Figure 3-15 shows the steps required to complete this optional procedure.

Figure 3-15. Procedure to Change Dump Priorities (Optional)



F24

Remove the boot area entirely from **vg00** with **lvrmboot**.

```
# lvrmboot -r /dev/vg00
```

F25

Verify that the boot area was removed with lvlnboot.

```
# lvlnboot -v
Boot Definitions for Volume Group /dev/vg00: The Boot Data Area is empty.
```

F26

Assign the root logical volume to **lvol1** on **/dev/vg00** with **lvlnboot**.

```
# lvlnboot -r lvol1 /dev/vg00
```

F27

Assign the swap to **lvol2** on **/dev/vg00**.

```
# lvlnboot -s lvol2 /dev/vg00

# swapinfo
             Kb       Kb      Kb    PCT    START/      Kb
TYPE       AVAIL    USED    FREE  USED   LIMIT RESERVE  PRI  NAME
dev       512000       0  512000    0%       0       -    1  /dev/vg00/lvol2
reserve        -   23144  -23144
memory   1671008   27324 1643684    2%
```

F28

Assign the dump to the logical volume **dump** on **/dev/vg00**.

```
# lvlnboot -d dump /dev/vg00
```

F29

Assign the secondary dump device as **lvol2** (primary swap) on **lvol2**.

```
# lvlnboot -d lvol2 /dev/vg00
```

F30

Run **lvlnboot** to confirm that the dump and swap are properly configured with priority "0" on 2 GByte disk **c0t5d0** and "1" on **c0t6d0**.

```
# lvlnboot -v     # after adding lvol2 as secondary dump

Boot Definitions for Volume Group /dev/vg00:
Physical Volumes belonging in Root Volume Group:
                  /dev/dsk/c0t6d0 (10/0.6.0) -- Boot Disk
                  /dev/dsk/c1t3d0 (10/4/4.3.0) -- Boot Disk
                  /dev/dsk/c0t5d0 (10/0.5.0)
Root: lvol1       on:      /dev/dsk/c0t6d0
                           /dev/dsk/c1t3d0
Swap: lvol2       on:      /dev/dsk/c0t6d0
                           /dev/dsk/c1t3d0
Dump: dump        on:      /dev/dsk/c0t5d0, 0
Dump: dump        on:      /dev/dsk/c0t6d0, 1
```

F31

Reboot the system and run steps F21-F23 to confirm proper operation of dump devices.

Although this procedure to reconfigure disks is for a specific system, it is useful for illustrating the many LVM commands required to perform such tasks. LVM, and disk management in general, are the areas I find consumes the most system administration time in mature HP-UX installations. There are many commands used in this procedure for which there is no way to "back out," so use caution whenever using LVM commands.

HP VERITAS Volume Manager

At the time of this writing, HP Veritas Volume Manager (what I'll call VxVM throughout much of this chapter) is a software product loaded from the HP-UX 11i Application CD-ROM. There is a version of VxVM bundled with 11i called Base HP VERITAS Volume Manager and a full version called HP VERITAS Volume Manager. The Base product is a subset of the full version. With both versions of VxVM there is a Java-based administration interface, striping (RAID 0), concatenation, path failover support, online resizing of volumes, and a task monitor. The full version performs all of the functions in the Base product and also supports load balancing, hot relocation and unrelocation, mirroring of up to 32 copies (RAID 1), mirrored stripes, striped mirrors, RAID 5, online migration, and online relayout. The features of each are described in the *HP VERITAS Volume Manager Release Notes*.

On the system used for the examples compiled for the VxVM part of this chapter the root disk setup at the time 11i was originally loaded on the system was under control of Logical Volume Manager (LVM.) After loading HP Veritas Volume Manager as an application we can then perform storage administration on other disks on the system. The root disk will remain under LVM control and not be placed under VxVM control.

In the upcoming sections, we'll load HP Veritas Volume Manager and perform some basic storage management tasks so you can get a feel for this product. This is a product that has a lot of functionality and manuals devoted to using it, so in this part of the chapter we'll cover some of the basics. Please see docs.hp.com for a complete list of manuals on VxVm. Two that contain much more detailed information on configuring and using VxVM are *HP VERITAS Volume Manager Release Notes* and *HP VERITAS Volume Manager Administrator's Guide*.

HP VERITAS Volume Manager Setup

After loading HP Veritas from the Applications CD-ROM we have to decide what disk(s) we want to control with VxVM. Let's run **ioscan** to view the disks in our L-Class system:

```
# ioscan -funC disk
Class     I  H/W Path     Driver S/W State   H/W Type     Description

==========================================================================

disk      0  0/0/1/1.2.0  sdisk CLAIMED     DEVICE       SEAGATE ST318203LC

                          /dev/dsk/c1t2d0   /dev/rdsk/c1t2d0

disk      1  0/0/2/0.2.0  sdisk CLAIMED     DEVICE       SEAGATE ST318203LC

                          /dev/dsk/c2t2d0   /dev/rdsk/c2t2d0

disk      2  0/0/2/1.4.0  sdisk CLAIMED     DEVICE       TOSHIBA CD-ROM XM-6201TA

                          /dev/dsk/c3t4d0   /dev/rdsk/c3t4d0
```

This output shows two internal disks and a CD-ROM drive (we'll later add two more disks to demonstrate a setup of mirroring and striping). The root disk under LVM control is *c1t2d0*. We want to use VxVM to perform various storage management functions on disk *c2t2d0*. If there is any LVM header information on this disk, it must be removed prior to proceeding with any VxVM functions on the disk. The following two LVM-related commands were issued to create and remove this disk from LVM:

```
# pvcreate -f /dev/rdsk/c2t2d0

Physical volume "/dev/rdsk/c2t2d0" has been successfully created.

# pvremove /dev/rdsk/c2t2d0

The physical volume associated with "/dev/rdsk/c2t2d0" has been
removed.
#
```

You may not have to issue the **pvcreate** command; however, I have found that issuing both commands works every time. This procedure is outlined in the *Release Notes* I mentioned earlier.

Next, we run **vxinstall** to perform the initial setup of VxVM. In the following procedure, we run **vxinstall** and select a *Quick Installation,* which walks us through evaluating the disks on the system and allows us to select those that we want to put under VxVM control:

```
# vxinstall

Populating VxVM DMP device directories ....
Generating list of attached controllers....
Volume Manager Installation

Menu: VolumeManager/Install
The Volume Manager names disks on your system using the controller
and disk number of the disk, substituting them into the following

  pattern:
             c<controller>t<disk>d<disk>

  NOTE:  With the Dynamic Multipathing (DMP) facility of VxVM, the
  controller number represents a multipath pseudo controller number
  for those disk devices with multiple access paths.  For example,
  if a disk has 2 paths from controllers c0 and c1, then the Volume
  Manager displays only one of them, such as c0, to represent both
  of the controllers.


  Some examples would be:

             c0t0d0 - first controller, first target, first disk
             c1t0d0 - second controller, first target, first disk
             c1t1d0 - second controller, second target, first disk
  The Volume Manager has detected the following controllers on your
system:
              c1:
              c2:

Hit RETURN to continue.

_____________________________________________________________________

Volume Manager Installation
Menu: VolumeManager/Install
  You will now be asked if you wish to use Quick Installation or
  Custom Installation.  Custom Installation allows you to select how
  the Volume Manager will handle the installation of each disk
  attached to your system.

  Quick Installation examines each disk attached to your system and
  attempts to create volumes to cover all disk partitions that might
  be used for file systems or for other similar purposes.

  If you do not wish to use some disks with the Volume Manager, or if
  you wish to reinitialize some disks, use the Custom Installation
  option Otherwise, we suggest that you use the Quick Installation
  option.

Hit RETURN to continue.
_____________________________________________________________________
Volume Manager Installation Options
Menu: VolumeManager/Install



 1 Quick Installation
 2 Custom Installation
 ? Display help about menu
 ?? Display help about the menuing system
 q Exit from menus


Select an operation to perform: 1


_____________________________________________________________________
```

```
Volume Manager Quick Installation
Menu: VolumeManager/Install/QuickInstall/c1

Generating list of attached disks on c1....
<excluding c1t2d0>
No disks were found attached to controller c1 !

Hit RETURN to continue.
```

```
Volume Manager Quick Installation
Menu: VolumeManager/Install/QuickInstall/c2

Generating list of attached disks on c2....
  The Volume Manager has detected the following disks on controller
c2:
  c2t2d0
Hit RETURN to continue.
```

```
Volume Manager Quick Installation For Controller c2


Menu: VolumeManager/Install/QuickInstall/c2

Initialize all disks on this controller ? (destroys data on these
disks)
[y,n,q,?] (default: n) y
Are you sure ? (destroys data on these disks)
[y,n,q,?] (default: n) y

 Volume Manager will now initialize all the disks on this controller


Hit RETURN to continue.
```

```
Volume Manager Quick Installation
Menu: VolumeManager/Install/QuickInstall/c2/Init

Use default disk names for these disks? [y,n,q,?] (default: y) y

  The c2t2d0 disk will be given disk name disk01

Hit RETURN to continue.
```

```
Volume Manager Quick Installation
Menu: VolumeManager/Install/QuickInstall

  The following is a summary of your choices.
              c2t2d0 New Disk
Is this correct [y,n,q,?] (default: y)
  The Volume Manager is now reconfiguring (partition phase)...
  Volume Manager: Initializing c2t2d0 as a new disk.
  The Volume Manager is now reconfiguring (initialization phase)...
  Volume Manager: Adding disk01 (c2t2d0) as a new disk.

  The Volume Daemon has been enabled for transactions
  Starting the relocation daemon, vxrelocd.
#
```

Notice that **vxinstall** found the one disk on our system not under LVM control and asked us if we wanted to initialize this disk. Since we have only one potential disk to place under VxVM control, it is the only disk found by **vxinstall**. The *Quick Installation* we chose, as opposed to *Custom Installation*, makes some of the

decisions for us, and in this case, helped us configure the disk quickly.

After running **vxinstall**, we can view the processes that have been started to support VxVM:

```
# ps -ef | grep vx
 root    34     0  0 12:52:28 ?         0:01 vxfsd
 root  2978     0  0 12:58:59 ?         0:00 vxiod
 root  4170  4156  0 13:07:35 ttyp4     0:00 vxnotify
 root  4079     1  0 13:05:51 ?         0:00 vxconfigd -k -m enable
 root  4165     1  0 13:07:35 ttyp4     0:00 /sbin/sh -
                                  /usr/lib/vxvm/bin/vxrelocd root
 root  4173  4165  0 13:07:35 ttyp4     0:00 vxnotify -f -w 15
 root  4628  2900  1 13:16:12 ttyp4     0:00 grep vx
#
```

The initial load and setup of VxVM is quick and easy. I suggest that you obtain documents from docs.hp.com if you don't have them in hardcopy in case you need to refer to them as part of the setup. Without the *Release Notes* I would have not known the procedure to free the second disk from LVM control so I could place it under VxVM control.

Volume Manager Storage Administrator

After the setup of HP Veritas Volume Manager is complete, its graphical interface is invoked with **/opt/HPvmsa/bin/vmsa** for Volume Manager Storage Administrator (I'll call this vmsa occasionally in this section). Figure 3-16 shows the interface for our L-Class system:

Figure 3-16. Volume Manager Storage Administrator

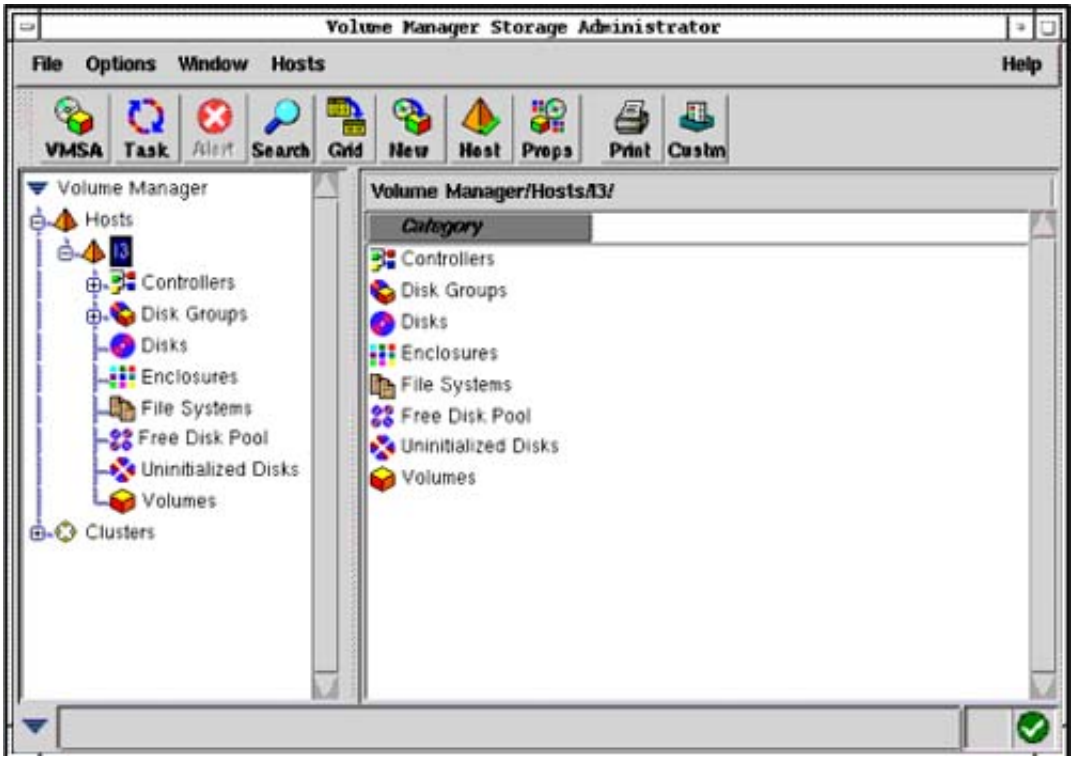


Figure 3-16 shows the many areas of administration that can be performed in this interface under our system name *I3*. We could also perform storage administration on additional systems, *Hosts*, and *Clusters*, which would be shown in the left-hand window along with system *I3* which is now shown.

Let's now select some of the administration icons and see what is reported for our L-Class system. Figure 3-17 shows the *Controllers* on our system:

Figure 3-17. Controllers

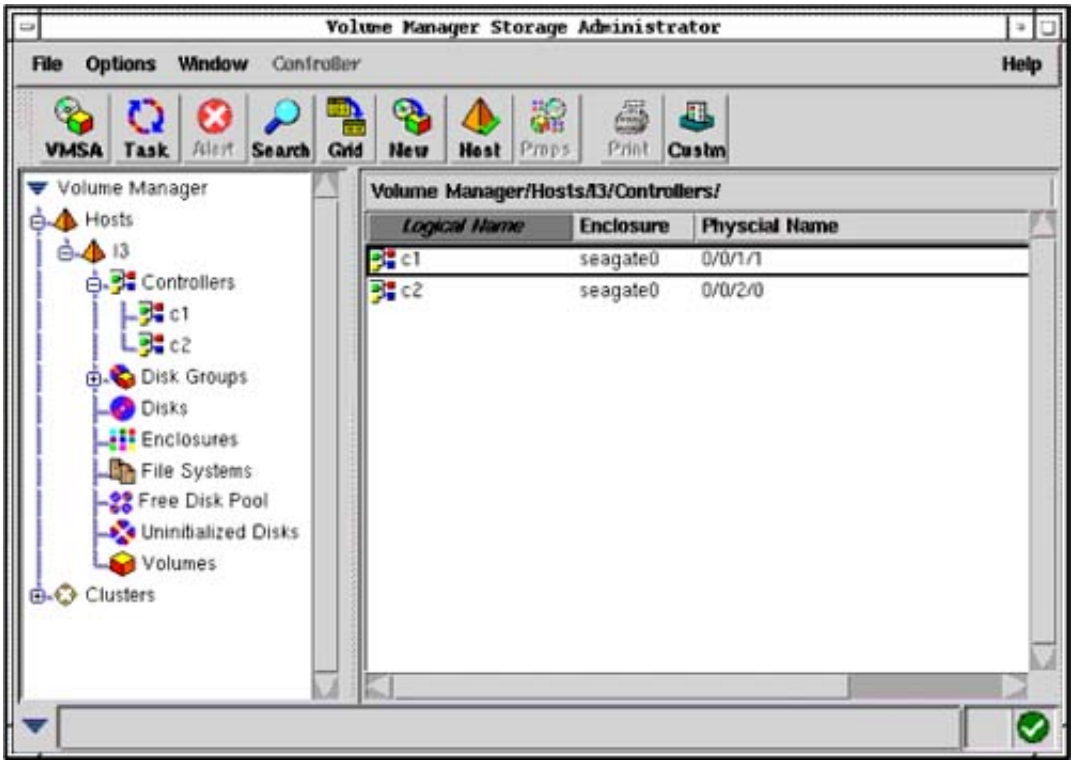


Figure 3-17 shows two internal controllers on our system, *c1* and *c2*. All L-Class systems have two internal controllers, and in our case, there are no additional controllers. We could also select these controllers individually to see what disks are connected to them if we so desired.

Let's now view the *Disk Groups* on our system as shown in Figure 3-18:

Figure 3-18. Disk Groups

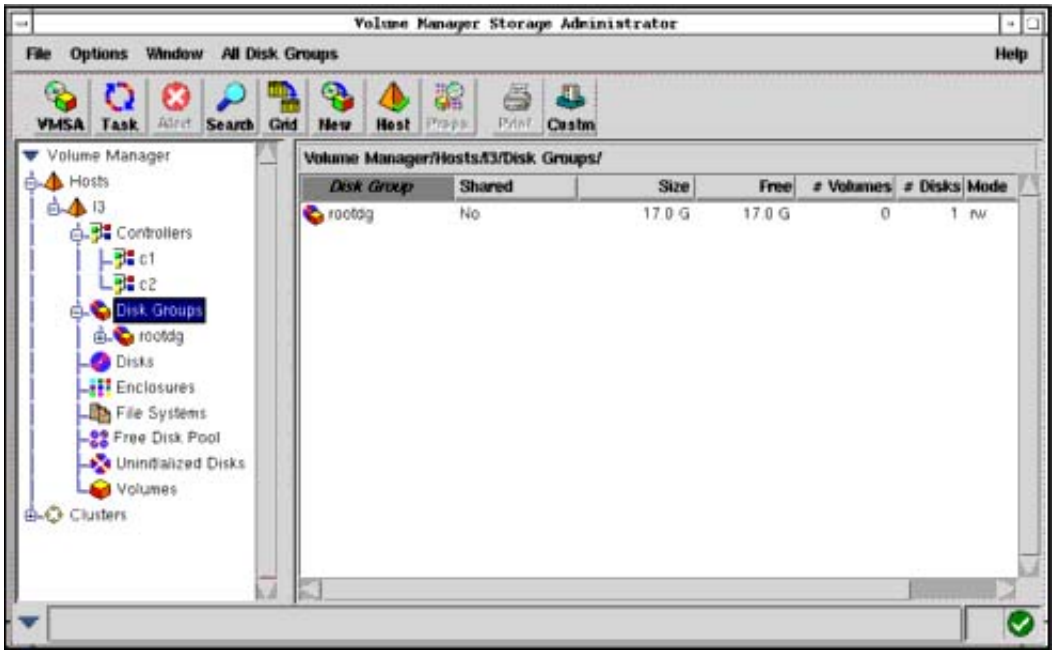


Figure 3-18 shows one disk group, called *rootdg,* which consists of one 17 GByte disk drive. This is the VxVM disk group we set up earlier. Notice that the LVM disk is not shown because it is not under VxVM control.

Next we'll view *Disks* in Figure 3-19:

Figure 3-19. Disks

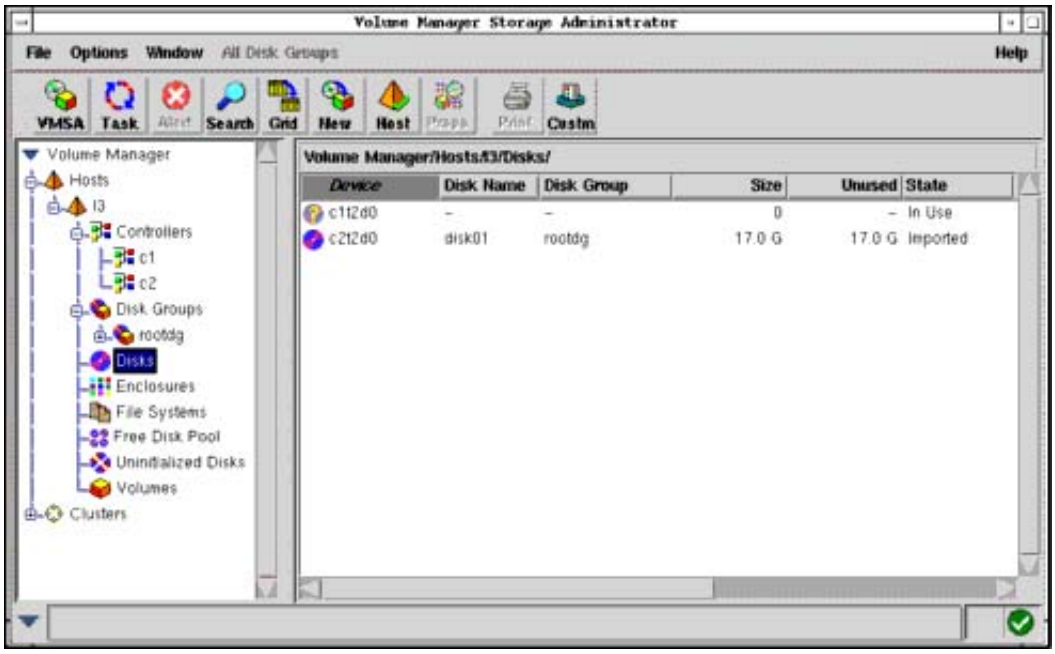


Figure 3-19 shows the two internal disks of our L-Class system. The first is in use by the Logical Volume Manager (root disk) and is not under the control of Veritas Volume Manager. It therefore has very little information associated with it because the VxVM interface does not recognize any of the LVM information. The second is our unused disk in *rootdg* that is called *disk01*.

Figure 3-20 shows the *File Systems* in use on our system:

Figure 3-20. File Systems

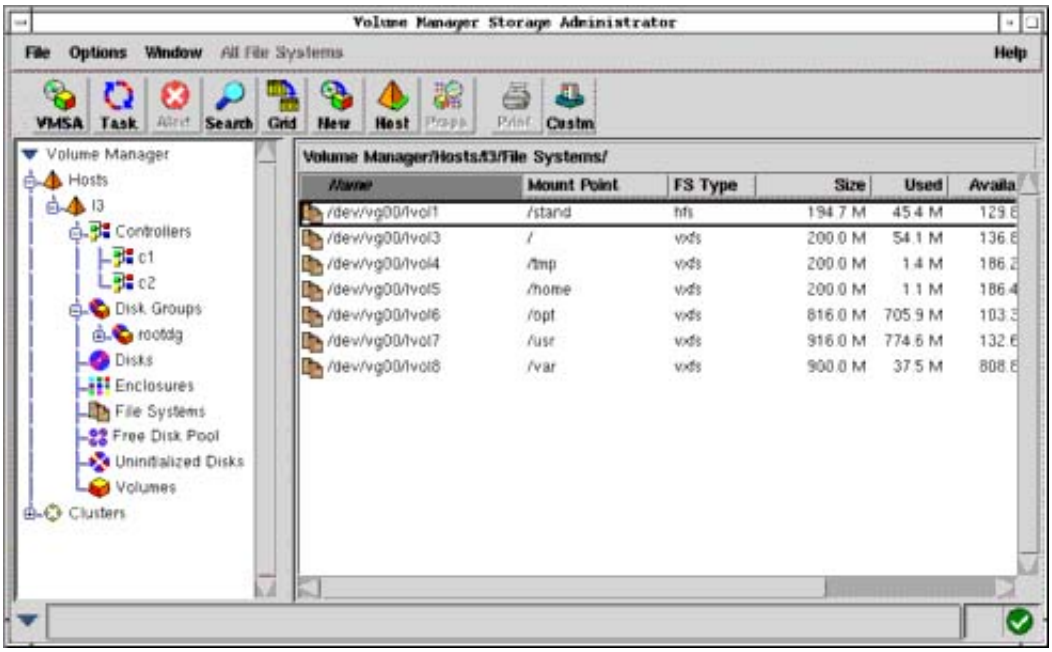


Figure 3-20 shows many filesystems in use. Notice that all but one of these filesystems is a Veritas filesystem as indicated by the *vxfs*. This is sometimes a point of confusion. There is a Veritas file system (*vxfs*) that defines the type of filesystem and there is the Veritas Volume Manager, which we're covering in this section and is used to manage filesystems. You can also manage Veritas filesystems on HP-UX using LVM.

We know that we have one disk in *rootdg* that is unused at this time. Let's issue a couple of **vx** commands and compare these to what we see in the graphical interface. Table 3-1 at the end of this section that describes the use of some **vx** commands.

First, let's get a list of disks on the system with **vxdisk**:

```
# vxdisk list
DEVICE       TYPE      DISK         GROUP        STATUS
c1t2d0       simple    -            -            LVM
c2t2d0       simple    disk01       rootdg       online
#
```

The output of **vxdisk** shows two disks in our system. The first is the root disk, which is under LVM control, and the second is *disk01,* which is under VxVM control.

Next let's get some detailed information with **vxprint -ht**:

```
# vxprint -ht
Disk group: rootdg
DG NAME         NCONFIG      NLOG     MINORS   GROUP-ID
DM NAME         DEVICE       TYPE     PRIVLEN  PUBLEN   STATE
RV NAME         RLINK_CNT    KSTATE   STATE    PRIMARY  DATAVOLS  SRL
RL NAME         RVG          KSTATE   STATE    REM_HOST REM_DG    REM_RLNK
V  NAME         RVG          KSTATE   STATE    LENGTH   USETYPE   PREFPLEX RDPOL
PL NAME         VOLUME       KSTATE   STATE    LENGTH   LAYOUT    NCOL/WID MODE
SD NAME         PLEX         DISK     DISKOFFS LENGTH   [COL/]OFF DEVICE   MODE
SV NAME         PLEX         VOLNAME  NVOLLAYR LENGTH   [COL/]OFF AM/NM    MODE
dg rootdg       default      default  0        969390349.1025.l3
dm disk01       c2t2d0       simple   1024     17782088 -
#
```

Notice that only the information related to *rootdg*, which is under the control of the Veritas Volume Manager, has been produced. The *dg* is information related to the *disk group* and *dm* is information about the *disk mechanism*. In an upcoming **vxprint**, we'll add the *-q* option to eliminate the extensive header information produced with this output.

Next let's see what we have free on *rootdg* with the **vxdg** command:

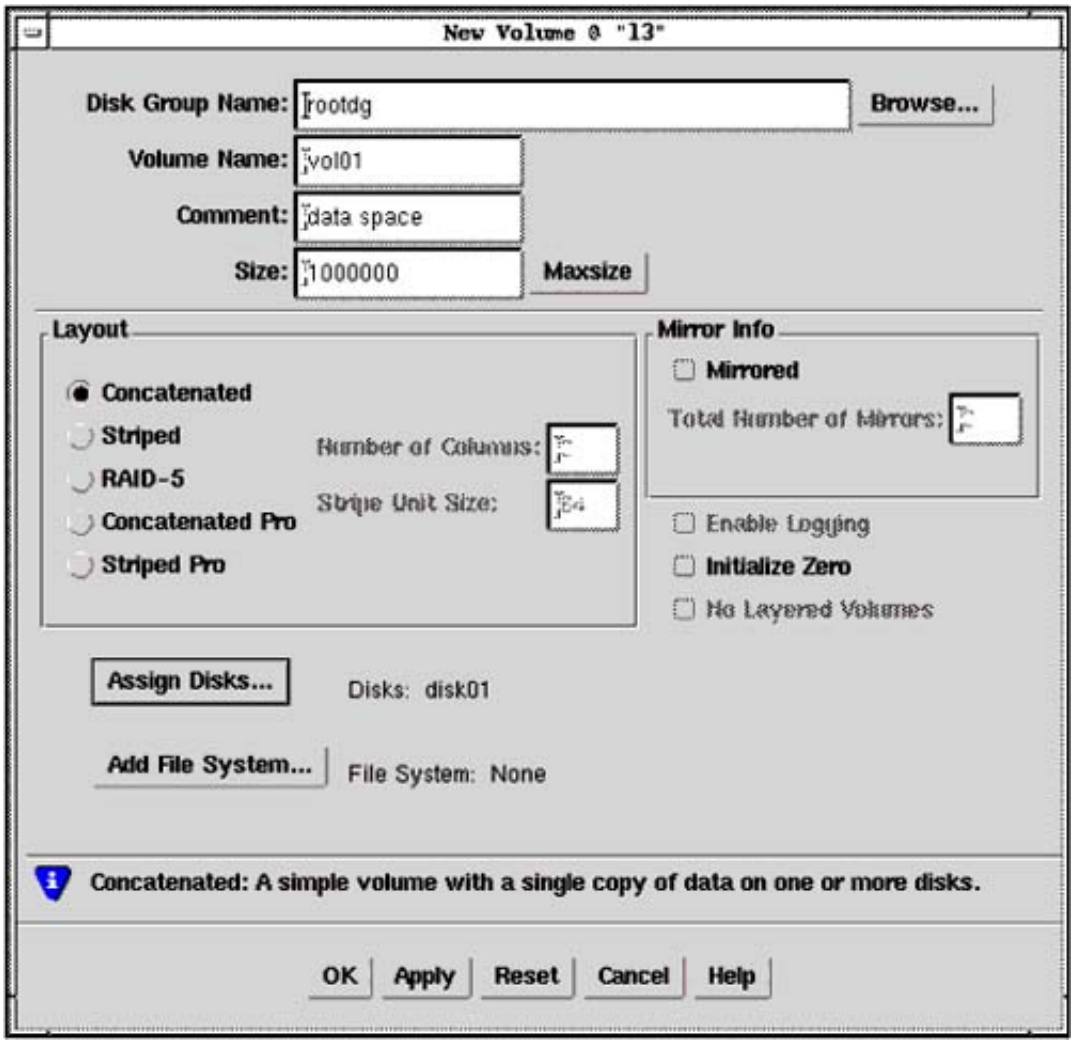
```
# vxdg free
GROUP        DISK         DEVICE       TAG          OFFSET    LENGTH    FLAGS
```

```
rootdg       disk01       c2t2d0       c2t2d0       0          17782088  -
```

This output shows that we have nearly the full 18 GBytes of the disk free at this time.

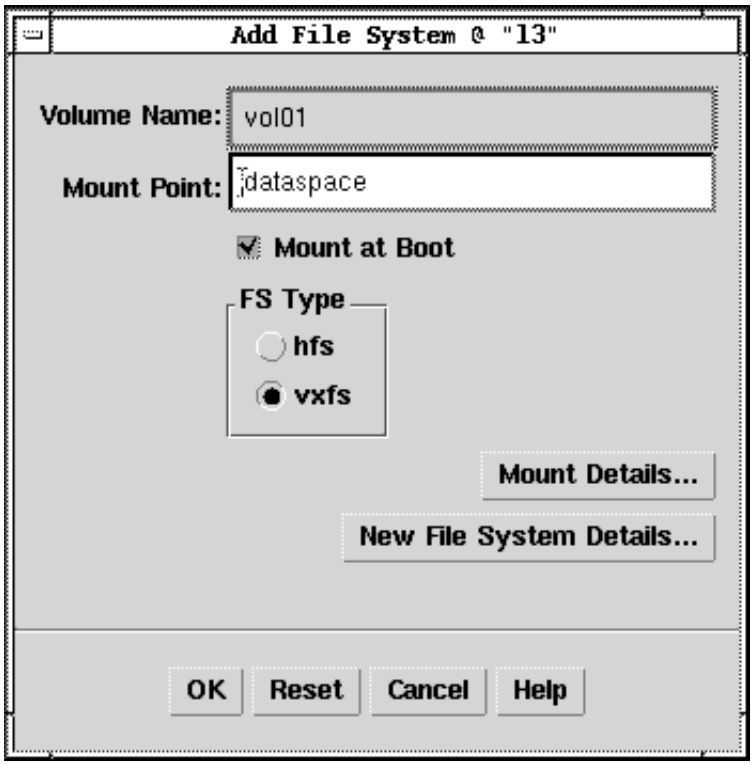
We can now go back to the graphical interface and create a usable volume by selecting *rootdg* and entering information related to the new volume as shown in Figure 3-21:

Figure 3-21. Creating a New Volume



The maximum size we could have made this volume is the total size of the unused disk of which *rootdg* is comprised, which is 17781760 bytes. We have selected about 1 GByte without RAID 5 because another disk would have been required. The default name of *vol01I* is used. In addition, selecting *Add File System...* from the bottom of Figure 3-22 brought up the window in Figure 3-22:

Figure 3-22. Specifying File System Information of a New Volume



In Figure 3-22 we selected a name of */dataspace* as the mount point and have selected *vxfs* as the file systemtype. After clicking *OK,* a new volume is created as shown in Figure 3-23:

Figure 3-23. vol01

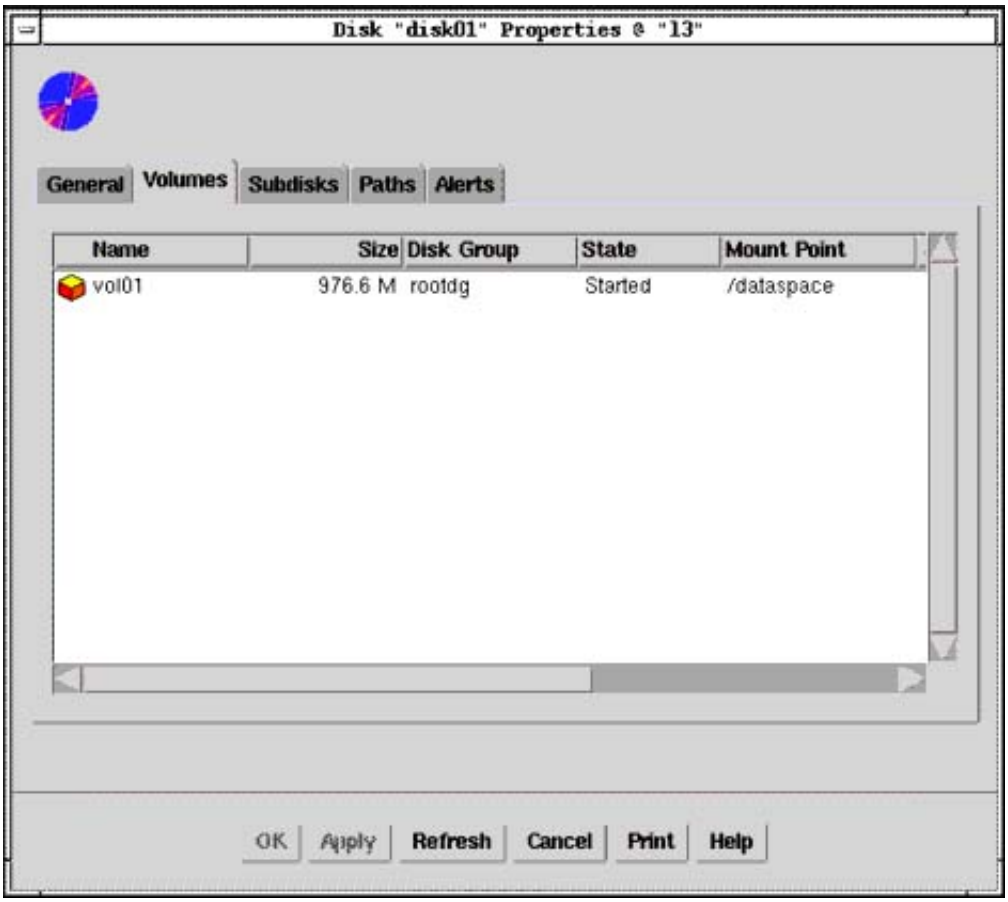


Figure 3-23 shows *vol01* with a mount point of */dataspace*. We can use the **vxprint** command with some useful options to see volume information. This includes information on the volume we just created. The following **vxprint** shows some useful options in the following listing:

```
# vxprint -AGtsq
Disk group: rootdg

dg rootdg       default      default  0        969390349.1025.l3
sd disk01-01    vol01-01     disk01   0        1000000  0         c2t2d0   ENA
#
```

This output shows that our roughly 1 GByte area on *c2t2d0* is in place. We issued this **vxprint** with the *-q* option to eliminate the header information shown in the earlier example.

This simple example demonstrates the ease with which volumes can be added using VxVM. In the next section, we'll add two disks to the system and perform some additional setup.

HP VERITAS Volume Manager Mirroring and Striping

Now that we've covered the basics of VxVM, let's take the next step and add two additional disks to our system and use mirroring and striping.

To begin with let's again use the **vxdisk** command to see the two new disks we've added to the system:

```
# vxdisk list
DEVICE       TYPE      DISK         GROUP        STATUS
c1t0d0       simple    -            -            online invalid
c1t2d0       simple    -            -            LVM
c2t0d0       simple    -            -            online invalid
c2t2d0       simple    disk01       rootdg       online
#
```

At this point, the disks are listed, but they have not been configured in any way. These disks were physically

added to the system, and no additional commands were issued prior to the **vxdisk**.

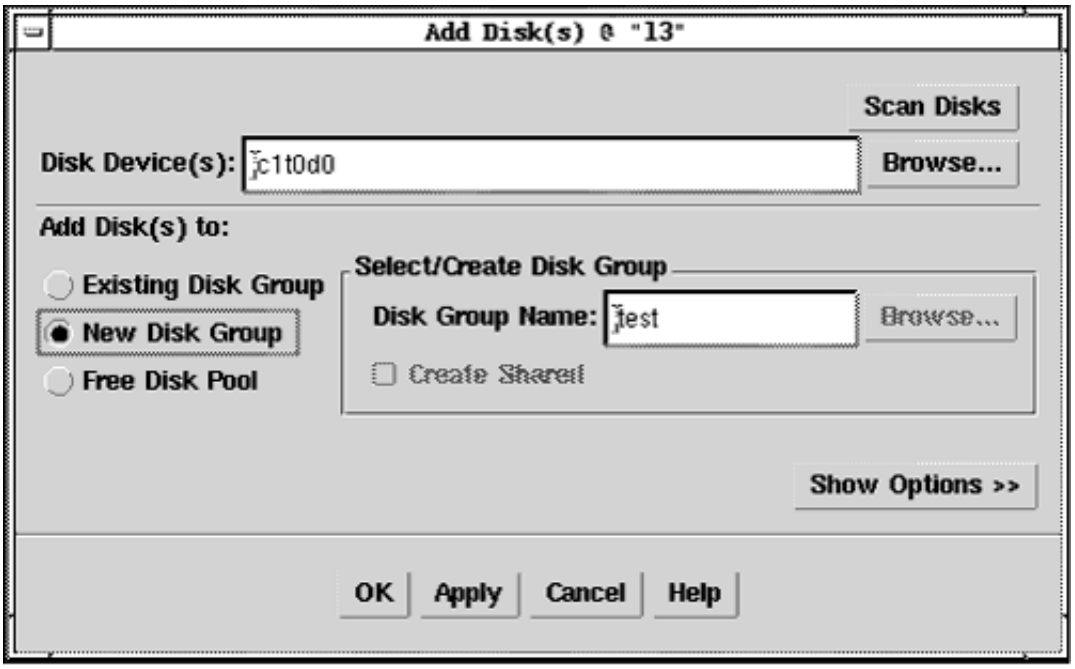
Let's now go to the graphical interface and configure these disks. Figure 3-24 shows a total of four disks, including our two new unconfigured disks:

Figure 3-24. Two New Disks in vmsa



Figure 3-24 shows that disks *c1t0d0* and *c2t0d0* are installed and *Not Initialized*. Let's now add these disks to a new disk group, called *test,* using the graphical interface as shown in Figure 3-25 for the first of the two disks:

Figure 3-25. Adding New Disk to Group *test*



We add both disks to *test* graphically using the *Add Disk(s)* window. This results in the screen shown in Figure 3-26, in which both new disks, *test01* and *test02,* are part of *test*:

Figure 3-26. Two New Disks Added to Group *test*



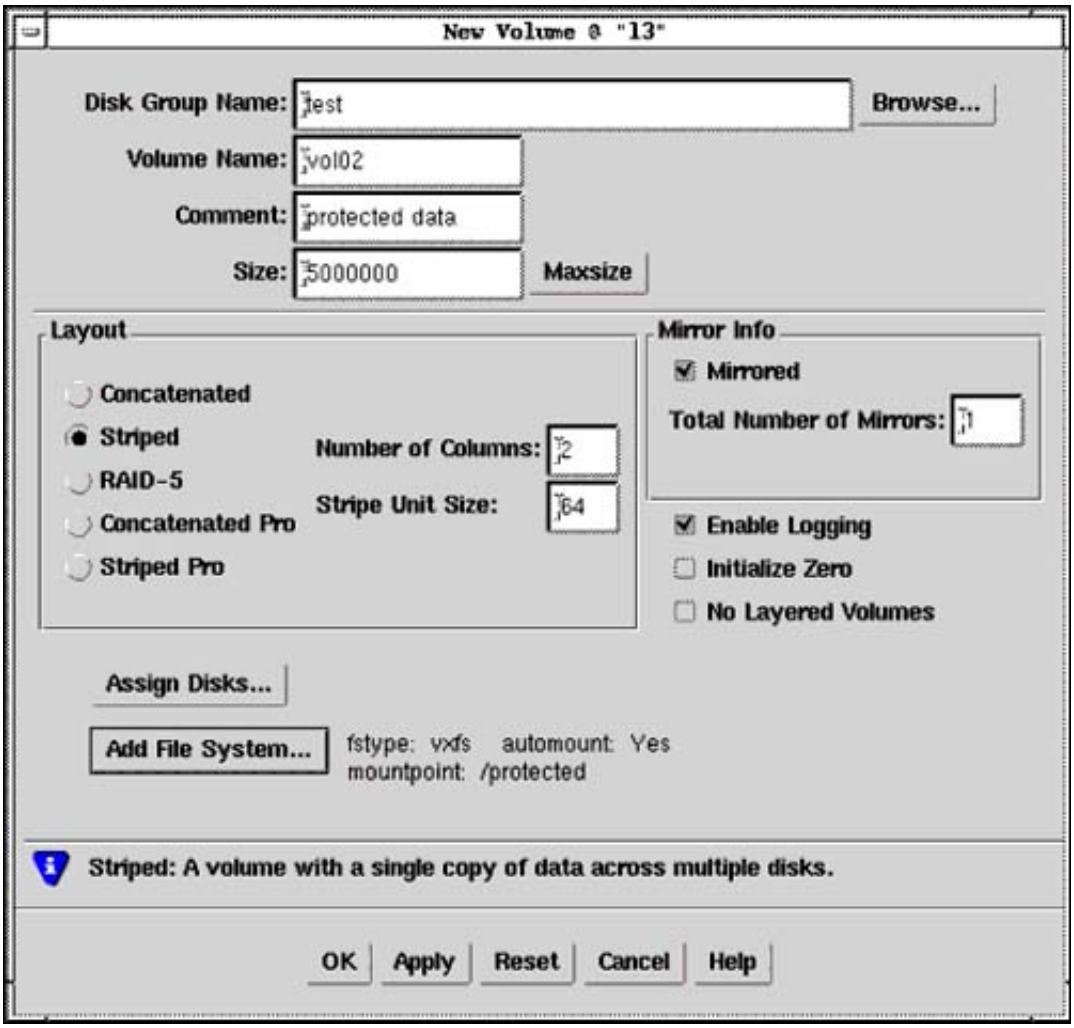
We can confirm the disks in *test* with the **vxdisk** command as shown in the following listing:

```
# vxdisk list
DEVICE       TYPE      DISK         GROUP        STATUS
c1t0d0       simple    test01       test         online
c1t2d0       simple    -            -            LVM
c2t0d0       simple    test02       test         online
c2t2d0       simple    disk01       rootdg       online
#
```

The disks are now shown as configured with names of *test01* and *test02* and both are *online*. Both of these disks are part of *Group test*.

Next, let's create a striped and mirrored volume in our new *disk group test*. Selecting *New* from the items on the *vmsa* window brings up the dialog box shown in Figure 3-27.

Figure 3-27. Creating a New Volume That is Striped and Mirrored



From Figure 3-27, you can see all of the characteristics specified for *vol02*. It is roughly 5 GBytes in size, it is striped, and it has one mirror copy. It is a filesystem type of *vxfs* and has a mount point of **/protected**.

After adding this volume, it appears in the *vmsa* window as shown in Figure 3-28:

Figure 3-28. Volume *vol02*



We could go to the command line to confirm the presence of *vol02* with **vxprint** and no options, as shown in the following listing:

```
# vxprint
Disk group: rootdg

TY NAME         ASSOC        KSTATE   LENGTH   PLOFFS   STATE    TUTIL0  PUTIL0
dg rootdg       rootdg       -        -        -        -        -       -

dm disk01       c2t2d0       -        17782088 -        -        -       -

v  vol01        fsgen        ENABLED  1000000  -        ACTIVE   -       -
pl vol01-01     vol01        ENABLED  1000000  -        ACTIVE   -       -
sd disk01-01    vol01-01     ENABLED  1000000  0        -        -       -

Disk group: test

TY NAME         ASSOC        KSTATE   LENGTH   PLOFFS   STATE    TUTIL0  PUTIL0
dg test         test         -        -        -        -        -       -

dm test01       c1t0d0       -        17782088 -        -        -       -
dm test02       c2t0d0       -        17782088 -        -        -       -

v  vol02        fsgen        ENABLED  5000000  -        ACTIVE   -       -
pl vol02-03     vol02        ENABLED  5000064  -        ACTIVE   -       -
sv vol02-S01    vol02-03     ENABLED  2500032  0        -        -       -
sv vol02-S02    vol02-03     ENABLED  2500032  0        -        -       -

v  vol02-L01    fsgen        ENABLED  2500032  -        ACTIVE   -       -
pl vol02-P01    vol02-L01    ENABLED  LOGONLY  -        ACTIVE   -       -
sd test01-03    vol02-P01    ENABLED  33       LOG      -        -       -
pl vol02-P02    vol02-L01    ENABLED  2500032  -        ACTIVE   -       -
sd test01-04    vol02-P02    ENABLED  2500032  0        -        -       -

v  vol02-L02    fsgen        ENABLED  2500032  -        ACTIVE   -       -
pl vol02-P03    vol02-L02    ENABLED  LOGONLY  -        ACTIVE   -       -
sd test01-05    vol02-P03    ENABLED  33       LOG      -        -       -
pl vol02-P04    vol02-L02    ENABLED  2500032  -        ACTIVE   -       -
sd test02-02    vol02-P04    ENABLED  2500032  0        -        -       -
#
```

This output shows our 5 GByte volume as mirrored.

In addition we want to see **/protected** as a mounted filesystem with **bdf**:

```
# bdf

Filesystem          kbytes    used   avail %used Mounted on
/dev/vg00/lvol3     204800   55509  140002   28% /
/dev/vg00/lvol1     199381   46526  132916   26% /stand
```

```
/dev/vg00/lvol8      921600   38739  827833    4% /var
/dev/vg00/lvol7      937984  793162  135806   85% /usr
/dev/vg00/lvol4      204800    1529  190628    1% /tmp
/dev/vg00/lvol6      835584  722803  105775   87% /opt
/dev/vg00/lvol5      204800    1162  190917    1% /home
/dev/vx/dsk/rootdg/vol01
                    1000000    1349  936243    0% /dataspace
/dev/vx/dsk/test/vol02
                    5000000    2693 4685038    0% /protected
#
```

Our new volume, *vol02,* has been created with a size of 5 GBytes and a mount point of **/protected**.

Several of the basics of using VxVM were covered in the previous examples. This by no means is an exhaustive coverage of VxVM, but hopefully it serves as a good introduction. Please refer to the detailed information in the manuals related to VxVM.

Although we focused mostly on *vmsa* in this section there are a number of **vx** commands that you can issue that perform the same functions as the graphical interface. I sometimes like to issue commands to view volume-related work I have performed in the graphical interface just to confirm the results. Table 3-1 lists some of the most commonly used **vx** commands and their functions.

Table 3-1. VxVM Commands

Command	Description
vxassist	Create and change volumes.
vxdctl	Manage the vxconfigd daemon.
vxdg	Perform tasks related to disk groups.
vxdisk	Perform tasks related to disks.
vxdiskadd	Used to add disks.
vxedit	Change VxVM objects.
vxmake	Create VxVM objects.
vxmend	Correct configuration problems.
vxplex	Perform plex-related tasks.
vxprint	Print configuration information.
vxsd	Perform tasks on subdisks.
vxstat	Print volume statistics.
vxtrace	Trace volume tasks.
vxunrelocate	Move relocated subdisks.
vxvol	Perform volume tasks.

These commands are covered in the HP documentation set, and there are also man pages available which provide detailed usage information on the commands.

Some Additional File System-Related Commands

Viewing File Systems with bdf

You can manually view the file systems you have mounted with the **bdf** command. **bdf** provides the following output:

File system	Block device file system name. In the following example, several logical volumes are shown.
KBytes	Number of KBytes of total disk space on the file system.
used	The number of used KBytes on the file system.
avail	The number of available KBytes on the file system.
%used	The percentage of total available disk space that is used on the file system.
Mounted on	The directory name on which the file system is mounted.
iused	Number of inodes in use (only if you use the *-i* option with **bdf**).
ifree	Number of free inodes (only if you use the *-i* option with **bdf**).
%iuse	Percentage of inodes in use (only if you use the ***-i*** option with **bdf**).

Here is an example of **bdf** that is also in "Logical Volume Manager Background," covered earlier in this chapter:

$ **bdf**

File system	kbytes	used	avail	%used	Mounted on
/dev/vg00/lvol3	47829	18428	24618	43%	/
/dev/vg00/lvol1	67733	24736	36223	41%	/stand
/dev/vg00/lvol8	34541	8673	22413	28%	/var
/dev/vg00/lvol7	299157	149449	119792	56%	/usr
/dev/vg00/lvol4	23013	48	20663	0%	/tmp
/dev/vg00/lvol6	99669	32514	57188	36%	/opt
/dev/vg00/lvol5	19861	9	17865	0%	/home
/dev/dsk/c0t6d0	802212	552120	169870	76%	/mnt/9.x

File System Maintenance with fsck

fsck is a program used for file system maintenance on HP-UX systems. **fsck** checks file system consistency and can make many "life-saving" repairs to a corrupt file system. **fsck** can be run with several options, including the following:

- *F* This option allows you to specify the file system type (see the explanation of file system types in Chapter 15. Be sure to specify a file system type. On some UNIX variants **/etc/fstab** will be used to determine the file system type on others it will not be used. See the **fstab** description later in this section.

- *m* This is a sanity check of the file system. If you run this, you'll be told whether your file system is okay or not. I did the following to check lvol5, which is mounted as **/home**:

```
$ umount /home
$ fsck -m /dev/vg00/lvol5

vxfs fsck: sanity check: /dev/vg00/lvol5 OK
```

- *y* **fsck** will ask questions if run in interactive mode, which is the default. Using the *y* option causes a "yes" response to all questions asked by **fsck**. Don't use this! If you have a serious problem with your file system, data will probably have to be removed, and the *-y* indicates that the response to every question, including removing data, will be "yes".

- *n* The response to all questions asked by **fsck** will be "no." Don't use this, either. If your file system is in bad shape, you may have to respond "yes" to some questions in order to repair the file system. All "no" responses will not do the job.

Since your system runs **fsck** on any file systems that were not marked as clean at the time you shut down the system, you can rest assured that when your system boots, any disks that were not properly shut down will be checked. It is a good idea to run **fsck** interactively on a periodic basis just so you can see firsthand that all of your file systems are in good working order.

Should **fsck** find a problem with a directory or file, it would place these in the **lost+found** directory, which is at the top level of each file system. If a file or directory appears in **lost+found,** you may be able to identify the file or directory by examining it and move it back to its original location. You can use the **file, what**, and **strings** commands on a file to obtain more information about it to help identify its origin.

How are file system problems created? The most common cause of a file system problem is improper shutdown of the system. The information written to file systems is first written to a buffer cache in memory. It is later written to the disk with the **sync** command by unmounting the disk, or through the normal use of filling the buffer and writing it to the disk. If you walk up to a system and shut off the power, you will surely end up with a file system problem. Data in the buffer that was not synchronized to the disk will be lost, the file system will not be marked as properly shut down, and **fsck** will be run when the system boots. A sudden loss of power can also cause an improper system shutdown.

Proper shutdown of the system is described with the **shutdown** command. Although **fsck** is a useful utility that has been known to work miracles on occasion, you don't want to take any unnecessary risks with your file systems. So be sure to properly shut down your system.

The **/etc/fstab** file mentioned earlier is used by **fsck** to determine the sequence of the file system check if it is required at the time of boot. The sequence of entries in **/etc/fstab** is important if a "pass number" for any of the entries does not exist. Here is an example of the **/etc/fstab** file:

```
# System /etc/fstab file. Static information about the file
# systems. See fstab(4) and for further details.

/dev/vg00/lvol3    /            vxfs   delaylog   0    1
/dev/vg00/lvol1    /stand       hfs    defaults   0    1
/dev/vg00/lvol4    /tmp         vxfs   delaylog   0    2
/dev/vg00/lvol6    /opt         vxfs   delaylog   0    2
/dev/vg00/lvol5    /home        vxfs   delaylog   0    2
/dev/vg00/lvol7    /usr         vxfs   delaylog   0    2
/dev/vg00/lvol8    /var         vxfs   delaylog   0    2
/dev/dsk/c0tt6d0   /tmp/mnt9.x  hfs    rw, suid   0    2
      |             |           |         |       |    |
      v              v          v         v       v    v
```

device special file	directory	type	options	backup frequency	pass #

device special file

This is the device block file, such as **/dev/vg00/lvol1** in the example.

directory — The name of the directory under which the device special file is mounted.

type — Can be one of several types including:

	cdfs (local CD-ROM file system)
	hfs (high performance local file system)
	nfs (network file system)
	vxfs (journaled file system)
	swap or
	swapfs
options	Several options are available, including those shown in the example. *rw* is read and write; *ro* is read only.
backup frequency	
	To be used by backup utilities in the future.
pass #	Used by **fsck** to determine the order in which file system checks (**fsck**) will take place.
comment	Anything you want, as long as it's preceded by a #.

Initialize with mediainit

A command you probably won't use, but should be aware of, is **mediainit**.

Here are some of the options of **mediainit**:

-v	This is the verbose option. **mediainit** normally just prints error messages to the screen. You can get continuous feedback on what **mediainit** is doing with the **-v** option.
-i interleave	This allows you to specify the interleave factor, which is the relationship between sequential logical and physical records. **mediainit** will provide this if one is not specified.
-f format	The format option allows you to specify format options for devices, such as floppy disks, that support different format options. This is not required for hard disks.
pathname	This is the character device file to be used for **mediainit**.

newfs, which was used in some of the earlier examples, is used to create a new file system. **newfs** calls the **mksf** command earlier covered. **newfs** builds a file system of the type you specify (this is one of the commands that uses the *-F* option, so you can specify the file system type).

[Team LiB]



Chapter 4. Building an HP-UX Kernel

Introduction

kcweb



Introduction

You may need to modify your HP-UX 11i kernel in some way, such as changing a kernel parameter, and then rebuild your kernel. You may need to create a new HP-UX kernel in order to add device drivers or subsystems, to tune the kernel to get improved performance, to alter configurable parameters, or to change the dump and swap devices. If you update or modify a dynamic element of your kernel, as shown in the example in this chapter, a reboot is not required. Updating or modifying a static element requires a reboot and may also require some additional steps.

With HP-UX 11i it is not necessary to rebuild your kernel for all changes that take place to it. In 11i, there are many *Dynamically Tunable Kernel Parameters* and *Dynamically Loadable Kernel Modules* that will modify your kernel but not require a reboot. Combined with many *Dynamic Patches* that are available in 11i, you will need to reboot your system less often. We'll cover the following two topics in this chapter:

- Manually build an HP-UX kernel - In the next section, we'll modify a *Dynamically Tunable Kernel Parameter*, thereby modifying the kernel, and do not have to reboot the system in order for the change to take place. We'll then make a change to the kernel and fully rebuild it so you can see the process of a complete rebuild, including a reboot. In this chapter, I discuss various commands related to kernel generation and cover the process by which you would manually create a kernel.

- Use **kcweb** to view, modify, and monitor the kernel - **kcweb** is a new Web-based kernel just becoming available at the time of this writing. Many such Web-based tools are planned for HP-UX, so we'll work with **kcweb** to perform many kernel-related functions.

Building a Kernel

New with 11i (first introduced in 11.0) was the introduction of dynamically loadable kernel modules. In 11.x, the infrastructure for this feature was put into place, providing a separate system file for each module. With 11.0 is provided the ability of specially created modules to be loaded or unloaded into the kernel without having to reboot the system as long as the module is not being used. HP-UX 11i continues to support all of this dynamic functionality. This new mechanism provides great flexibility and improved system uptime. Detailed information about this advanced feature can be reviewed in the H*P-UX 11.x Release Notes.* Most of the dynamically loadable kernel modules available at the time of this writing are third party. The *IT Resource Center* Web site (itrc.hp.com) contains information on this topic, including a developer's guide.

To begin, let's take a look at an existing kernel running on an HP-UX 11i L-Class system used in many of the examples throughout this book. The **sysdef** command is used to analyze and report tunable parameters of a currently running system. You can specify a particular file to analyze if you don't wish to use the currently running system. The following is a *partial* listing of having run **sysdef** on an 11i L-Class system:

```
# /usr/sbin/sysdef
NAME                    VALUE       BOOT        MIN-MAX        UNITS   FLAGS
acctresume                  4          -       -100-100                -
acctsuspend                 2          -       -100-100                -
allocate_fs_swapmap         0          -           -                   -
bufpages                32074          -          0-           Pages   -
create_fastlinks            0          -           -                   -
dbc_max_pct                50          -           -                   -
dbc_min_pct                 5          -           -                   -
default_disk_ir             0          -           -                   -
dskless_node                0          -          0-1                  -
eisa_io_estimate          768          -           -                   -
eqmemsize                  23          -           -                   -
file_pad                   10          -          0-                   -
fs_async                    0          -          0-1                  -
hpux_aes_override           0          -           -                   -
maxdsiz                     2          -          0-655360     Pages   -
maxdsiz_64bit           16384          -        256-1048576    Pages   -
maxfiles                   60          -         30-2048               -
maxfiles_lim             1024          -         30-2048               -
maxssiz                 65536          -          0-655360     Pages   -
maxssiz_64bit          262144          -        256-1048576    Pages   -
maxswapchunks             512          -          1-16384              -
```

```
maxtsiz                     2048          -          0-655360     Pages   -
maxtsiz_64bit               2048          -        256-1048576    Pages   -
maxuprc                       75          -          3-                   -
maxvgs                        10          -           -                   -
msgmap                   2555904          -          3-                   -
nbuf                       18720          -          0-                   -
ncallout                     515          -          6-                   -
ncdnode                      150          -           -                   -
ndilbuffers                   30          -          1-                   -
netisr_priority               -1          -         -1-127                -
netmemmax                      0          -           -                   -
nfile                        920          -         14-                   -
nflocks                      200          -          2-                   -
ninode                       476          -         14-                   -
no_lvm_disks                   0          -           -                   -
nproc                        400          -         10-                   -
npty                          60          -          1-                   -
nstrpty                       60          -           -                   -
nswapdev                      10          -          1-25                 -
nswapfs                       10          -          1-25                 -
public_shlibs                  1          -           -                   -
remote_nfs_swap                0          -           -                   -
rtsched_numpri                32          -           -                   -
sema                           0          -          0-1                  -
semmap                   4128768          -          4-                   -
shmem                          0          -          0-1                  -
shmmni                       200          -          3-1024               -
streampipes                    0          -          0-                   -
swapmem_on                     1          -           -                   -
swchunk                     2048          -       2048-16384      kBytes  -
timeslice                     10          -         -1-2147483648 Ticks   -
unlockable_mem              1800          -          0-           Pages   -
#
```

In addition to the tunable parameters, you may want to see a report of all the hardware found on your system. The **ioscan** command does this for you. Using **sysdef** and **ioscan,** you can see what your tunable parameters are set to and what hardware exists on your system. You will then know how your system is set up and can then make changes to your kernel. The following is an **ioscan** output of the same HP-UX 11i L-Class system for which **sysdef** was run:

```
# /usr/sbin/ioscan -f
Class       I  H/W Path     Driver    S/W State   H/W Type     Description
============================================================================
root        0               root      CLAIMED     BUS_NEXUS
ioa         0  0            sba       CLAIMED     BUS_NEXUS    System Bus Adapter (582)
ba          0  0/0          lba      CLAIMED     BUS_NEXUS    Local PCI Bus Adapter (782)
lan         0  0/0/0/0      btlan    CLAIMED     INTERFACE    HP PCI 10/100Base-TX Core
ext_bus     0  0/0/1/0      c720     CLAIMED     INTERFACE      SCSI C896 Fast Wide LVD
target      0  0/0/1/0.7    tgt      CLAIMED     DEVICE
ctl         0  0/0/1/0.7.0  sctl     CLAIMED     DEVICE         Initiator
ext_bus     1  0/0/1/1      c720     CLAIMED     INTERFACE
                                                    SCSI C896 Ultra Wide Single-Ended
target      1  0/0/1/1.2    tgt      CLAIMED     DEVICE
disk        1  0/0/1/1.2.0  sdisk    CLAIMED     DEVICE       SEAGATE ST318203LC
target      2  0/0/1/1.7    tgt      CLAIMED     DEVICE
ctl         1  0/0/1/1.7.0  sctl     CLAIMED     DEVICE       Initiator
ext_bus     2  0/0/2/0      c720     CLAIMED     INTERFACE    SCSI C875 Ultra
Wide Single-Ended
target      3  0/0/2/0.2    tgt       CLAIMED     DEVICE
disk        2  0/0/2/0.2.0  sdisk     CLAIMED     DEVICE       SEAGATE ST318203LC
target      4  0/0/2/0.7    tgt       CLAIMED     DEVICE
ctl         2  0/0/2/0.7.0  sctl      CLAIMED     DEVICE       Initiator
ext_bus     3  0/0/2/1      c720      CLAIMED     INTERFACE
                                                    SCSI C875 Ultra Wide Single-Ended
target      5  0/0/2/1.4    tgt       CLAIMED     DEVICE
disk        3  0/0/2/1.4.0  sdisk    CLAIMED     DEVICE       TOSHIBA CD-ROM XM-6201TA
target      6  0/0/2/1.7    tgt       CLAIMED     DEVICE
ctl         3  0/0/2/1.7.0  sctl      CLAIMED     DEVICE       Initiator
tty         0  0/0/4/0      asio0     CLAIMED     INTERFACE    PCI Serial (103c1048)
tty         1  0/0/5/0      asio0     CLAIMED     INTERFACE    PCI Serial (103c1048)
ba          1  0/1          lba      CLAIMED     BUS_NEXUS   Local PCI Bus Adapter (782)
ba          2  0/2          lba      CLAIMED     BUS_NEXUS   Local PCI Bus Adapter (782)
ba          3  0/3          lba      CLAIMED     BUS_NEXUS   Local PCI Bus Adapter (782)
lan         1  0/3/0/0      btlan     CLAIMED     INTERFACE
```

```
                                                    HP A5230A/B5509BA PCI 10/100Base-TX Addon
ba         4  0/4         lba      CLAIMED     BUS_NEXUS   Local PCI Bus Adapter (782)
ext_bus     4  0/4/0/0      c720       CLAIMED     INTERFACE
                                                        C875 Fast Wide Differential
target      7  0/4/0/0.7    tgt        CLAIMED     DEVICE
ctl         4  0/4/0/0.7.0  sctl       CLAIMED     DEVICE       Initiator
ext_bus     5  0/4/0/1      c720       CLAIMED     INTERFACE    SCSI C875 Fast
Wide Differential
target      8  0/4/0/1.7    tgt        CLAIMED     DEVICE
ctl         5  0/4/0/1.7.0  sctl       CLAIMED     DEVICE       Initiator
ba         5  0/5         lba      CLAIMED     BUS_NEXUS   Local PCI Bus Adapter (782)
ba         6  0/6         lba      CLAIMED     BUS_NEXUS   Local PCI Bus Adapter (782)
ba         7  0/7         lba      CLAIMED     BUS_NEXUS   Local PCI Bus Adapter (782)
ext_bus     6  0/7/0/0      c720       CLAIMED     INTERFACE
                                                        SCSI C875 Fast Wide Differential
target      9  0/7/0/0.7    tgt        CLAIMED     DEVICE
ctl         6  0/7/0/0.7.0  sctl       CLAIMED     DEVICE       Initiator
ext_bus     7  0/7/0/1      c720       CLAIMED     INTERFACE
                                                        SCSI C875 Fast Wide Differential
target     10  0/7/0/1.7    tgt        CLAIMED     DEVICE
ctl         7  0/7/0/1.7.0  sctl       CLAIMED     DEVICE       Initiator
memory      0  8            memory     CLAIMED     MEMORY       Memory
processor   0  160          processor CLAIMED     PROCESSOR    Processor
processor   1  166          processor CLAIMED     PROCESSOR    Processor
#
```

I normally run **ioscan** with the *-f* option because it includes the *Driver, S/W State,* and *H/W Type* columns. I am interested in the driver associated with the hardware in the system that the *-f* option produces.

The **ioscan** output shows all of the hardware that comprises the system, including the two processors in the system.

The file **/stand/vmunix** is the currently running kernel. Here is a long listing of the directory **/stand** on the L-Class system, which shows the file **/stand/vmunix**:

```
# ls -l
total 74274
-rw-r--r--   1 root     sys          19 Aug  4 11:37 bootconf
drwxr-xr-x   4 root     sys        2048 Aug 25 11:24 build
drwxr-xr-x   5 root     sys        1024 Aug 24 13:00 dlkm
drwxr-xr-x   5 root     sys        1024 Aug  4 12:45 dlkm.vmunix.prev
-rw-r--r--   1 root     sys        3024 Aug  4 12:26 ioconfig
-r--r--r--   1 root     sys          82 Aug  4 12:27 kernrel
drwxr-xr-x   2 root     sys        1024 Aug 29 11:39 krs
drwxr-xr-x   2 root     root       1024 Aug 29 11:33 krs_lkg
drwxr-xr-x   2 root     root       1024 Aug 29 11:39 krs_tmp
drwxr-xr-x   2 root     root       8192 Aug  4 11:36 lost+found
-rw-------   1 root     root         12 Aug 29 11:33 rootconf
-rw-rw-rw-   1 root     sys        1180 Aug 24 12:52 system
-r--r--r--   1 root     sys        1026 Aug  4 12:21 system.prev
-rwxr-xr-x   1 root     sys    14774416 Aug 24 12:53 vmunix
-rwxr-xr-x   1 root     sys    23184584 Aug  4 12:22 vmunix.prev
#
```

Notice that among the directories shown are two related to Dynamically Loadable Kernel Modules (DLKM). These are kernel modules that can be included in the kernel without having to reboot the system.

In order to make a change to the kernel, we would change to the **/stand/build** directory, where all work in creating a new kernel is performed, and issue the **system_prep** command as shown below:

```
# cd /stand/build
# /usr/lbin/sysadm/system_prep  -s  system
```

We can now proceed to make the desired changes to the kernel, including adding a driver or subsystem such as cdfs for a CD-ROM file system. With the dynamically loadable kernel module (DLKM) structure in place with 11i, we must use **kmsystem** and **kmtune** to make changes to the kernel system and system description files.

You can use **kmtune** to view the value and parameters related to existing kernel parameters as well as to make proposed modifications to the kernel. The following listing shows issuing **kmtune** (without the *-l* option to view details) to view a summary of the currently running kernel:

```
# kmtune
```

```
Parameter              Current Dyn Planned                    Module     Version
================================================================================
NSTRBLKSCHED                 -  -  2
NSTREVENT                   50  -  50
NSTRPUSH                    16  -  16
NSTRSCHED                    0  -  0
STRCTLSZ                  1024  -  1024
STRMSGSZ                 65535  -  65535
acctresume                   4  -  4
acctsuspend                  2  -  2
aio_listio_max             256  -  256
aio_max_ops               2048  -  2048
aio_physmem_pct             10  -  10
aio_prio_delta_max          20  -  20
allocate_fs_swapmap          0  -  0
alwaysdump                   1  -  1
bootspinlocks                -  -  256
bufcache_hash_locks        128  -  128
bufpages                     0  -  0
chanq_hash_locks           256  -  256
create_fastlinks             0  -  0
dbc_max_pct                 50  -  50
dbc_min_pct                  5  -  5
default_disk_ir              0  -  0
desfree                      -  -  0
disksort_seconds             0  -  0
dnlc_hash_locks            512  -  512
dontdump                     0  -  0
dskless_node                 -  -  0
dst                          1  -  1
effective_maxpid             -  -  ((NPROC<22500)?30000:(NPROC*5/4))
eisa_io_estimate             -  -  0x300
enable_idds                  0  -  0
eqmemsize                   15  -  15
executable_stack             1  -  1
fcp_large_config             0  -  0
file_pad                     -  -  10
fs_async                     0  -  0
ftable_hash_locks           64  -  64
hdlpreg_hash_locks         128  -  128
hfs_max_ra_blocks            8  -  8
hfs_max_revra_blocks         8  -  8
hfs_ra_per_disk             64  -  64
hfs_revra_per_disk          64  -  64
hp_hfs_mtra_enabled          1  -  1
hpux_aes_override            -  -  0
initmodmax                  50  -  50
io_ports_hash_locks         64  -  64
iomemsize                    -  -  40000
ksi_alloc_max              2208  -  2208
ksi_send_max                32  -  32
lotsfree                     -  -  0
max_async_ports             50  -  50
max_fcp_reqs               512  -  512
max_mem_window               0  -  0
max_thread_proc             64  -  64
maxdsiz            0x10000000  -  0x10000000
maxdsiz_64bit      0x40000000  -  0X40000000
maxfiles                    60  -  60
maxfiles_lim              1024  Y  1024
maxqueuetime                 -  -  0
maxssiz             0x800000  -  0X800000
maxssiz_64bit       0x800000  -  0X800000
maxswapchunks             512  -  512
maxtsiz            0x4000000  Y  0X4000000
maxtsiz_64bit      0x40000000  Y  0x40000000
maxuprc                     77  Y  77
maxusers                    32  -  32
maxvgs                      10  -  10
mesg                         1  -  1
minfree                      -  -  0
modstrmax                  500  -  500
msgmap                      42  -  42
msgmax                    8192  Y  8192
msgmnb                   16384  Y  16384
```

```
msgmni                      50  -  50
msgseg                    2048  -  2048
msgssz                       8  -  8
msgtql                      40  -  40
nbuf                         0  -  0
ncallout                   515  -  515
ncdnode                    150  -  150
nclist                     612  -  612
ncsize                    5596  -  5596
ndilbuffers                30  -  30
netisr_priority             -  -  -1
netmemmax                   -  -  0
nfile                      910  -  910
nflocks                    200  -  200
nhtbl_scale                  0  -  0
ninode                     476  -  476
nkthread                   499  -  499
nni                         -  -  2
no_lvm_disks                 0  -  0
nproc                      400  -  500
npty                        60  -  60
nstrpty                     60  -  60
nstrtel                     60  -  60
nswapdev                    10  -  10
nswapfs                     10  -  10
nsysmap                    800  -  800
nsysmap64                  800  -  800
num_tachyon_adapters         0  -  0
o_sync_is_o_dsync            0  -  0
page_text_to_local          -  -  0
pfdat_hash_locks           128  -  128
public_shlibs                1  -  1
region_hash_locks          128  -  128
remote_nfs_swap              0  -  0
rtsched_numpri              32  -  32
scroll_lines               100  -  100
scsi_maxphys           1048576  -  1048576
sema                         1  -  1
semaem                   16384  -  16384
semmap                      66  -  66
semmni                      64  -  64
semmns                     128  -  128
semmnu                      30  -  30
semume                      10  -  10
semvmx                   32767  -  32767
sendfile_max                 0  -  0
shmem                        1  -  1
shmmax               0x4000000  Y  0X4000000
shmmni                     200  -  200
shmseg                     120  Y  120
st_ats_enabled               1  -  1
st_fail_overruns             0  -  0
st_large_recs                0  -  0
streampipes                  0  -  0
swapmem_on                   1  -  1
swchunk                   2048  -  2048
sysv_hash_locks            128  -  128
tcphashsz                    0  -  0
timeslice                   10  -  10
timezone                   420  -  420
unlockable_mem               0  -  0
vas_hash_locks             128  -  128
vnode_cd_hash_locks        128  -  128
vnode_hash_locks           128  -  128
vps_ceiling                 16  -  16
vps_chatr_ceiling      1048576  -  1048576
vps_pagesize                 4  -  4
vx_fancyra_enable            0  -  0
vx_maxlink               32767  -  32767
vx_ncsize                 1024  -  1024
vxfs_max_ra_kbytes        1024  -  1024
vxfs_ra_per_disk          1024  -  1024
#
```

Issuing **kmtune** with the -/ option produces a detailed listing of the kernel. The following shows just the

output for one of the parameters:

```
# kmtune -l
Parameter:     maxuprc
Current:       77
Planned:       77
Default:       75
Minimum:       -
Module:        -
Version:       -
Dynamic:       Yes
#
```

This parameter is *Dynamic* (*Yes*) meaning that the kernel can be dynamically updated. After having viewed this output we can now modify the value of this dynamic parameter. The following command changes the value of the following parameter from *77*, which is the existing value, to *80*:

```
# kmtune -s maxuprc=80
#
```

We can now issue the **kmtune** to again view the existing and proposed value of the *maxuprc* parameter:

```
# kmtune
Parameter             Current Dyn Planned                    Module     Version
================================================================================
NSTRBLKSCHED                -  -  2
NSTREVENT                  50  -  50
NSTRPUSH                   16  -  16
NSTRSCHED                   0  -  0
STRCTLSZ                 1024  -  1024
STRMSGSZ                65535  -  65535
acctresume                  4  -  4
acctsuspend                 2  -  2
aio_listio_max            256  -  256
aio_max_ops              2048  -  2048
aio_physmem_pct            10  -  10
aio_prio_delta_max         20  -  20
allocate_fs_swapmap         0  -  0
alwaysdump                  1  -  1
bootspinlocks               -  -  256
bufcache_hash_locks       128  -  128
bufpages                    0  -  0
chanq_hash_locks          256  -  256
create_fastlinks            0  -  0
dbc_max_pct                50  -  50
dbc_min_pct                 5  -  5
default_disk_ir             0  -  0
desfree                     -  -  0
disksort_seconds            0  -  0
dnlc_hash_locks           512  -  512
dontdump                    0  -  0
dskless_node                -  -  0
dst                         1  -  1
effective_maxpid            -  -  ((NPROC<22500)?30000:(NPROC*5/4))
eisa_io_estimate            -  -  0x300
enable_idds                 0  -  0
eqmemsize                  15  -  15
executable_stack            1  -  1
fcp_large_config            0  -  0
file_pad                    -  -  10
fs_async                    0  -  0
ftable_hash_locks          64  -  64
hdlpreg_hash_locks        128  -  128
hfs_max_ra_blocks           8  -  8
hfs_max_revra_blocks        8  -  8
hfs_ra_per_disk            64  -  64
hfs_revra_per_disk         64  -  64
hp_hfs_mtra_enabled         1  -  1
hpux_aes_override           -  -  0
initmodmax                 50  -  50
io_ports_hash_locks        64  -  64
iomemsize                   -  -  40000
ksi_alloc_max            2208  -  2208
ksi_send_max               32  -  32
lotsfree                    -  -  0
```

```
max_async_ports            50  -  50
max_fcp_reqs              512  -  512
max_mem_window              0  -  0
max_thread_proc            64  -  64
maxdsiz            0x10000000  -  0x10000000
maxdsiz_64bit      0x40000000  -  0X40000000
maxfiles                   60  -  60
maxfiles_lim             1024  Y  1200
maxqueuetime                -  -  0
maxssiz             0x800000  -  0X800000
maxssiz_64bit       0x800000  -  0X800000
maxswapchunks             512  -  512
maxtsiz            0x4000000  Y  0X4000000
maxtsiz_64bit      0x40000000  Y  0X40000000
maxuprc                    77  Y  80
maxusers                   32  -  32
maxvgs                     10  -  10
mesg                        1  -  1
minfree                     -  -  0
modstrmax                 500  -  500
msgmap                     42  -  42
msgmax                   8192  Y  8192
msgmnb                  16384  Y  16384
msgmni                     50  -  50
msgseg                   2048  -  2048
msgssz                      8  -  8
msgtql                     40  -  40
nbuf                        0  -  0
ncallout                  515  -  515
ncdnode                   150  -  150
nclist                    612  -  612
ncsize                   5596  -  5596
ndilbuffers                30  -  30
netisr_priority             -  -  -1
netmemmax                   -  -  0
nfile                     910  -  910
nflocks                   200  -  200
nhtbl_scale                 0  -  0
ninode                    476  -  476
nkthread                  499  -  499
nni                         -  -  2
no_lvm_disks                0  -  0
nproc                     400  -  400
npty                       60  -  60
nstrpty                    60  -  60
nstrtel                    60  -  60
nswapdev                   10  -  10
nswapfs                    10  -  10
nsysmap                   800  -  800
nsysmap64                 800  -  800
num_tachyon_adapters        0  -  0
o_sync_is_o_dsync           0  -  0
page_text_to_local          -  -  0
pfdat_hash_locks          128  -  128
public_shlibs               1  -  1
region_hash_locks         128  -  128
remote_nfs_swap             0  -  0
rtsched_numpri             32  -  32
scroll_lines              100  -  100
scsi_maxphys          1048576  -  1048576
sema                        1  -  1
semaem                  16384  -  16384
semmap                     66  -  66
semmni                     64  -  64
semmns                    128  -  128
semmnu                     30  -  30
semume                     10  -  10
semvmx                  32767  -  32767
sendfile_max                0  -  0
shmem                       1  -  1
shmmax             0x4000000  Y  0X4000000
shmmni                    200  -  200
shmseg                    120  Y  120
st_ats_enabled              1  -  1
st_fail_overruns            0  -  0
```

```
st_large_recs               0  -  0
streampipes                 0  -  0
swapmem_on                  1  -  1
swchunk                  2048  -  2048
sysv_hash_locks           128  -  128
tcphashsz                   0  -  0
timeslice                  10  -  10
timezone                  420  -  420
unlockable_mem              0  -  0
vas_hash_locks            128  -  128
vnode_cd_hash_locks       128  -  128
vnode_hash_locks          128  -  128
vps_ceiling                16  -  16
vps_chatr_ceiling     1048576  -  1048576
vps_pagesize                4  -  4
vx_fancyra_enable           0  -  0
vx_maxlink              32767  -  32767
vx_ncsize                1024  -  1024
vxfs_max_ra_kbytes       1024  -  1024
vxfs_ra_per_disk         1024  -  1024
#
```

This output shows that the change to our parameter is pending.

We can apply the change to the dynamic parameter *maxuprc* from *77* to *80* by issuing **kmtune** with the *-u* option:

```
# kmtune -u
The kernel's value of maxuprc has been set to 80 (0x50).
#
```

This output shows that the change we wanted made to the kernel has been made. We can confirm this by running **kmtune** again and searching for *maxuprc*:

```
# kmtune | grep maxuprc
maxuprc                   80  Y  80
#
```

Both the *Current* and *Planned* values have been updated to *80*. This dynamic update can be done using **kmsystem** to add dynamic drivers to your kernel.

There are many other procedures for which you would have to perform additional steps to include modifications in the kernel and rebuild it. With these non-dynamic changes you would create a new kernel, which will be generated as **/stand/build/vmunix_test,** using the command shown below:

```
# mk_kernel -s system
Compiling conf.c...
Loading the kernel...
Generating kernel symbol table...
#
```

At this point, the new kernel exists in the **/stand/build** directory. The existing kernel is updated with the newly generated kernel with **kmupdate**. **kmupdate** moves the new kernel files into the **/stand** directory. I would first recommend moving the existing **/stand/system** kernel file to a backup file, and then updating the new kernel as shown below:

```
# mv /stand/system /stand/system.prev    (may want to move additional
# kmupdate /stand/build/vmunix_test      files shown in Figure 4-1)

  Kernel update request is scheduled.

  Default kernel /stand/vmunix will be updated by
  newly built kernel /stand/build/vmunix_test
  at next system shutdown or startup time.
#
```

kmupdate will automatically create backup copies of **/stand/vmunix** and **/stand/dlkm** for you. These will be created as **/stand/vmunix.prev** and **/stand/dlkm.vmunix.prev,** respectively.

You can now shut down the system and automatically boot from the new kernel if your update did not take place dynamically and requires a reboot.

Figure 4-1 summarizes the process of building a new kernel in HP-UX 11i.

Figure 4-1. Creating a Kernel in HP-UX 11i

Step	Comments
1) run **sysdef** and **ioscan -f**	Analyzes and reports tunable parameters of currently running kernel.
2) perform long listing of **/stand** directory	The file **vmunix** is the existing kernel, and **system** is used to build a new kernel.
3) **cd /stand/build**	This is the directory where the new kernel will be built.
4) **/usr/lbin/sysadm/system_prep -s system**	This extracts the **system** file from the currently running kernel.
5) use **kmsystem** and **kmtune** to make changes	Takes place in the **/stand/build** directory. Dyamic update complete here.
6) **mk_kernel -s system**	Makes a new kernel in the **/stand/build** directory called **vmunix_test**. DLKM files are produced in **dlkm.vmunix_test/***.
7) **mv /stand/system /stand/system.prev** **mv /stand/vmunix /stand/vmunix.prev** **mv /stand/dlkm /stand/dlkm.vmunix.prev**	Saves the existing files as **.prev**.
8) **mv /stand/build/system /stand/system** **kmupdate /stand/build/vmunix_test**	Updates the kernel with the newly generated kernel. Automatically saves the old versions in **/stand** as follows: **vmunix** as **/stand/vmunix.prev** **dlkm** as **/dlkm.vmunix.prev**
9) **cd /** **shutdown -r 0**	Changes directory to / and shuts down the sytem so that it comes up with the new kernel. This may not be required if your change could be implemented dynamically.

There are really two different procedures for generating your kernel - one for dynamic elements, such as the parameter *maxuprc* shown in the earlier example, and one for static elements. The static procedure consists of several additional steps and a reboot. With HP-UX 11i, more and more kernel objects will be updated dynamically, resulting in fewer reboots when modifying your kernel.

[Team LiB]

PREVIOUS NEXT

kcweb

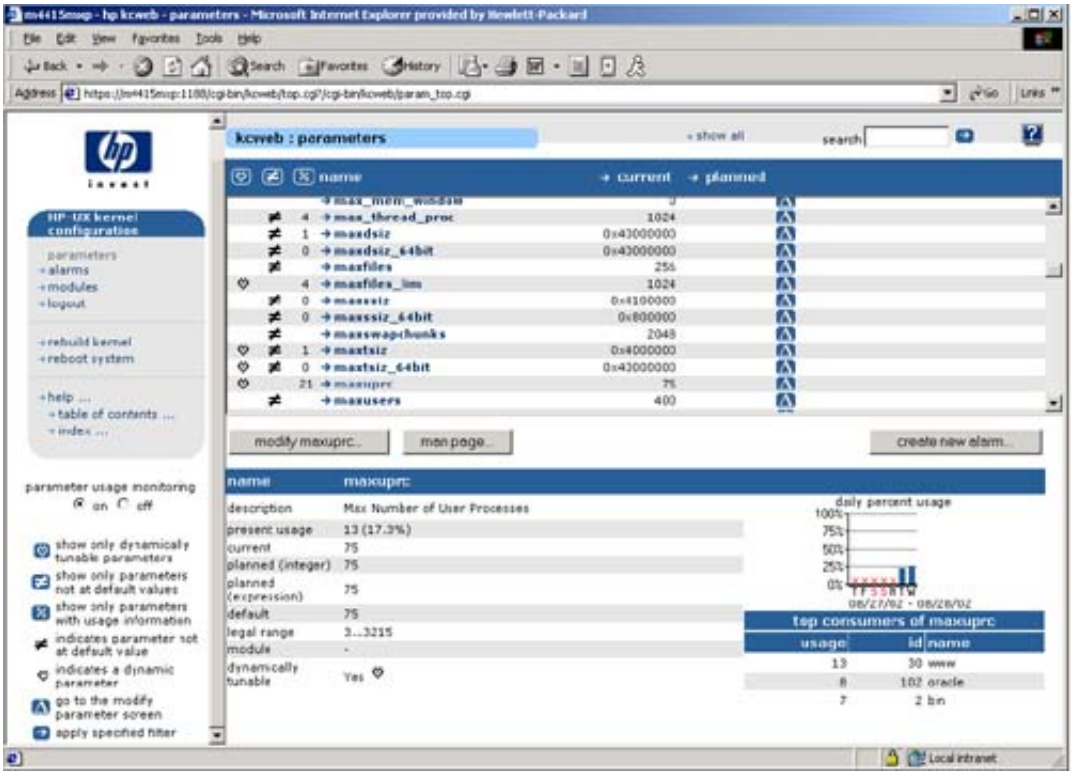
At the time of this writing **kcweb** is a stand-alone tool that is downloaded from www.software.hp.com. Future plans are for **kcweb** to be included with HP-UX distributions and for additional Web-based management to be part of HP-UX. At this time the tool is simple to download and install.

In this section we are able to perform a variety of functions through the Web-based interface. In this section we'll perform the following in **kcweb**:

- View kernel parameters
- Get details on a specific kernel parameter in the bottom of the **kcweb** page and the man page.
- Modify a dynamic kernel parameter and apply the new value.
- Set an alarm to inform us when a kernel parameter exceeds the specified value.

At the time of this writing, **kcweb** is invoked at the command line with **kcweb**. On my system this opens the browser window shown in Figure 4-2. Figure 3-2 shows **kcweb** with several kernel parameters.

Figure 4-2. kcweb Showing a Variety of Parameters



The upper left of Figure 4-2 shows that **kcweb** has *parameters, alarms,* and *modules* functions. We'll cover an example of working with *parameters* and *alarms* in this section. There aren't too many dynamically loadable kernel modules at this time but the technique for working with dynamically loadable modules is similar to that of dynamically tunable parameters so the examples will give you a good idea of the way in which you work with **kcweb**.

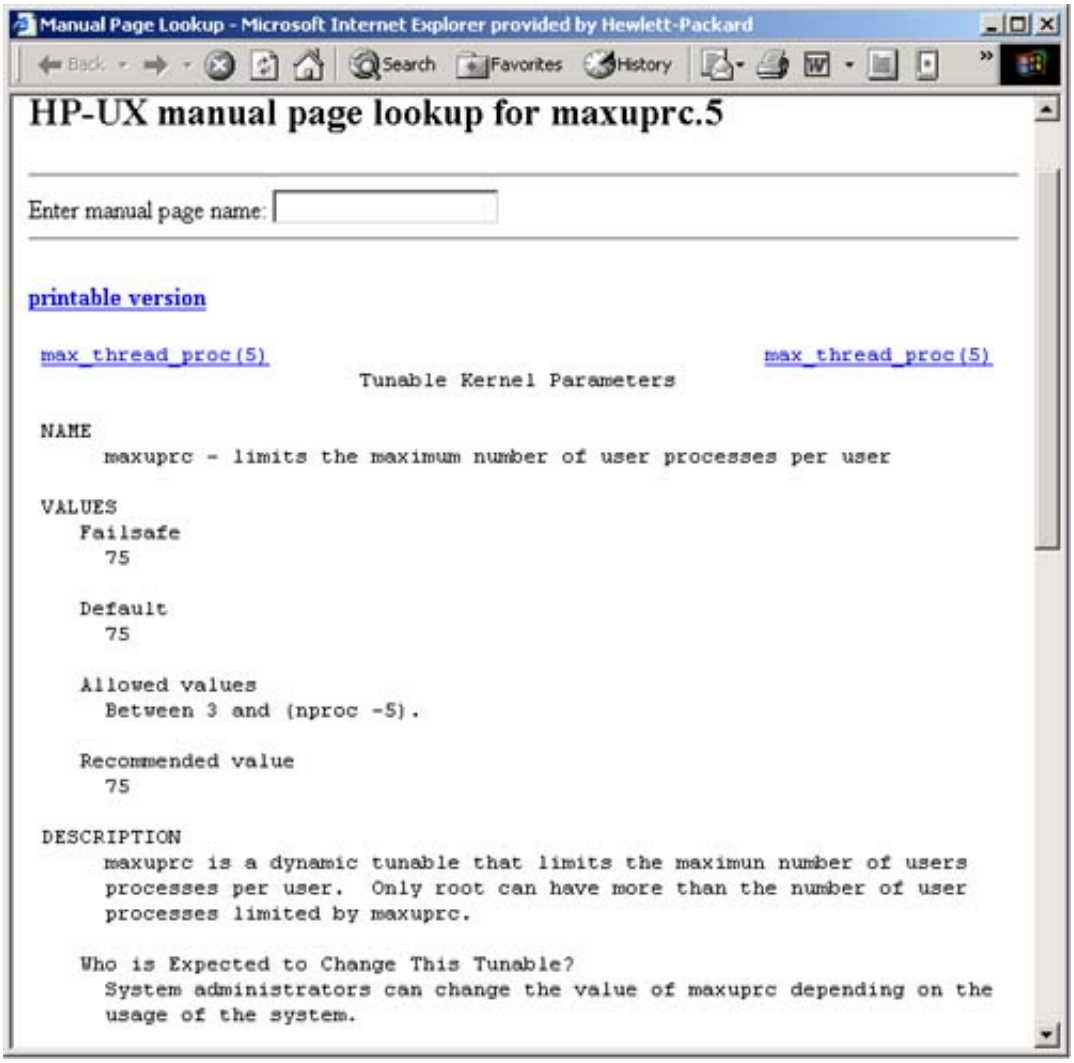
Notice in the bottom left of the figure that there is a legend that includes descriptions of the symbols that

are used in **kcweb**. Those parameters with a heart next to them are dynamically tunable parameters. If you select the heart on the bar across the top of the kernel parameters, then only dynamically tunable parameters will be shown. The "not equal to" sign indicates parameters that are not set to their default value. There are several other entries in the legend as well. This makes for viewing groups of icons easy and the legend helps identify the status of icons.

The bottom of the screen provides information about the kernel parameter selected: in this case *maxuprc.* There is a graph in the bottom right showing the usage of this parameter over time. For system-wide parameters the graph will show usage on a system basis. For user-specific or process-specific parameters, such as *maxuprc*, the graph includes the top five consumers of the parameter.

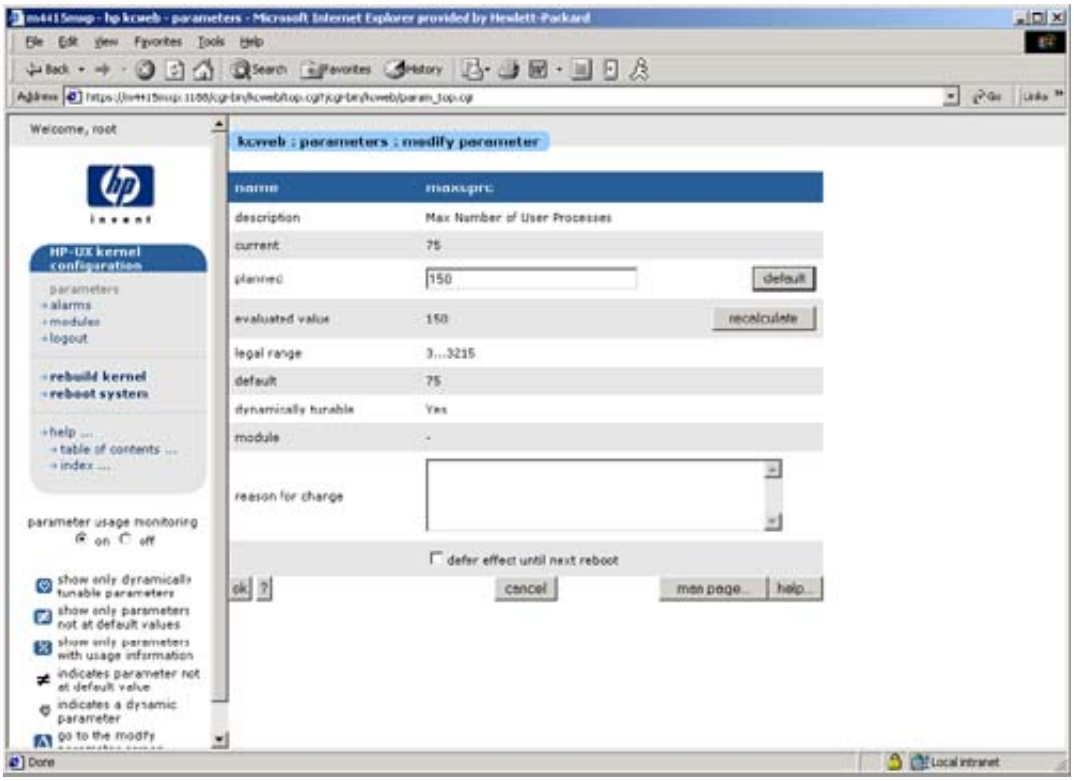
You can get detailed information about a kernel parameter by selecting the *man page...* button as shown in Figure 4-3:

Figure 4-3. kcweb Showing *man page*... For *maxuprc*



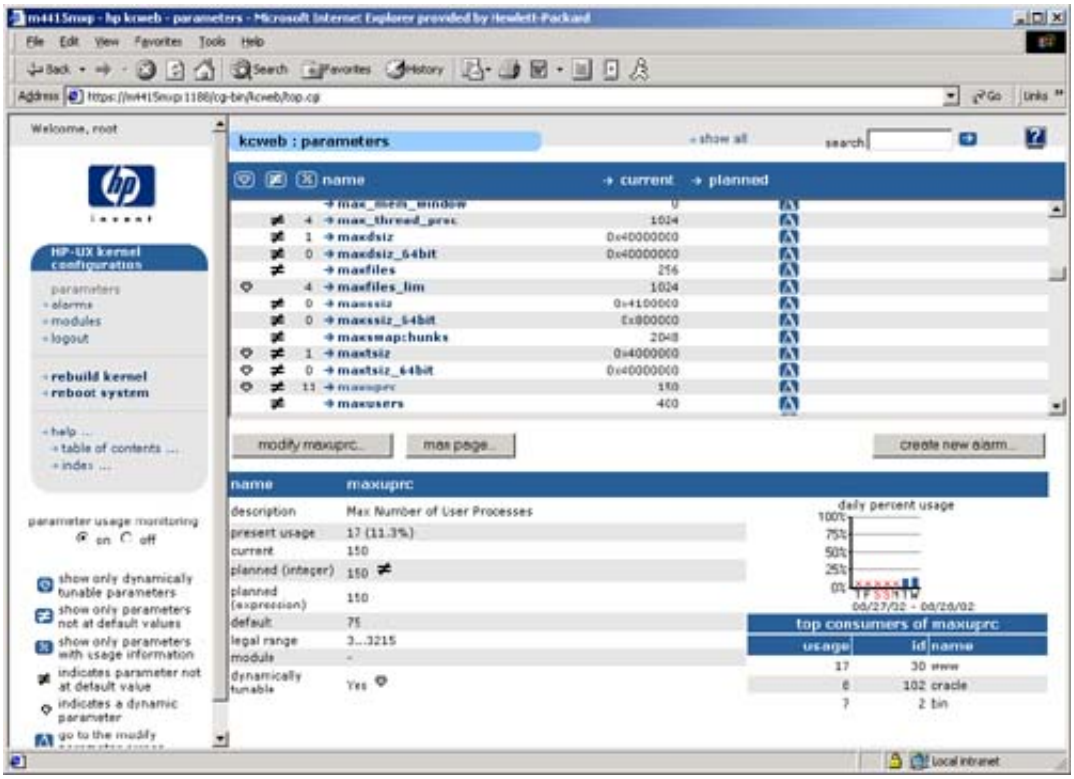
You can also modify one of the parameters by highlighting the parameter and then selecting the *modify <parameter name>* as we've done for the *maxuprc* parameter as we've done in Figure 4-4:

Figure 4-4. kcweb Showing *modify maxuprc*



We've chosen to increase *maxuprc* from *75* to *150* in Figure 4-4. To implement the change we unselect the *defer until next reboot* box and select *ok*. The change is then implemented as you can see in Figure 4-5:

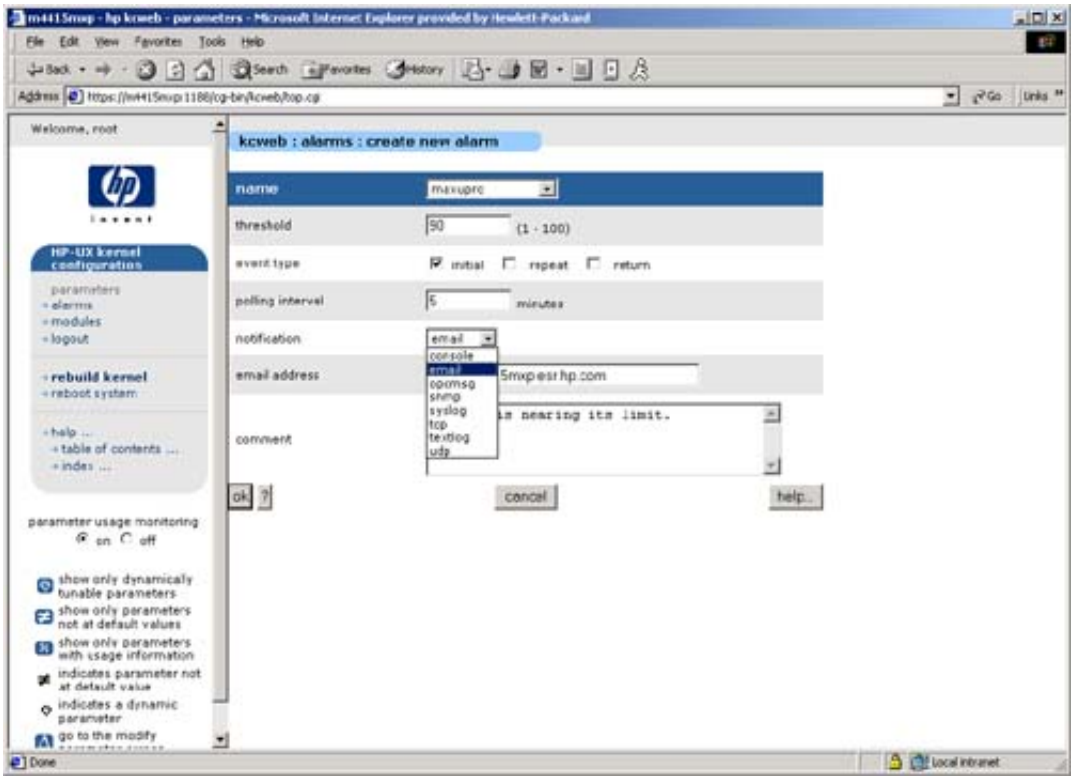
Figure 4-5. kcweb Showing *maxuprc planned* Change



This is a parameter that can be modified dynamically, so it is updated immediately. If this were not a dynamic parameter, we could *rebuild kernel* to update the kernel with the desired change.

Now that we have modified *maxuprc* we can set an alarm to inform us when the parameter reaches a specified threshold. Figure 4-6 shows setting up this alarm:

Figure 4-6. kcweb Showing *maxuprc* Alarm



We get to the *kcweb:alarms* page by selecting *create new alarm...* in the window shown earlier. All of the parameters related to the alarm are shown in Figure 4-6. The setup of the alarm specifies a *threshold* of *90*. All of the options for notification are shown. In this case we've selected email to *root@m4415mxp.esr.hp.com* and specified a comment to appear in the email address.

We can also work with kernel modules in the same way that we work with kernel parameters. Those that are dynamic can be loaded on-the-fly, and those that are not dynamic can be built into the kernel with a rebuild.

This was a quick overview of **kcweb** that included some of the most commonly performed tasks. Since this is a Web-based interface, it is easy to use and most of the screens and information are self explanatory. More Web based management tools will be included in HP-UX over time.



Chapter 5. Devices

Background on Device Files and Peripherals

Device Files in HP-UX 11i

An Example of Adding a Peripheral

Device Files in Cell-Based Systems



Background on Device Files and Peripherals

HP-UX creates special files for most devices it finds attached at the time the system boots. You may, however, have to create your own device files or, at a minimum, have to understand some device files that exist on your system so the information in this chapter is important for every system administrator to know. It's important to know about the devices files for all of the components in your system especially on partition-based systems on which you're always viewing what devices are devoted to what partitions. I'll cover the following topics in this chapter:

- An overview of device files in HP-UX 11i.
- An example of adding a peripheral.
- Some examples of device files in nPartitions (hard partitions) on cell-based systems. These examples were taken directly from Chapter 17 so please see that chapter if you want more detail on device files in nPartitions.

The following section is a general overview of device files.

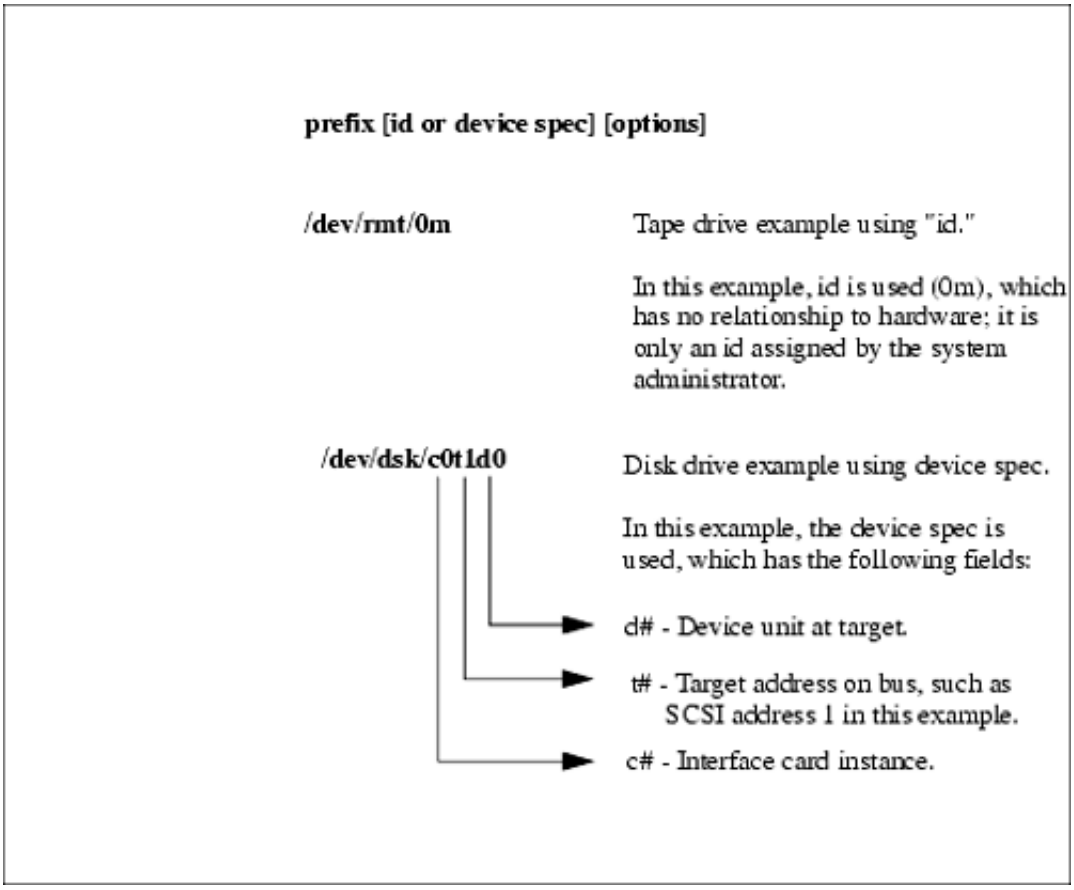
Device Files in HP-UX 11i

What could be more confusing in the UNIX world than device files? Fortunately, in HP-UX, device files for workstations and servers are nearly identical, so if you learn one, your knowledge applies to the other. In this section, I cover:

- The structure of device files
- Some commands associated with helping you work with device files
- Some examples of creating device files

A device file provides the HP-UX kernel with important information about a specific device. The HP-UX kernel needs to know a lot about a device before Input/Output operations can be performed. With HP-UX 11i, the device file naming convention is the same for workstations and server systems. Device files are in the **/dev** directory. There may also be a subdirectory under **/dev** used to further categorize the device files. Examples of a subdirectories are **/dev/dsk,** where disk device files are usually located, and **/dev/rmt**, where tape drive device files are located. Figure 5-1 shows the HP-UX 11i device file-naming convention.

Figure 5-1. HP-UX 11i Device File Naming Convention



There are a number of commands that you can use as you go about creating device files. The **ioscan** command is the first of these. Some of the examples in this section were used in Chapter 3, when building

a kernel was covered. The following is an **ioscan** output of the same workstation for which **sysdef** was run when describing how a kernel is created in Chapter 3. (Using *-f* with **ioscan** would have created a full listing; you should try it with and without *-f.*) I included four **ioscan** outputs so you could see examples from a variety of different systems. The last listing is from a large V-Class system used in some of the examples in Chapter 8, where disks and file systems are covered.

(on workstation)

```
$ /usr/sbin/ioscan

H/W Path     Class          Description
============================================================

             bc
1            graphics       Graphics
2            ba
2/0          unknown
2/0/1           ext_bus     Built-in SCSI
2/0/1.1            target
2/0/1.1.0            disk   HP      C2247
2/0/1.2            target
2/0/1.2.0            disk   TOSHIBA CD-ROM XM-3301TA
2/0/1.6            target
2/0/1.6.0            disk   HP      C2247
2/0/2           lan         Built-in LAN
2/0/4           tty         Built-in RS-232C
2/0/6           ext_bus     Built-in Parallel Interface
2/0/8           audio       Built-in Audio
2/0/10          pc          Built-in Floppy Drive
2/0/10.1           floppy   HP_PC_FDC_FLOPPY
2/0/11          ps2         Built-in Keyboard
8            processor      Processor
9            memory         Memory
```

The following is an **ioscan** output from a server system. Note the four processors shown in this output.

(on server)

```
$ /usr/sbin/ioscan

H/W Path     Class          Description
============================================================

             bc
8            bc             I/O Adapter
10           bc             I/O Adapter
10/0            ext_bus     GSC built-in Fast/Wide SCSI
10/0.3             target
10/0.3.0              disk   HP      C2490WD
10/0.4             target
10/0.4.0              disk   HP      C2490WD
10/0.5             target
10/0.5.0              disk   HP      C2490WD
10/0.6             target
10/0.6.0              disk   HP      C2490WD
10/4         bc             Bus Converter
10/4/0             tty      MUX
10/12        ba             Core I/O Adapter
10/12/0         ext_bus     Built-in Parallel Interface
10/12/5         ext_bus     Built-in SCSI
10/12/5.0          target
10/12/5.0.0           tape   HP       HP35480A
10/12/5.2          target
10/12/5.2.0           disk   TOSHIBA CD-ROM XM-4101TA
10/12/6         lan         Built-in LAN
10//12/7        ps2         Built-in Keyboard/Mouse
32          processor       Processor
34          processor       Processor
36          processor       Processor
38          processor       Processor
49          memory          Memory
```

The next command that helps you when creating device files is **lsdev. lsdev** lists the drivers configured into your system. When adding a device file, you need to have the driver for the device configured into the

system. You can use the manual kernel configuration process covered earlier to add a driver to the kernel. There are columns for the major number for a character device and block device, the driver name, and the class of the driver. The major number, character device, and other parameters are defined later. Here is an example of running **lsdev** on the same workstation on which **ioscan** was run:

(on workstation)

```
$ /usr/sbin/lsdev


Character     Block     Driver     Class
    0          -1       cn         pseudo
    1          -1       ansio0     tty
    3          -1       mm         pseudo
   16          -1       ptym       ptym
   17          -1       ptys       ptys
   24          -1       hil        hil
   27          -1       dmem       pseudo
   46          -1       netdiag1   unknown
   52          -1       lan2       lan
   64          64       lv         lvm
   66          -1       audio      audio
   69          -1       dev_config pseudo
   72          -1       clone      pseudo
   73          -1       strlog     pseudo
   74          -1       sad        pseudo
   75          -1       telm       strtelm
   76          -1       tels       strtels
   77          -1       tlctls     pseudo
   78          -1       tlcots     pseudo
   79          -1       tlcotsod   pseudo
  114          -1       ip         pseudo
  115          -1       arp        pseudo
  116          -1       echo       pseudo
  119          -1       dlpi       pseudo
  130          -1       rawip      pseudo
  136          -1       tcp        pseudo
  137          -1       udp        pseudo
  138          -1       stcpmap    pseudo
  139          -1       nuls       pseudo
  140          -1       netqa      pseudo
  141          -1       tun        pseudo
  142          -1       btlan3     unknown
  143          -1       fddi3      unknown
  144          -1       fddi0      unknown
  145          -1       fcT1_cntl  unknown
  156          -1       ptm        strptym
  157          -1       ptm        strptys
  159          -1       ps2        ps2
  164          -1       pipedev    unknown
  168          -1       beep       graf_pseudo
  169          -1       fcgsc_lan  lan
  170          -1       lpr0       unknown
  174          -1       framebuf   graf_pseudo
  183          -1       diag1      diag
  188          31       sdisk      disk
  189          -1       klog       pseudo
  196          -1       eeprom     da
  203          -1       sctl       ctl
  205          -1       stape      tape
  207          -1       sy         pseudo
  216          -1       CentIF     ext_bus
  227          -1       kepd       pseudo
  229          -1       ite        graf_pseudo
  232          -1       diag2      diag
```

Here is an example of running **lsdev** on the same server on which **ioscan** was run:

(on server)

```
$ /usr/sbin/lsdev

Character     Block     Driver     Class
    0          -1       cn         pseudo
    1          -1       asio0      tty
```

```
  3          -1       mm          pseudo
 16          -1       ptym        ptym
 17          -1       ptys        ptys
 28          -1       diag0       diag
 46          -1       netdiag1    unknown
 52          -1       lan2        lan
 64          64       lv          lvm
 69          -1       dev_config  pseudo
 72          -1       clone       pseudo
 73          -1       strlog      pseudo
 74          -1       sad         pseudo
 75          -1       telm        strtelm
 76          -1       tels        strtels
 77          -1       tlctls      pseudo
 78          -1       tlcots      pseudo
 79          -1       tlcotsod    pseudo
114          -1       ip          pseudo
116          -1       echo        pseudo
119          -1       dlpi        pseudo
130          -1       rawip       pseudo
136          -1       lpr0        unknown
137          -1       udp         pseudo
138          -1       stcpmap     pseudo
139          -1       nuls        pseudo
140          -1       netqa       pseudo
141          -1       tun         pseuod
142          -1       btlan3      unknown
143          -1       fddi3       unknown
144          -1       fddi0       unknown
156          -1       ptm         strptym
157          -1       ptm         strptys
159          -1       ps2         ps2
164          -1       pipedev     unknown
168          -1       beep        graf_pseudo
174          -1       framebuf    graf_pseudo
188          31       sdisk       disk
189          -1       klog        pseudo
193          -1       mux2        tty
203          -1       sctl        ctl
205          -1       stape       tape
207          -1       sy          pseudo
216          -1       CentIF      ext_bus
227          -1       kepd        pseudo
229          -1       ite         graf_pseudo
```

From these two **lsdev** outputs, you can observe some minor differences in the devices. The workstation, for instance, has such classes as audio and floppy, the server has a multiplexer.

You can use **ioscan** to show the device files for a particular peripheral. Going back to the server that had four disks and a CD-ROM attached to it, you could issue the following **ioscan** command to see the device files associated with *disk:*

(on server)

```
$ /usr/sbin/ioscan -fn -C disk

Class  I  H/W Path   Driver  S/W State  H/W Type   Description
============================================================
disk   0 10/0.3.0    sdisk    CLAIMED    DEVICE     HP C2490WD
                     /dev/dsk/c0t3d0  /dev/rdsk/c0t3d0

disk   1 10/0.4.0    sdisk    CLAIMED    DEVICE     HP C2490WD
                     /dev/dsk/c0t4d0  /dev/rdsk/c0t4d0

disk   2 10/0.5.0    sdisk    CLAIMED    DEVICE     HP C2490WD
                     /dev/dsk/c0t5d0  /dev/rdsk/c0t5d0

disk   3 10/0.6.0    sdisk    CLAIMED    DEVICE     HP C2490WD
                     /dev/dsk/c0t6d0  /dev/rdsk/c0t6d0

disk   3 10/12/5/2/0 sdisk    CLAIMED    DEVICE     CD-ROM
                     /dev/dsk/c1t2d0  /dev/rdsk/c1t2d0
```

You can see from this **ioscan** all of the device files associated with *disk,* including the CD-ROM.

You could find out more information about one of these devices with the **diskinfo** command. Specify the character device you want to know more about, as shown below (using the *-v* option for *verbose* provides more detailed information).

```
$ diskinfo /dev/rdsk/c0t5d0

SCSI describe of /dev/rdsk/c0t5d0
             vendor: HP
         product id: C2490WD
               type: direct access
               size: 2082636 bytes
   bytes per sector: 512
```

[Team LiB]

PREVIOUS NEXT

An Example of Adding a Peripheral

In this section, we'll construct a device file for a tape drive. Chapter 8 covers the process of using device files and all Logical Volume Manager commands for adding disk devices that you may want to view as well. Before we construct a device file, let's view two existing device files on a workstation (what we used to call a Series 700) and see where some of this information appears. The first long listing is that of the tape drive, and the second is the disk, both of which are on the workstation in the earlier listings.

(on workstation)

```
$ ll /dev/rmt/0m

crw-rw-rw- 2 bin bin 205 0x003000 Feb 12 03:00 /dev/rmt/0m
```

(on workstation)

```
$ ll /dev/dsk/c0t1d0

brw-r----- 1 bin sys 31 0x001000 Feb 12 03:01 /dev/dsk/c0t1d0
```

The tape drive device file, **/dev/rmt/0m**, shows a major number of 205 corresponding to that shown for the *character* device driver *stape* from **lsdev**. The disk drive device file, **/dev/dsk/c0t6d0**, shows a major number of *31* corresponding to the *block* device driver *sdisk* from **lsdev**. Since the tape drive requires only a character device file and no major number exists for a block *stape* device, as indicated by the *-1* in the *block* column of **lsdev**, this file is the only device file that exists for the tape drive. The disk, on the other hand, may be used as either a block device or a character device (also referred to as the *raw device*). Therefore, we should see a character device file, **/dev/rdsk/c0t6d0**, with a major number of *188*, as shown in **lsdev** for *sdisk*.

(on workstation)

```
$ ll /dev/rdsk/c0t0d0

crw-r-----  1   root   sys   188   0x001000   Feb   12   03:01
/dev/rdsk/c0t1d0
```

We can now create a device file for a second tape drive, this time at SCSI address 2, and a disk device file for a disk drive at SCSI address 5, using the **mksf** command. You can run **mksf** in two different ways. The first form of mksf requires you to include less specific information, such as the minor number. The second form requires you to include more of this specific information. Some of these arguments relate only to the specific form of **mksf** you use.

- *-d* Use the device driver specified. A list of device drivers is obtained with the **lsdev** command.
- *-C* The device specified belongs to this class. The class is also obtained with the **lsdev** command.
- *-H* Use the hardware path specified. Hardware paths are obtained with the **ioscan** command.
- *-m* The minor number of the device is supplied.
- *-r* Create a character, also known as a raw device file. The default is to create a block file.
- *-v* Use verbose output, which prints the name of each special file as it is created.

We will now create the device files for a disk drive. Both a block and a character device file are required. The *0x005000* in the example corresponds to the address of 5 on the disk drive. This number, used in both the block and character device files, is unique for every disk drive created.

We can now create a *block* device file for a disk at SCSI address 5 using the following **mksf** command:

(on workstation)

```
$ /sbin/mksf -v -C disk -m 0x005000 /dev/dsk/c0t5d0

        making /dev/dsk/c0t5d0 b 31 0x005000
```

Similarly, we can now create a *character* device file for a disk at SCSI address 5 using the second form of **mksf**:

(on workstation)

```
$ /sbin/mksf -v -r -C disk -m 0x005000 /dev/dsk/c0t5d0

        making /dev/rdsk/c0t5d0 c 188 0x005000
```

The ***-v*** option used in these examples prints out each device file as it is created. If you wanted to add a second tape drive at SCSI address 2 to your system in addition to the existing tape drive (**/dev/rmt/0m**), you might use the following **mksf** command:

(on workstation)

```
$ /sbin/mksf  -v -C tape -m 0x002000 /dev/rmt/1m

         making /dev/rmt/1m c 205 0x002000
```

Character devices are automatically produced for tape drives, since no block device drivers are required. This fact was found in the output from the **lsdev** command, as indicated by a *-1* in the "Block" column.

With this level of device file background, you should have a good understanding of the device files that SAM will build for you when you add peripherals. By the way, printer device files look very different from the device files I covered here, but all the same principles apply.

[Team LiB]

PREVIOUS NEXT

Device Files in Cell-Based Systems

The information in this section is taken directly from Chapter 17 where you will find a complete overview of nPartitions (hard partitions). Here we provide an overview of the paths and other information you'll see on some cell-based systems.

To start with let's take a look at detailed information for the local nPartition. We're interested in all of the components of our local nPartition as well as a detail of PCI I/O. The following listing shows the PCI detail with the **rad -q** command:

```
# rad -q
                                                                  Driver(s)
Slot        Path        Bus   Speed   Power   Occupied    Suspended   Capable
0-0-1-0     4/0/0       0     33      On      Yes         No          No
0-0-1-1     4/0/1/0     8     33      On      Yes         No          Yes
0-0-1-2     4/0/2/0     16    33      On      Yes         No          Yes
0-0-1-3     4/0/3/0     24    33      On      Yes         No          Yes
0-0-1-4     4/0/4/0     32    66      On      Yes         No          Yes
0-0-1-5     4/0/6/0     48    33      On      Yes         No          Yes
0-0-1-6     4/0/14/0    112   66      On      Yes         No          Yes
0-0-1-7     4/0/12/0    96    33      On      Yes         No          Yes
0-0-1-8     4/0/11/0    88    66      On      Yes         No          Yes
0-0-1-9     4/0/10/0    80    33      On      Yes         No          Yes
0-0-1-10    4/0/9/0     72    33      On      Yes         No          Yes
0-0-1-11    4/0/8/0     64    33      On      Yes         No          Yes
0-0-3-0     5/0/0       0     33      On      Yes         No          No
0-0-3-1     5/0/1/0     8     33      On      Yes         No          Yes
0-0-3-2     5/0/2/0     16    33      On      Yes         No          Yes
0-0-3-3     5/0/3/0     24    33      On      Yes         No          Yes
0-0-3-4     5/0/4/0     32    66      On      Yes         No          Yes
0-0-3-5     5/0/6/0     48    33      On      Yes         No          Yes
0-0-3-6     5/0/14/0    112   66      On      Yes         No          Yes
0-0-3-7     5/0/12/0    96    33      On      Yes         No          Yes
0-0-3-8     5/0/11/0    88    66      On      Yes         No          Yes
0-0-3-9     5/0/10/0    80    33      On      Yes         No          Yes
0-0-3-10    5/0/9/0     72    33      On      Yes         No          Yes
0-0-3-11    5/0/8/0     64    33      On      Yes         No          Yes
0-1-1-0     1/0/0       0     33      On      Yes         No          No
0-1-1-1     1/0/1/0     8     33      On      Yes         No          Yes
0-1-1-2     1/0/2/0     16    33      On      Yes         No          Yes
0-1-1-3     1/0/3/0     24    33      On      Yes         No          Yes
0-1-1-4     1/0/4/0     32    66      On      Yes         No          Yes
0-1-1-5     1/0/6/0     48    33      On      Yes         No          Yes
0-1-1-6     1/0/14/0    112   66      On      Yes         No          Yes
0-1-1-7     1/0/12/0    96    33      On      Yes         No          Yes
0-1-1-8     1/0/11/0    88    66      On      Yes         No          Yes
0-1-1-9     1/0/10/0    80    33      On      Yes         No          Yes
0-1-1-10    1/0/9/0     72    33      On      Yes         No          Yes
0-1-1-11    1/0/8/0     64    33      On      Yes         No          Yes
0-1-3-0     0/0/0       0     33      On      Yes         No          No
0-1-3-1     0/0/1/0     8     33      On      Yes         No          Yes
0-1-3-2     0/0/2/0     16    33      On      Yes         No          Yes
0-1-3-3     0/0/3/0     24    33      On      Yes         No          Yes
0-1-3-4     0/0/4/0     32    66      On      Yes         No          Yes
0-1-3-5     0/0/6/0     48    33      On      Yes         No          Yes
0-1-3-6     0/0/14/0    112   66      On      Yes         No          Yes
0-1-3-7     0/0/12/0    96    33      On      Yes         No          Yes
0-1-3-8     0/0/11/0    88    66      On      Yes         No          Yes
0-1-3-9     0/0/10/0    80    33      On      Yes         No          Yes
0-1-3-10    0/0/9/0     72    33      On      Yes         No          Yes
0-1-3-11    0/0/8/0     64    33      On      Yes         No          Yes
#
```

rad produces an output that gives us a lot of detail about the nPartitions in our system. You should run this command on your Superdome complex so that you can see all of the components of which your Superdome is comprised. There is a lot of detail in this output that is not as easy to understand as the **parstatus** output (covered in Chapter 17). so let's take a closer look at a couple of these fields.

The first field is the slot information, which is in the following form:

```
Cabinet-Bay-Chassis-Slot     such as 0-0-1-0 for the first entry
```

The second field, which is the *Path*, contains the following

```
Cell/SBA/LBA/Device     such as 0/0/1/1 for the second entry
```

As with Virtual Partitions (vPars) covered in Chapter 16, you need to know something about the structure of your system in order to work with nPartitions. The System Bus Adapter (SBA) and Local Bus Adapters (LBAs) are not components that you would typically worry about if you were not working with partitions. For the purpose of working through the information in this chapter, it is sufficient to know that there is a hierarchical I/O structure on HP systems in which the SBA exists at a higher level than the LBA and there are typically several LBAs per SBA. That is why for the second field, SBA of *0* for instance, you'll see several third fields, which are many LBAs per SBA.

Next, let's issue **ioscan -f** to see all of the components of our local nPartition number *0* in a two-nPartition rp8400:

```
mtvnhp01:/>ioscan -f
Class       I  H/W Path        Driver    S/W State   H/W Type     Description
==========================================================================
root        0                   root      CLAIMED     BUS_NEXUS
cell        0  0                cell      CLAIMED     BUS_NEXUS
ioa         0  0/0              sba       CLAIMED     BUS_NEXUS    System Bus Ada
pter (804)
ba          0  0/0/0            lba       CLAIMED     BUS_NEXUS    Local PCI Bus
Adapter (782)
unknown    -1  0/0/0/0/0                 UNCLAIMED   UNKNOWN      PCI BaseSystem
 (103c128d)
tty         0  0/0/0/0/1        asio0     CLAIMED     INTERFACE    PCI Serial (10
3c1048)
lan         0  0/0/0/1/0        gelan     CLAIMED     INTERFACE    HP A3639-60019
 1000Base-T Built-in I/O
ext_bus     0  0/0/0/2/0        c720      CLAIMED     INTERFACE    SCSI C87x Ultr
a Wide Single-Ended
target      0  0/0/0/2/0.6      tgt       CLAIMED     DEVICE
disk        0  0/0/0/2/0.6.0    sdisk     CLAIMED     DEVICE       HP 36.4GST3367
06LC
target      1  0/0/0/2/0.7      tgt       CLAIMED     DEVICE
ctl         0  0/0/0/2/0.7.0    sctl      CLAIMED     DEVICE       Initiator
ext_bus     1  0/0/0/2/1        c720      CLAIMED     INTERFACE    SCSI C87x Fast
 Wide Single-Ended
target      2  0/0/0/2/1.2      tgt       CLAIMED     DEVICE
disk        1  0/0/0/2/1.2.0    sdisk     CLAIMED     DEVICE       HP      DVD-RO
M 305
target      3  0/0/0/2/1.7      tgt       CLAIMED     DEVICE
ctl         1  0/0/0/2/1.7.0    sctl      CLAIMED     DEVICE       Initiator
ext_bus     2  0/0/0/3/0        c720      CLAIMED     INTERFACE    SCSI C896 Fast
 Wide Single-Ended
target      4  0/0/0/3/0.6      tgt       CLAIMED     DEVICE
disk        9  0/0/0/3/0.6.0    sdisk     CLAIMED     DEVICE       HP 36.4GST3367
04LC
target      5  0/0/0/3/0.7      tgt       CLAIMED     DEVICE
ctl         2  0/0/0/3/0.7.0    sctl      CLAIMED     DEVICE       Initiator
ext_bus     3  0/0/0/3/1        c720      CLAIMED     INTERFACE    SCSI C896 Fast
 Wide LVD
target      6  0/0/0/3/1.7      tgt       CLAIMED     DEVICE
ctl         3  0/0/0/3/1.7.0    sctl      CLAIMED     DEVICE       Initiator
ba          1  0/0/1            lba       CLAIMED     BUS_NEXUS    Local PCI Bus
Adapter (782)
ba          2  0/0/2            lba       CLAIMED     BUS_NEXUS    Local PCI Bus
Adapter (782)
ba          3  0/0/4            lba       CLAIMED     BUS_NEXUS    Local PCI Bus
Adapter (782)
ba          4  0/0/6            lba       CLAIMED     BUS_NEXUS    Local PCI Bus
Adapter (782)
ba          5  0/0/8            lba       CLAIMED     BUS_NEXUS    Local PCI Bus
Adapter (782)
fc          0  0/0/8/0/0        td        CLAIMED     INTERFACE    HP Tachyon TL/
TS Fibre Channel Mass Storage Adapter
fcp         2  0/0/8/0/0.10     fcp       CLAIMED     INTERFACE    FCP Domain
ext_bus    20  0/0/8/0/0.10.30.0.0        fcparray  CLAIMED     INTERFACE    FCP
 Array Interface
target      7  0/0/8/0/0.10.30.0.0.0      tgt       CLAIMED     DEVICE
```

```
disk       12  0/0/8/0/0.10.30.0.0.0.0    sdisk     CLAIMED     DEVICE       HP
     A6188A
disk       13  0/0/8/0/0.10.30.0.0.0.1    sdisk     CLAIMED     DEVICE       HP
     A6188A
disk       16  0/0/8/0/0.10.30.0.0.0.2    sdisk     CLAIMED     DEVICE       HP
     A6188A
ext_bus    21  0/0/8/0/0.10.30.255.0      fcpdev    CLAIMED     INTERFACE    FCP
 Device Interface
target      8  0/0/8/0/0.10.30.255.0.0    tgt       CLAIMED     DEVICE
ctl        14  0/0/8/0/0.10.30.255.0.0.0  sctl      CLAIMED     DEVICE       HP
     A6188A
ext_bus    22  0/0/8/0/0.10.31.0.0        fcparray  CLAIMED     INTERFACE    FCP
 Array Interface
target      9  0/0/8/0/0.10.31.0.0.0      tgt       CLAIMED     DEVICE
disk       14  0/0/8/0/0.10.31.0.0.0.0    sdisk     CLAIMED     DEVICE       HP
     A6188A
disk       15  0/0/8/0/0.10.31.0.0.0.1    sdisk     CLAIMED     DEVICE       HP
     A6188A
disk       17  0/0/8/0/0.10.31.0.0.0.2    sdisk     CLAIMED     DEVICE       HP
     A6188A
ext_bus    23  0/0/8/0/0.10.31.255.0      fcpdev    CLAIMED     INTERFACE    FCP
 Device Interface
target     10  0/0/8/0/0.10.31.255.0.0    tgt       CLAIMED     DEVICE
ctl        15  0/0/8/0/0.10.31.255.0.0.0  sctl      CLAIMED     DEVICE       HP
     A6188A
ba          6  0/0/10          lba       CLAIMED     BUS_NEXUS    Local PCI Bus
Adapter (782)
ba          7  0/0/10/0/0      PCItoPCI  CLAIMED     BUS_NEXUS    PCItoPCI Bridg
e
lan         1  0/0/10/0/0/4/0  btlan     CLAIMED     INTERFACE    HP A5506B PCI
10/100Base-TX 4 Port
lan         2  0/0/10/0/0/5/0  btlan     CLAIMED     INTERFACE    HP A5506B PCI
10/100Base-TX 4 Port
lan         3  0/0/10/0/0/6/0  btlan     CLAIMED     INTERFACE    HP A5506B PCI
10/100Base-TX 4 Port
lan         4  0/0/10/0/0/7/0  btlan     CLAIMED     INTERFACE    HP A5506B PCI
10/100Base-TX 4 Port
ba          8  0/0/12          lba       CLAIMED     BUS_NEXUS    Local PCI Bus
Adapter (782)
ba          9  0/0/14          lba       CLAIMED     BUS_NEXUS    Local PCI Bus
Adapter (782)
memory      0  0/5             memory    CLAIMED     MEMORY       Memory
processor   0  0/10            processor CLAIMED     PROCESSOR    Processor
processor   1  0/11            processor CLAIMED     PROCESSOR    Processor
processor   2  0/12            processor CLAIMED     PROCESSOR    Processor
processor   3  0/13            processor CLAIMED     PROCESSOR    Processor
cell        1  2               cell      CLAIMED     BUS_NEXUS
memory      1  2/5             memory    CLAIMED     MEMORY       Memory
processor   4  2/10            processor CLAIMED     PROCESSOR    Processor
processor   5  2/11            processor CLAIMED     PROCESSOR    Processor
processor   6  2/12            processor CLAIMED     PROCESSOR    Processor
processor   7  2/13            processor CLAIMED     PROCESSOR    Processor
mtvnhp01:/>
```

This **ioscan** is context sensitive in that all of the components in nPartition *0* are shown in its output, including a total of eight processors. Cells *0* and *2* are in this nPartition. The following listing shows an **ioscan** output of the second nPartition in the same rp8400:

```
mtvnhp02:/>ioscan -f
Class       I  H/W Path        Driver    S/W State   H/W Type     Description
==========================================================================
root        0                  root      CLAIMED     BUS_NEXUS
cell        0  1               cell      CLAIMED     BUS_NEXUS
ioa         0  1/0             sba       CLAIMED     BUS_NEXUS    System Bus Ada
pter (804)
ba          0  1/0/0           lba       CLAIMED     BUS_NEXUS    Local PCI Bus
Adapter (782)
unknown    -1  1/0/0/0/0                 UNCLAIMED   UNKNOWN      PCI BaseSystem
 (103c128d)
tty         0  1/0/0/0/1       asio0     CLAIMED     INTERFACE    PCI Serial (10
3c1048)
lan         0  1/0/0/1/0       gelan     CLAIMED     INTERFACE    HP A3639-60019
 1000Base-T Built-in I/O
ext_bus     0  1/0/0/2/0       c720      CLAIMED     INTERFACE    SCSI C87x Ultr
a Wide Single-Ended
```

```
target      0  1/0/0/2/0.6     tgt       CLAIMED     DEVICE
disk        0  1/0/0/2/0.6.0   sdisk     CLAIMED     DEVICE       HP 36.4GST3367
06LC
target      1  1/0/0/2/0.7     tgt       CLAIMED     DEVICE
ctl         0  1/0/0/2/0.7.0   sctl      CLAIMED     DEVICE       Initiator
ext_bus     1  1/0/0/2/1       c720      CLAIMED     INTERFACE    SCSI C87x Ultr
a Wide Single-Ended
target      2  1/0/0/2/1.0     tgt       CLAIMED     DEVICE
tape        0  1/0/0/2/1.0.0   stape     CLAIMED     DEVICE       HP      C5683A
target      3  1/0/0/2/1.7     tgt       CLAIMED     DEVICE
ctl         1  1/0/0/2/1.7.0   sctl      CLAIMED     DEVICE       Initiator
ext_bus     2  1/0/0/3/0       c720      CLAIMED     INTERFACE    SCSI C896 Ultr
a2 Wide Single-Ended
target      4  1/0/0/3/0.6     tgt       CLAIMED     DEVICE
disk        1  1/0/0/3/0.6.0   sdisk     CLAIMED     DEVICE       HP 36.4GST3367
04LC
target      5  1/0/0/3/0.7     tgt       CLAIMED     DEVICE
ctl         2  1/0/0/3/0.7.0   sctl      CLAIMED     DEVICE       Initiator
ext_bus     3  1/0/0/3/1       c720      CLAIMED     INTERFACE    SCSI C896 Ultr
a2 Wide LVD
target      6  1/0/0/3/1.7     tgt       CLAIMED     DEVICE
ctl         3  1/0/0/3/1.7.0   sctl      CLAIMED     DEVICE       Initiator
ba          1  1/0/1           lba       CLAIMED     BUS_NEXUS    Local PCI Bus
Adapter (782)
ba          2  1/0/2           lba       CLAIMED     BUS_NEXUS    Local PCI Bus
Adapter (782)
ba          3  1/0/4           lba       CLAIMED     BUS_NEXUS    Local PCI Bus
Adapter (782)
ba          4  1/0/6           lba       CLAIMED     BUS_NEXUS    Local PCI Bus
Adapter (782)
ba          5  1/0/8           lba       CLAIMED     BUS_NEXUS    Local PCI Bus
Adapter (782)
fc          0  1/0/8/0/0       td        CLAIMED     INTERFACE    HP Tachyon TL/
TS Fibre Channel Mass Storage Adapter
fcp         0  1/0/8/0/0.10    fcp       CLAIMED     INTERFACE    FCP Domain
ext_bus     4  1/0/8/0/0.10.30.0.0        fcparray  CLAIMED     INTERFACE    FCP
 Array Interface
target      7  1/0/8/0/0.10.30.0.0.0      tgt       CLAIMED     DEVICE
disk        2  1/0/8/0/0.10.30.0.0.0.0    sdisk     CLAIMED     DEVICE       HP
     A6188A
disk        3  1/0/8/0/0.10.30.0.0.0.1    sdisk     CLAIMED     DEVICE       HP
     A6188A
disk        4  1/0/8/0/0.10.30.0.0.0.2    sdisk     CLAIMED     DEVICE       HP
     A6188A
ext_bus     5  1/0/8/0/0.10.30.255.0      fcpdev    CLAIMED     INTERFACE    FCP
 Device Interface
target      8  1/0/8/0/0.10.30.255.0.0    tgt       CLAIMED     DEVICE
ctl         4  1/0/8/0/0.10.30.255.0.0.0  sctl      CLAIMED     DEVICE       HP
     A6188A
ext_bus     6  1/0/8/0/0.10.31.0.0        fcparray  CLAIMED     INTERFACE    FCP
 Array Interface
target      9  1/0/8/0/0.10.31.0.0.0      tgt       CLAIMED     DEVICE
disk        5  1/0/8/0/0.10.31.0.0.0.0    sdisk     CLAIMED     DEVICE       HP
     A6188A
disk        6  1/0/8/0/0.10.31.0.0.0.1    sdisk     CLAIMED     DEVICE       HP
     A6188A
disk        7  1/0/8/0/0.10.31.0.0.0.2    sdisk     CLAIMED     DEVICE       HP
     A6188A
ext_bus     7  1/0/8/0/0.10.31.255.0      fcpdev    CLAIMED     INTERFACE    FCP
 Device Interface
target     10  1/0/8/0/0.10.31.255.0.0    tgt       CLAIMED     DEVICE
ctl         5  1/0/8/0/0.10.31.255.0.0.0  sctl      CLAIMED     DEVICE       HP
     A6188A
ba          6  1/0/10          lba       CLAIMED     BUS_NEXUS    Local PCI Bus
Adapter (782)
ba          7  1/0/10/0/0      PCItoPCI  CLAIMED     BUS_NEXUS    PCItoPCI Bridg
e
lan         1  1/0/10/0/0/4/0  btlan     CLAIMED     INTERFACE    HP A5506B PCI
10/100Base-TX 4 Port
lan         2  1/0/10/0/0/5/0  btlan     CLAIMED     INTERFACE    HP A5506B PCI
10/100Base-TX 4 Port
lan         3  1/0/10/0/0/6/0  btlan     CLAIMED     INTERFACE    HP A5506B PCI
10/100Base-TX 4 Port
lan         4  1/0/10/0/0/7/0  btlan     CLAIMED     INTERFACE    HP A5506B PCI
10/100Base-TX 4 Port
```

```
ba          8  1/0/12          lba       CLAIMED     BUS_NEXUS    Local PCI Bus
Adapter (782)
ba          9  1/0/14          lba       CLAIMED     BUS_NEXUS    Local PCI Bus
Adapter (782)
memory      0  1/5             memory    CLAIMED     MEMORY       Memory
processor   0  1/10            processor CLAIMED     PROCESSOR    Processor
processor   1  1/11            processor CLAIMED     PROCESSOR    Processor
processor   2  1/12            processor CLAIMED     PROCESSOR    Processor
processor   3  1/13            processor CLAIMED     PROCESSOR    Processor
mtvnhp02:/>
```

This **ioscan** output includes all components in nPartition *0* in this rp8400. Cell *1* is shown in this **ioscan** output as well.

The **ioscan** outputs produced a lot of components for our local partition. The form of the **ioscan** output for an nPartition looks like the following:

```
    Field 1         Field 2        3     Field 4    Field 5  Field 6

Global cell no./proc, mem, or SBA/LBA/Card address/Function/dev addr
```

The six fields in the **ioscan** provide you with a lot of useful information about the components in your local nPartition. The LBA number is not the same as the PCI card number in many cases. Be careful when configuring I/O on any HP 9000 to check the PCI card slot vs. the LBA number.

Please see Chapter 17 for more information on commands related to working with cell-based systems.

[Team LiB]



Chapter 6. Backup

Built-In Backup Programs

Backup of Virtual Partitions



Built-In Backup Programs

Most HP-UX system administrators employ a dual backup strategy. The first is to use the **make_tape_recovery** command of Ignite-UX to create a bootable backup tape of their root volume (**make_tape_recovery** is covered in Chapter 9). You can create a bootable recovery archive across a network with **make_net_recovery**. Secondly, a backup program is used to back up the balance of data on the system. In this chapter I'll give an overview of several backup commands. I won't cover any advanced backup programs such as HP's OmniBack. Advanced backup tools, however, can make tasks such as centralized backup and overall management of backup much easier.The last section in this chapter covers backup of Virtual Partitions (vPars) and can be applied to hard partitions (nPartitions) as well.

Here is a brief overview of backup programs I'll cover in upcoming sections:

tar	**tar** is widely considered the most *portable* of the backup and restore programs. **tar** is the most popular generic backup utility. You will find that many applications are shipped on **tar** tapes. This is the most widely used format for exchanging data with other UNIX systems. **tar** is the oldest UNIX backup method and therefore runs on all UNIX systems. You can append files to the end of a **tar** tape, which you can't do with **fbackup**. When sending files to another UNIX user, I would strongly recommend **tar**, but **tar** is as slow as molasses, so you won't want to use it for your full or incremental backups. One highly desirable aspect of **tar** is that when you load files onto a tape with **tar** and then restore them onto another system, the original users and groups are retained.
fbackup	**fbackup** has a lot of functionality associated with it, such as specifying whether the backup is full or incremental, different *levels* of backup, files and directories to be included or excluded, support for a *graph* file which specifies files to be included and excluded, and other advanced features. **fbackup** is an HP-UX-only utility, and tapes can be read using **frecover** on HP-UX systems only.
cpio	**cpio** is also portable and easy to use, like **tar**. In addition, **cpio** is much faster than **tar** - not as fast as **fbackup**, but much faster than **tar**. **cpio** is good for replicating directory trees.
dd	This is a bit-for-bit copy. It is not smart in the sense that it does not copy files and ownerships; it just copies bits. You could not, therefore, select an individual file from a **dd** tape as you could with **frecover, tar**, **restore** or **cpio. dd** is used primarily to convert data such as EBCDIC to ASCII.
dump	**dump** is similar to **fbackup**. If you use **fbackup** on HP-UX, you will see much similarity when you use **dump. dump** provides the same level backup scheme as **fbackup** and creates **/var/adm/dumpdates,** which lists the last time a file system was backed up. **restore** is used to read information backed up with **dump**. **dump**, however, works only with HFS file systems and not VxFS, and it assumes that you are using a reel tape.

tar

tar is widely considered the most *portable* of the backup and restore programs. You will find that many applications are shipped on **tar** tapes and many UNIX files downloaded from the Internet are in **tar** format. This is the most widely used format for exchanging data with other UNIX systems. **tar** is the oldest UNIX backup method and therefore runs on all UNIX systems. You can append files to the end of a **tar** tape, which you can't do with many other programs. When sending files to another UNIX user, I would strongly recommend **tar**. **tar** is as slow as molasses, so you won't want to use it for your full or incremental backups if you have a lot of data to back up. One highly desirable aspect of **tar** is that when you load files onto a tape with **tar** and then restore them onto another system, the original users and groups are retained.

We'll use several tar commands in the upcoming examples, including the following:

```
# tar cf /dev/rmt/0m /var       ;use tar to create (c) an archive of
                                the directory /var and put it on
                                tape /dev/rmt/0m.

# tar tvf /dev/rmt/0m           ;obtain table of contents (t) from
```

```
                                tape /dev/rmt/0m and produce
                                produce verbose (v) output.


# tar xvf /dev/rmt/0m             ;extract (x) the entire contents
                                 of the archive on tape /dev/rmt/0m
                                to default destination.

# tar xvf /dev/rmt/0m file1       ;extract (x) only file1
                                 from the archive on tape /dev/rmt/0m
                                 to default destination.
```

You'll notice when you view the man pages for tar that options are preceded by a hyphen. The command works without the hyphen so most tar examples, including those in this chapter, omit the hyphen.

Let's take a look at some examples using **tar**. Let's begin by performing a **tar** backup (usually called *creating an archive*) of the directory **/var** to tape device **/dev/rmt/0m**. We use the *c* option to create a backup and the *f* option to specify the file of the tape drive **/dev/rmt/0m**:

```
# tar cf /dev/rmt/0m /var
tar: /var/opt/dce/rpc/local/01060/reaper is not a file. Not dumped
tar: /var/opt/dce/rpc/local/00997/reaper is not a file. Not dumped
tar: /var/opt/dce/rpc/local/00997/c-3/7000 is not a file. Not dumped
tar: /var/opt/dce/rpc/local/00997/c-3/shared is not a file. Not dumped
tar: /var/opt/dce/rpc/local/00997/c-3/7002 is not a file. Not dumped
tar: /var/opt/dce/rpc/local/s-0/135 is not a file. Not dumped
tar: /var/opt/dce/rpc/local/s-0/2121 is not a file. Not dumped
tar: /var/opt/dce/rpc/local/s-3/135 is not a file. Not dumped
tar: /var/opt/dce/rpc/local/s-3/2121 is not a file. Not dumped
tar: /var/spool/sockets/pwgr/client933 is not a file. Not dumped
tar: /var/spool/sockets/pwgr/client1028 is not a file. Not dumped
tar: /var/spool/sockets/pwgr/client1152 is not a file. Not dumped
tar: /var/spool/sockets/pwgr/client1172 is not a file. Not dumped
tar: /var/spool/sockets/pwgr/client1173 is not a file. Not dumped
tar: /var/spool/sockets/pwgr/client1139 is not a file. Not dumped
tar: /var/spool/sockets/pwgr/client2500 is not a file. Not dumped
tar: /var/spool/sockets/pwgr/client2592 is not a file. Not dumped
tar: /var/spool/sockets/pwgr/client2490 is not a file. Not dumped
tar: /var/spool/sockets/pwgr/client2593 is not a file. Not dumped
tar: /var/spool/pwgr/daemon is not a file. Not dumped
#
```

The result of this command printed only problem-related messages to *standard output*. You will often see the *v* option used with tar to produce verbose output, which would have listed both the messages above and those related to files that were successfully written to the tape archive.

Next let's take a look at only the files on the tape with the string *eaaa* in them. To produce a table of contents, we will use the *t* option. The following example also uses *v* for verbose output:

```
# tar tvf /dev/rmt/0m | grep eaaa
rw-rw-rw-   0/3     28 Jul 11 15:37 2000 /var/tmp/eaaa01299
rw-rw-rw-   0/3     28 Jul 11 15:37 2000 /var/tmp/eaaa01333
rw-rw-rw-   0/3     28 Jul 11 15:38 2000 /var/tmp/eaaa01354
rw-rw-rw-   0/3     28 Jul 11 15:40 2000 /var/tmp/eaaa01380
rw-rw-rw-   0/3     28 Jul 11 15:40 2000 /var/tmp/eaaa01405
rw-rw-rw-   0/3     28 Jul 11 15:45 2000 /var/tmp/eaaa01487

#
```

This output shows several files that begin with *eaaa* on the tape. We'll delete the last of these files from the computer and restore it from tape using the *x* option to extract the file from the **tar** archive. We'll then list the directory on the system to confirm that the file we deleted has been restored to the directory from tape.

```
# rm /var/tmp/eaaa01487
#
# tar xvf /dev/rmt/0m /var/tmp/eaaa01487

x /var/tmp/eaaa01487, 28 bytes, 1 tape blocks

#
# ls -l /var/tmp/eaaa*
-rw-rw-rw-   1 root     sys           28 Jul 11 15:37 /var/tmp/eaaa01299
-rw-rw-rw-   1 root     sys           28 Jul 11 15:37 /var/tmp/eaaa01333
-rw-rw-rw-   1 root     sys           28 Jul 11 15:38 /var/tmp/eaaa01354
```

```
-rw-rw-rw-   1 root     sys           28 Jul 11 15:40 /var/tmp/eaaa01380
-rw-rw-rw-   1 root     sys           28 Jul 11 15:40 /var/tmp/eaaa01405
-rw-rw-rw-   1 root     sys           28 Jul 11 15:45 /var/tmp/eaaa01487

#
```

This backup and restore using **tar** is simple and gets the job done.

A common use for **tar** is to back up files from one directory and restore them to another directory. We'll backup the contents of **/var/tmp** and restore them to the directory **/tmp/puttarfileshere**. In the following example, we create a **tar** backup archive to a file rather than to tape. The file is called **tartest**. We then move this file to the destination directory and extract it there. We don't use a tape at all in this example.

```
# cd /var/tmp
# ls -l
total 72
-rw-------   1 root       sys             0 Jul 11 15:57 OBAMFEAa01630
-rw-------   1 root       sys             0 Jul 11 15:20 OBAMHBAa01020
-rw-------   1 root       sys             0 Jul 11 15:50 OBAMHBAa01540
-rw-rw-rw-   1 root       sys           102 Jul 11 15:20 aaaa01112
-rw-rw-rw-   1 root       sys           102 Jul 11 15:37 aaaa01299
-rw-rw-rw-   1 root       sys           102 Jul 11 15:37 aaaa01333
-rw-rw-rw-   1 root       sys           102 Jul 11 15:38 aaaa01354
-rw-rw-rw-   1 root       sys           102 Jul 11 15:40 aaaa01380
-rw-rw-rw-   1 root       sys            99 Jul 11 15:40 aaaa01405

                  -
                  -
                  -
 1 root       sys            28 Jul 11 15:37 eaaa01333
-rw-rw-rw-   1 root       sys             28 Jul 11 15:38 eaaa01354
-rw-rw-rw-   1 root       sys             28 Jul 11 15:40 eaaa01380
-rw-rw-rw-   1 root       sys             28 Jul 11 15:40 eaaa01405
-rw-rw-rw-   1 root       sys             28 Jul 11 15:45 eaaa01487
-rwxr--r--   1 root       root            28 Jul 11 16:04 envd.action2
-rwxr--r--   1 root       root            28 Jul 11 16:04 envd.action5
dr-xr-xr-x   2 bin        bin             96 Jul 11 13:50 ntp
-rw-r--r--   1 root       sys            600 Jul 11 15:27 swagent.log
#
# tar cvf /tmp/tartest 'ls'
a OBAMFEAa01630 0 blocks
a OBAMHBAa01020 0 blocks
a OBAMHBAa01540 0 blocks
a aaaa01112 1 blocks
a aaaa01299 1 blocks
a aaaa01333 1 blocks
a aaaa01354 1 blocks
a aaaa01380 1 blocks
a aaaa01405 1 blocks

                  -
                  -
                  -
a eaaa01354 1 blocks
a eaaa01380 1 blocks
a eaaa01405 1 blocks
a eaaa01487 1 blocks
a envd.action2 1 blocks
a envd.action5 1 blocks
a swagent.log 2 blocks
#
# cd /tmp
# mkdir puttarfileshere
# cp tartest puttarfileshere
# cd puttarfileshere
# ls -l
total 80
-rw-rw-rw-   1 root       sys          40960 Jul 11 17:09 tartest
#
# tar xvf tartest
x OBAMFEAa01630, 0 bytes, 0 tape blocks
x OBAMHBAa01020, 0 bytes, 0 tape blocks
x OBAMHBAa01540, 0 bytes, 0 tape blocks
```

```
x aaaa01112, 102 bytes, 1 tape blocks
x aaaa01299, 102 bytes, 1 tape blocks
x aaaa01333, 102 bytes, 1 tape blocks
x aaaa01354, 102 bytes, 1 tape blocks
x aaaa01380, 102 bytes, 1 tape blocks
x aaaa01405, 99 bytes, 1 tape blocks
                  -
                  -
                  -

x daaa01405, 28 bytes, 1 tape blocks
x daaa01487, 28 bytes, 1 tape blocks
x eaaa01299, 28 bytes, 1 tape blocks
x eaaa01333, 28 bytes, 1 tape blocks
x eaaa01354, 28 bytes, 1 tape blocks
x eaaa01380, 28 bytes, 1 tape blocks
x eaaa01405, 28 bytes, 1 tape blocks
x eaaa01487, 28 bytes, 1 tape blocks
x envd.action2, 28 bytes, 1 tape blocks
x envd.action5, 28 bytes, 1 tape blocks
x swagent.log, 600 bytes, 2 tape blocks
#
```

When creating the **tar** backup, I first changed to the **/var/tmp** directory and then used the **ls** command (a *grav* or *accent*, which is near the upper left of most keyboards on the same key as a *tilde*; appears before and after the **ls**). This produced relative path names that I could easily restore to the **/tmp/puttarfileshere** directory. Alternatively, I could simply have changed directory to **/var** and issued the command **tar cf /dev/rmt/0m tmp** to back up the entire contents of the **/var/tmp** directory.

This entire process could have been done on a single command line. The following line is from the **tar** file man page and shows the procedure for producing an archive in the *fromdir* and restoring it to the *todir*:

**cd** *fromdir* ; **tar cf - . | (cd** *todir* ; **tar xf -i)**

The "-" in the **tar cf** command tells **tar** to send its data to *standard output*. The "-" in the **tar xf** command tells **tar** to look to *standard input* for data, which is the data produced by **tar cf -** issued earlier on the command line.

cpio

cpio is a powerful utility that is used in conjunction with **find** (see Chapter 25 dedicated to **find**) in order to perform full and incremental backups. **cpio** is an established UNIX utility that works similarly on most UNIX variants.

We'll use several commands in the upcoming examples, including the following:

```
# find . -print | cpio -oBv > /dev/rmt/0m  ;find the contents of
                                            the current dir and
                                            write them to tape.

# cpio -it < /dev/rmt/0m       ;read table of contents (t) of tape.


# cpio -icvBdum < /dev/rmt/0m ;restore (i) the contents of tape,
                              this is the most widely used
                               cpio command.


# find . -print | cpio -oBv | (remsh tapesys dd of=/dev/rmt/0m)
          ;find the contents of the current dir and
           write (o) them to tape on remote machine tapesys.


# remsh tapesys "dd if=/dev/rmt/0m bs=8k" | cpio -icvBdum
                  ;restore the contents (i) of a tape on remote system
                   tapesys to the local system.
```

The first command we'll issue is to **find** the contents in **/var/tmp** and write them to our tape device **/dev/rmt/0m**. The options to **cpio** used in the following example are *o* for output mode, *B* for block output, and *v* for verbose reporting:

```
# cd /var/tmp
# find . -print | cpio -oBv > /dev/rmt/0m
(Using tape drive with immediate report mode enabled (reel #1).)
.
envd.action2
envd.action5
swagent.log
ntp
OBAMHBAa01020
aaaa01558
aaaa01426
aaaa01112
OBAMHBAa01540
aaaa01299
                -
                -
                -
eaaa01487
OBAMFEAa01630
cmd_res8215
tmp_cfg_file
cmd_res8708
exclude.temp
arch.include.1
3570 blocks
#
```

In the example, we first changed directory to **/var/tmp**, then issue the **find** command and pipe its output to **cpio**. **cpio** is almost always used in conjunction with **find** in the manner shown in the example. This produced a backup with relative path names because we changed to the directory **/var/tmp** before issuing the backup commands.

Next we'll view the contents of the tape to see the files we wrote to it with cpio. The *i* option is used for input, and the *t* option is used to get a table of contents in the following listing:

```
# cpio -it < /dev/rmt/0m
.
envd.action2
envd.action5
swagent.log
ntp
OBAMHBAa01020
aaaa01558
aaaa01426
aaaa01112
OBAMHBAa01540
aaaa01299
                -
                -
                -
eaaa01487
OBAMFEAa01630
cmd_res8215
tmp_cfg_file
cmd_res8708
exclude.temp
arch.include.1
3570 blocks
#
```

Now that we have written to the tape and viewed its table of contents, we'll restore the contents of **/var/tmp**. In the following example, we use several options to **cpio**, including *i* for input mode, *c* for ASCII header format, *v* for verbose, *B* for block output, *d* for directories, *u* for unconditional write over existing files, and *m* to restore the original modification times:

```
# cpio -icvBdum < /dev/rmt/0m
.
envd.action2
envd.action5
swagent.log
ntp
OBAMHBAa01020
aaaa01558
aaaa01426
```

```
aaaa01112
OBAMHBAa01540
aaaa01299
                -
                -
                -
eaaa01487
OBAMFEAa01630
cmd_res8215
tmp_cfg_file
cmd_res8708
exclude.temp
arch.include.1
3570 blocks
#
```

The **cpio** command produces a list of files that will be read from the tape and restored to the system. Since we included the verbose option, we'll see all the information related to the restore.

Now that we've seen how to write a tape, produce a table of contents, and read the contents of a tape on a local system, let's work with a tape drive on a remote system. We'll perform a backup to a remote tape drive, view the table of contents on the tape, and then restore using the remote tape drive.

First let's perform a backup to a remote tape drive. The local system, which does not have a tape drive attached to it, is *or1.* The remote system, which has a tape drive attached to it, is *tapesys*. We'll run **cpio** (using the same three options earlier described) on *or1* and run a remote shell and **dd** on *tapesys,* which will store the contents of the backup. We'll run these commands from **/var/tmp** on *or1* in the following example:

```
# find . -print | cpio -oBv | (remsh tapesys dd of=/dev/rmt/0m)

.
envd.action2
envd.action5
swagent.log
ntp
OBAMHBAa01020
aaaa01558
aaaa01426
aaaa01112
OBAMHBAa01540
aaaa01299
                -
                -
                -
eaaa01487
OBAMFEAa01630
cmd_res8215
tmp_cfg_file
cmd_res8708
exclude.temp
arch.include.1
3570 blocks
#
```

Now let's come back to our local system without the tape drive and restore the contents of the **cpio** tape we just produced, but let's restore them to a different directory. The directory to which we'll restore the contents of the tape (originally in **/var/tmp**) is **/tmp/remotecpiofiles**. This is similar to the process we performed in the tar section earlier in this chapter in which we restored tar files to a different location.

In the following example, we issue a series of commands while on *or1*. The last of these commands is to issue a **remsh** to system *tapesys* which has on it a tape drive with the **cpio** tape we just created. We **dd** the information and pipe it through **cpio** to restore the contents of the tape. In this example, we use the same restore options to **cpio**, including *i* for input mode, *c* for ASCII header format, *v* for verbose, *B* for block output, *d* for directories, *u* for unconditional write over existing files, and *m* to restore original modification times:

```
# hostname
or1
# cd /tmp
# mkdir remotecpiofiles
# cd remotecpiofiles
# pwd
/tmp/remotecpiofiles
```

```
# remsh tapesys "dd if=/dev/rmt/0m bs=8k" | cpio -icvBdum
.
envd.action2
envd.action5
swagent.log
ntp
OBAMHBAa01020
aaaa01558
aaaa01426
aaaa01112
OBAMHBAa01540
aaaa01299
                    -
                    -
                    -
eaaa01487
OBAMFEAa01630
cmd_res8215
tmp_cfg_file
cmd_res8708
exclude.temp
arch.include.1
3570 blocks
#
# pwd
/tmp/remotecpiofiles
# ls
envd.action2
envd.action5
swagent.log
ntp
OBAMHBAa01020
aaaa01558
aaaa01426
aaaa01112
OBAMHBAa01540
aaaa01299
                    -
                    -
                    -
eaaa01487
OBAMFEAa01630
cmd_res8215
tmp_cfg_file
cmd_res8708
exclude.temp
arch.include.1
#
```

The **ls** we issued at the end of this example confirmed that we did indeed write the contents of the tape on the remote system to the new directory **/tmp/remotecpiofiles**. You may want to add the *-a* command to the **ls** option to ensure that the files have contents.

You can build from the simple examples in this cpio section to develop backup and restore commands to meet your needs in a modest environment.

fbackup and frecover

fbackup and **frecover** are the preferred backup and restore programs on HP-UX. Backups produced with **fbackup** are not portable to other UNIX variants. If you're working in a heterogeneous environment, you won't be able to take **fbackup** tapes produced on an HP-UX system and recover them to a system running a different UNIX variant.

In this section I'll cover issuing fbackup and frecover at the command line. Although **fbackup** and **frecover** are the most advanced programs bundled with your HP-UX system for backup and restore, your needs may go beyond these programs. There are also advanced backup programs you can procure from both HP and third parties. In general, I find that the capabilities of **fbackup** and **frecover** are sufficient for new HP-UX installations. If, however, you have a highly distributed environment or need to back up large amounts of data, perform backups on systems with a variety of operating systems, or need to use several backup devices simultaneously, you may want to consider a more advanced product.

fbackup has the capability of performing backups at different *levels*. The levels define the amount of information to be included in the backup. A full backup, which is covered in this section, is backup level 0. The other levels define various degrees of incremental backups. I am a strong advocate of performing a full backup, and then performing incremental backups of every file that has changed since the *last full backup*. This means that to recover from a completely "hosed" (a technical term meaning *destroyed*) system, you would need your full backup tape and only one incremental tape (you would restore your root volume with a bootable Ignite-UX tape produced with **make_tape_recovery**, which is covered in Chapter 9). If, for instance, you performed a full backup on Sunday and an incremental backups on Monday through Friday, you would need to load only Sunday's full backup tape and Friday's incremental backup tape to completely restore your system. **fbackup** supports this scheme.

Here is an explanation of the **fbackup** command and *some* of its options:

```
/usr/sbin/fbackup -f device [-0-9] [-u] [-i path] [-e path] [-g graph]
```

-f device	The tape drive for the backup, such as /**dev/rmt/0m** for your local tape drive.
[-0-9]	This is the level of the backup. If you run a full backup on Sunday at level 0, then you would run an incremental backup at level 1 the other days of the week. An incremental backup will back up all information changed since a backup was made at a lower level. You could back up at 0 on Sunday, 1 on Monday, 2 on Tuesday, and so on. However, to recover your system, you would need to load Sunday's tape, then Monday's tape, then Tuesday's tape, and so on, to fully recover.
[-u]	This updates the database of past backups so that it contains such information as the backup level, time of the beginning and end of the backup session, and the graph file (described below) used for the backup session. This is valid only with the *-g* (graph) option.
[-i path]	The specified path is to be included in the backup. This can be issued any number of times.
[-e path]	The specified path is to be excluded from the backup. This can also be specified any number of times.
[-g graph]	The graph file contains the list of files and directories to be included or excluded from the backup.

Although **fbackup** is quite thorough and easy to use, it does not have embedded in it the day and time at which full and incremental backups will be run. You have to make a **cron** entry to run **fbackup** automatically.

In its simplest form, we could run **fbackup** and specify only the tape drive with the *f* option and the directory to back up with the *i* option as shown in the following example:

```
# fbackup -f /dev/rmt/0m -i /var/tmp
fbackup(1004): session begins on Wed Jul 12 14:26:30 2000
fbackup(3205): WARNING: unable to read a volume header
fbackup(3024): writing volume 1 to the output file /dev/rmt/0m
fbackup(3055): total file blocks read for backup: 3606
fbackup(3056): total blocks written to output file /dev/rmt/0m: 3857
fbackup(1030): warnings encountered during backup
#
```

fbackup did not produce a list of files included in the backup since we did not include the *v* option for verbose.

To view the contents of the tape, we run **frecover** with the options *r* for read, *N* to prevent the contents of the tape from being restored to the system, and *v* for verbose, as shown in the following command:

```
# frecover -rNv -f /dev/rmt/0m
drwxr-xr-x      root    root    /
dr-xr-xr-x      bin     bin     /var
drwxrwxrwx      bin     bin     /var/tmp
-rw-------      root    sys     /var/tmp/OBAMFEAa01630
-rw-------      root    sys     /var/tmp/OBAMHBAa01020
-rw-------      root    sys     /var/tmp/OBAMHBAa01540
-rw-------      root    sys     /var/tmp/OBAMHBAa07762
-rw-rw-rw-      root    sys     /var/tmp/aaaa01112
                  -
                  -
                  -
```

```
-rw-rw-rw-      root    sys     /var/tmp/eaaa01487
-rwxr--r--      root    root    /var/tmp/envd.action2
-rwxr--r--      root    root    /var/tmp/envd.action5
-rw-rw-rw-      root    sys     /var/tmp/exclude.temp
dr-xr-xr-x      bin     bin     /var/tmp/ntp
-rw-r--r--      root    sys     /var/tmp/swagent.log
-rw-rw-rw-      root    sys     /var/tmp/tmp_cfg_file
#
```

Let's now delete a file from the system that was included as part of the **fbackup**. We'll then restore only the file we deleted. We'll use the *x* option for extract and the *i* option to specify the file to include with **frecover**, as shown in the following example:

```
# cd /var/tmp
# ls -l aa*
-rw-rw-rw-   1 root       sys            102 Jul 11 15:20 aaaa01112
-rw-rw-rw-   1 root       sys            102 Jul 11 15:37 aaaa01299
-rw-rw-rw-   1 root       sys            102 Jul 11 15:37 aaaa01333
-rw-rw-rw-   1 root       sys            102 Jul 11 15:38 aaaa01354
-rw-rw-rw-   1 root       sys            102 Jul 11 15:40 aaaa01380
-rw-rw-rw-   1 root       sys             99 Jul 11 15:40 aaaa01405
-rw-rw-rw-   1 root       sys            102 Jul 11 14:57 aaaa01426
-rw-rw-rw-   1 root       sys             99 Jul 11 15:45 aaaa01487
-rw-rw-rw-   1 root       sys            102 Jul 11 14:24 aaaa01558
# rm aaaa01487
# cd /
# frecover -x -i /var/tmp/aaaa01487 -f /dev/rmt/0m
# cd /var/tmp
# ls -l aaa*
-rw-rw-rw-   1 root       sys            102 Jul 11 15:20 aaaa01112
-rw-rw-rw-   1 root       sys            102 Jul 11 15:37 aaaa01299
-rw-rw-rw-   1 root       sys            102 Jul 11 15:37 aaaa01333
-rw-rw-rw-   1 root       sys            102 Jul 11 15:38 aaaa01354
-rw-rw-rw-   1 root       sys            102 Jul 11 15:40 aaaa01380
-rw-rw-rw-   1 root       sys             99 Jul 11 15:40 aaaa01405
-rw-rw-rw-   1 root       sys            102 Jul 11 14:57 aaaa01426
-rw-rw-rw-   1 root       sys             99 Jul 11 15:45 aaaa01487
-rw-rw-rw-   1 root       sys            102 Jul 11 14:24 aaaa01558
#
```

In the previous example, we successfully restored the file **/var/tmp/aaa01487** from the tape using **frestore**.

There are some powerful aspects to **fbackup** that we did not employ in our example. These include *backup levels, graph files*, and *index files*.

fbackup supports backup levels 0-9. 0 is used for full backups and the other digits indicate various incremental backup levels (see the **fbackup** man page at the end of this chapter for additional information).

Graph files are used to specify the files to be included in the backup.

Index files contain a list of files produced as part of the backup.

Let's take a look at an example that employs all of these functions. First, we create a graph file that contains the files we wish to include (*i*) or exclude (*e*) as part of the backup. In our case the file will contain only the following line:

```
i /var/tmp
```

Let's now run **fbackup** with *u* to update the backup database, *0* for a full backup, *f* to specify the file to which we want to write the backup (we'll use a file rather than tape in this example), *g* to specify our graph file, *I* to specify the name of the index file, and finally, we'll redirect messages to a file that will contain the backup log. We'll add the date and time to the end of the index and backup log files.

```
# fbackup -0u -f /tmp/testbackup -g /tmp/backupgraph
              -I /tmp/backupindex.`date '+%y%m%d.%H:%M'` 2>
              /tmp/backuplog.`date '+%y%m%d.%H:%M'`
#
```

Let's now see what files were produced as a result of having issued this command. First let's look at the backup index and backup log files:

```
# ls /tmp/backup*
```

```
backupgraph
backupgraph000712.15.04
backupindex.000712.15:04
backuplog.000712.15:04
#
# cat /tmp/backupindex000712.15:04
1024                    1 /
1024                    1 /var
2048                    1 /var/tmp
0                       1 /var/tmp/OBAMFEAa01630
0                       1 /var/tmp/OBAMHBAa01020
0                       1 /var/tmp/OBAMHBAa01540
336                     1 /var/tmp/OBAMHBAa07762
102                     1 /var/tmp/aaaa01112
                    -
                    -
                    -
28                      1 /var/tmp/eaaa01487
28                      1 /var/tmp/envd.action2
28                      1 /var/tmp/envd.action5
4608                    1 /var/tmp/exclude.temp
96                      1 /var/tmp/ntp
1595                    1 /var/tmp/swagent.log
205                     1 /var/tmp/tmp_cfg_file
#
# cat /tmp/backupgraph000712.15:04
i/var/tmp
#
# cat /tmp/backuplog000712.15:04

fbackup(1004): session begins on Wed Jul 12 14:56:19 2000
fbackup(3024): writing volume 1 to the output file /tmp/testbackup
fbackup(1030): warnings encountered during backup
fbackup(3055): total file blocks read for backup: 3614
fbackup(3056): total blocks written to output file /tmp/testbackup:3876
#
```

The three files with the date appended (*7/12/00 time 15:04*) to the end of the filename were produced by the **fbackup** command issued earlier. The date appended to the end of the file can help in the organization of backup files. We could restore any or all of these files with **frestore** from the file that contains the backup information (**/tmp/testbackup**) as demonstrated earlier.

We are not restricted to performing backups to a tape drive attached to the local system. We can backup to a remote tape with **fbackup** by specifying the system name and tape drive, or file, to which we want to store the files. The following example uses **fbackup** options covered earlier and also includes the name of the system with the tape drive:

```
# fbackup -f tapesys:/dev/rmt/0m -i /var/tmp -v
fbackup(1004): session begins on Wed Jul 12 15:56:22 2000
fbackup(3307): volume 1 has been used -1 time(s) (maximum: 100)
fbackup(3024): writing vol 1 to output file tapesys:/dev/rmt/0m
    1: / 2
    2: /var 2
    3: /var/tmp 4
    4: /var/tmp/AAAa11812 0
    5: /var/tmp/AAAa11992 0
    6: /var/tmp/BEQ19522 19
    7: /var/tmp/DCE19522 0
    8: /var/tmp/DEC19522 0
    9: /var/tmp/ISPX19522 0
                    -
                    -
                    -
   73: /var/tmp/eaaa13306 1
   74: /var/tmp/ems_inittab.old 2
   75: /var/tmp/envd.action2 1
   76: /var/tmp/envd.action5 1
   77: /var/tmp/inetd.conf.old 9
   78: /var/tmp/net19522 1799
   79: /var/tmp/swagent.log 16
fbackup(1005): run time: 43 seconds
fbackup(3055): total file blocks read for backup: 1974
fbackup(3056): total blocks written to output file or1:/dev/rmt/0m: 0
#
```

This command performs the backup of **/var/tmp** on system *or1* by sending the files to the tape drive on system *tapesys*. We could have used many additional options to **fbackup** as demonstrated in earlier examples, but I wanted to keep the example simple so it would be easy to see the remote tape drive specification.

dd

dd is a utility for writing the contents of a device, such as a disk, to tape. You can also use **dd** to copy an image of a tape to a file on your system, and then you can look at the file.

First, let's write the contents of a directory to tape using **tar**, then we'll use **dd** to copy the tape contents as a file. The only option to **dd** required in this example is *if,* which specifies the input file as the tape drive.

```
# cd /var/tmp
# tar cf /dev/rmt/0m `ls`
# cd /tmp
# dd if=/dev/rmt/0m > /tmp/tapecontents
183+0 records in
183+0 records out
# tar tv /tmp/tapecontents
rw-------   0/3      0 Jul 11 15:57 2000 OBAMFEAa01630
rw-------   0/3      0 Jul 11 15:20 2000 OBAMHBAa01020
rw-------   0/3      0 Jul 11 15:50 2000 OBAMHBAa01540
rw-------   0/3    336 Jul 12 13:18 2000 OBAMHBAa07762
rw-------   0/3      0 Jul 12 13:44 2000 OBAMHBAa08333
rw-rw-rw-   0/3    102 Jul 11 15:20 2000 aaaa01112
                   -
                   -
                   -
rw-rw-rw-   0/3     28 Jul 11 15:45 2000 eaaa01487
rwxr--r--   0/0     28 Jul 11 21:53 2000 envd.action2
rwxr--r--   0/0     28 Jul 11 21:53 2000 envd.action5
rw-rw-rw-   0/3   2304 Jul 12 12:42 2000 exclude.temp
r-xr-xr-x   2/2      0 Jul 11 13:50 2000 ntp/
rw-r--r--   0/3   1595 Jul 12 12:23 2000 swagent.log
rw-rw-rw-   0/3    205 Jul 12 12:39 2000 tmp_cfg_file
#
```

Now we can look at the contents of the file with **tar** as shown in an earlier section, with the following command:

```
# tar tv /tmp/tapecontents
```

Another common use of **dd** is to extract a **cpio** archive from a tape drive on a remote system to a local system. In the following example the local system without a tape drive is *or1,* and the remote system with a tape drive is *tapesys*. We'll read the tape to a directory on *or1* using **dd** and **cpio** (see the earlier cpio section for an explanation of the options) in the following example:

```
# hostname
or1
# pwd
/tmp/remotecpiofiles
# remsh tapesys "dd if=/dev/rmt/0m bs=8k" | cpio -icvBdum
.
envd.action2
envd.action5
swagent.log
ntp
OBAMHBAa01020
aaaa01558
                -
                -
                -
eaaa01487
OBAMFEAa01630
cmd_res8215
tmp_cfg_file
cmd_res8708
OBAMHBAa07762
exclude.temp
arch.include.1
```

```
3580 blocks
0+358 records in
0+358 records out
#
```

This command runs a remote shell (**remsh**) to run the **dd** command on *tapesys*. The output of this command is piped to **cpio** on our local system to extract the archive. In this example, only the *if* option was used to specify the input file. The *of* option, which specifies the output file was not needed. If you were to perform a **dd** of a disk device to a tape drive, the *of* would be the tape drive on your system, such as **/dev/rmt/0m**.

dump and restore

dump is similar to **fbackup**. If you use **fbackup** on HP-UX, you will see much similarity when you use **dump**. **dump** provides levels as part of the backup scheme and creates **/var/adm/dumpdates,** which lists the last time a filesystem was backed up. **restore** is used to read information backed up with **dump**. **dump**, however, works only with HFS file systems, not with VxFS, and it assumes that you are using a reel tape. **vxdump** and **vxrestore** are used for VxFS. Generally speaking, you will not find **dump** (and **vxdump**) and **restore** (and **vxrestore**) recommended as backup and restore programs on HP-UX. **fbackup** and **cpio** are the preferred backup programs on HP-UX. There is, however, no reason why you can't use **dump** and **restore** as long as you keep in mind the file system type limitation.

Let's take a look at some examples using **dump** and **restore**. Our examples will actually use **vxdump** and **vxrestore**; however. Nearly the same usage applies to both the HFS and VxFS programs. We'll use several commands in the upcoming examples, including the following:

```
# vxdump 0fu /dev/rmt/0m /var     ;dump vxfs file system /var to tape
                                /dev/rmt/0m using level 0 and
                                 update /var/adm/dumpdates.



# vxrestore tf /dev/rmt/0m | grep eaaa  ;obtain table of contents from
                                        tape /dev/rmt/0m and look for
                                        file name containing "eaaa"

# vxrestore -x -f /dev/rmt/0m ./tmp/eaaa01487     ;restore file to
                                                 current directory
```

Let's take a look at some of these commands in more detail and I'll provide more explanation for what is taking place.

Our first example runs **vxdump** to back up the directory **/var** with a backup level of *0* for full backup, the *f* option to specify the output file **/dev/rmt/0m**, and *u* for a write to **/var/adm/dumpdates**:

```
# vxdump 0fu /dev/rmt/0m /var
  vxdump: Date of this level 0 dump: Tue Jul 11 16:41:27 2000
  vxdump: Date of last level 0 dump: the epoch
  vxdump: Dumping /dev/vg00/rlvol10 to /dev/rmt/0m
  vxdump: mapping (Pass I) [regular files]
  vxdump: mapping (Pass II) [directories]
  vxdump: estimated 428058 blocks (209.01MB).
  vxdump: dumping (Pass III) [directories]
  vxdump: dumping (Pass IV) [regular files]
  vxdump: vxdump: 214146 tape blocks on 1 volumes(s)
  vxdump: level 0 dump on Tue Jul 11 16:41:27 2000
  vxdump: Closing /dev/rmt/0m
  vxdump: vxdump is done
#
```

vxdump provides information related to the backup to *standard output*.

Next, let's view the table of contents on the tape using **vxrestore**, looking for files that begin with *eaaa*. We'll then delete one of these files from the system and use **vxrestore** to restore it from tape. To produce the table of contents, we use the *t* option, and to extract the file from tape, we use the *x* option to **vxrestore** as shown in the following listing:

```
#
# vxrestore tf /dev/rmt/0m | grep eaaa
       404      ./tmp/eaaa01299
       678      ./tmp/eaaa01333
```

```
       700      ./tmp/eaaa01354
       736      ./tmp/eaaa01380
       741      ./tmp/eaaa01405
       717      ./tmp/eaaa01487
#
#
# rm /var/tmp/eaaa01487
#
# vxrestore -x -f /dev/rmt/0m ./tmp/eaaa01487
You have not read any tapes yet.
Unless you know which volume your file(s) are on you should start
with the last volume and work towards the first.
Specify next volume #: 1
set owner/mode for '.'? [yn] y


# cd /var/tmp
# ls -l eaaa01487
total 2
-rw-rw-rw-   1 root       sys             28 Jul 11 15:45 eaaa01487
#
```

Notice that as part of restoring the file, we had to specify a volume number of *1* and whether or not we wanted to set the mode for the file.

The examples in this section showed creating a backup tape with **vxdump**, producing a table of contents, and restoring with **vxrestore**. Although **fbackup** and **cpio** are the recommended backup solutions on HP-UX, you can use **dump** and **restore** if you are familiar with these programs and would like to use them. Due to the portability of **tar**, it is often used for backup and restore as well.

[Team LiB]

PREVIOUS NEXT

Backup of Virtual Partitions

Backup of Virtual Partitions takes place in almost the same manner as in non-vPars systems. The primary difference is that rather than run one backup on a physical system, you'll have to run a backup of every Virtual Partition that exists on your physical system. You'll also want to run a **make_net_recovery** on every Virtual Partition as well. This will create a bootable recovery archive for the Virtual Partition. This is covered in the Ignite/UX Chapter 9.

We'll perform a backup of the first vPar on a system that has the tape drive configured as one of its components (this is a backup to a local tape drive). We'll then perform a backup of a second vPar on the same computer to the tape drive that is configured on the first vPar (this is a backup to a remote tape drive). The tape drive is remote to the second vPar because it does not have the tape drive configured as one of its I/O components. Since the two vPars are distinct and independent systems, the tape drive on the first vPar is remote to the second vPar.

Backup to a Tape Drive on the Local Virtual Partition

Let's now backup a vPar called *cable1* to the local DDS drive at path *0.0.1.0.3.0.* This is a backup of a vPar to a local tape drive, meaning that the tape drive resource is tied to the vPar for which the backup will be performed. The following **vparstatus** shows that the Local Bus Adapter (LBA) at *0.0* is dedicated to *cable1*:

```
# vparstatus -p cable1 -v

[Virtual Partition Details]
Name:         cable1
State:        Up
Attributes:   Dynamic,Manual
Kernel Path:  /stand/vmunix
Boot Opts:

[CPU Details]
Min/Max:  1/3
Bound by User [Path]:
Bound by Monitor [Path]:  33
Unbound [Path]:  97

[IO Details]
   0.0                        <-- path of Core I/O card
   0.0.0.0
   0.0.1.1.2.0  BOOT
   0.0.4.0  CONSOLE

[Memory Details]
Specified [Base  /Range]:
          (bytes) (MB)
Total Memory (MB):  1024

#
```

Although the tape drive at *0.0.1.0.3.0* does not appear in this **vparstatus -v** output, the tape drive is connected to the Core I/O card at *0.0* and is therefore accessible to *cable1* to use as a backup device.

As a side note, if the tape interface were not on the Core I/O card, which is devoted to *cable1*, we could move the interface from *cable1* to a second vPar on the same computer called *cable2*, confirm that the driver is in **/stand/system**, and then backup *cable2*

Since there are many devices that you may want to move among the vPars running on your system you want to make sure that you have all of the device drivers required to support these devices in your kernel.

Let's now run **fbackup** with *u* to update the backup database, *0* for a full backup, *f* to specify the file to which we want to write the backup, *g* to specify our graph file, *I* to specify the name of the index file, and finally, we'll redirect messages to a file that will contain the backup log. We'll add the date and time to the end of the index and backup log files.

The following example shows the **fbackup** command issued:

```
fbackup -0u -f /dev/rmt/0m -g /tmp/backupgraph -I
         /tmp/backupindex.`date "+%y%m%d.%H:M"` 2>
         /tmp/backuplog.`date "+%y%m%d.%H:%M"`
```

Graph files are used to specify the files to be included in the backup. Index files contain a list of files produced as part of the backup. The graph file contains the files we wish to include (*i*) or exclude (*e*) as part of the backup. In our case the file contains only the following line:

```
i /var/tmp
```

Let's now see what files were produced as a result of having issued this command. First let's look at the backup index and backup log files:

[View full width]

```
# ls -l /tmp/backup*
-rw-rw-rw-   1 root   sys      11 Aug 29 14:45 backupgraph
-rw-------   1 root   sys     778 Aug 29 15:08 backupindex.010829.15:08
-rw-rw-rw-   1 root   sys     521 Aug 29 15:08 backuplog.010829.15:08
#



# cat /tmp/backupindex.010829.15:08
1024                    1 /
1024                    1 /var
1024                    1 /var/tmp
108                     1 /var/tmp/aaaa04686
1241                    1 /var/tmp/ems_inittab.old
28                      1 /var/tmp/envd.action2
28                      1 /var/tmp/envd.action5
4106                    1 /var/tmp/inetd.conf.old
96                      1 /var/tmp/ntp
0                       1 /var/tmp/rdskUBAa02185
0                       1 /var/tmp/rdskWAAa01840
310                     1 /var/tmp/sh1649.1
343                     1 /var/tmp/sh1649.2
166                     1 /var/tmp/sh1649.3
440                     1 /var/tmp/sh1649.4
611                     1 /var/tmp/swagent.log
0                       1 /var/tmp/sysstat_em.fmt
22734848                1 /var/tmp/vmunix.noreloc
#


# cat /tmp/backupgraph
i/var/tmp
#


# cat /tmp/backuplog.010829.15:08
fbackup(1417): cannot open the dates file /var/adm/fbackupfiles/dates for reading
fbackup(1004): session begins on Wed Aug 29 15:08:10 2001
fbackup(3203): volume 1 has been used 1 time(s)
fbackup(3024): writing volume 1 to the output file /dev/rmt/0m
fbackup(1423): WARNING: could not open the dates file /var/adm/fbackupfiles/dates for
➥ writing
fbackup(1030): warnings encountered during backup
fbackup(3055): total file blocks read for backup: 44432
#
```

The two files with the date appended (*8/29/01 time 15:08*) to the end of the filename were produced by the **fbackup** command issued earlier. The date appended to the end of the file can help in the organization of backup files. We could restore any or all of these files with **frestore**.

The backup of vPar *cable1* to the local tape drive is a backup of the system with a hostname of *cvhdcon3*. The other vPar running on this system, *cable2*, was not included in the backup because its host, *cvhdcon4,* is viewed as a separate system. In the next section we'll see how we would backup *cable2* to the tape drive dedicated to *cable1*.

Backup to a Tape Drive on a Different Local Virtual Partition

The tape drive in the preceding section was connected to *cable1* and used to backup *cable1*. Let's now backup *cable2* with the tape drive as part of the *cable1* Virtual Partition. Keep in mind that vPar *cable2* with its hostname *cvhdcon4* is a separate system from *cable1* and its host name *cvhdcon3*. We will, therefore, have to backup vPar *cable2* to a remote tape drive on *cable1*.

We will backup to a remote tape, connected to *cable1*, with **fbackup** by specifying the system name and tape drive, or file, to which we want to store the files. The following example uses **fbackup** options. We are connected to *cable2* and specify that it is to be backed up to the tape drive on *cable1* (hostname *cvhdcon3*.) The **fbackup** command below will perform the backup to the remote tape drive:

```
# fbackup -f cvhdcon3:/dev/rmt/0m -i /var/tmp -v


fbackup(1004): session begins on Wed Aug 29 16:09:28 2001
fbackup(3307): volume 1 has been used 6 time(s) (maximum: 100)
fbackup(3024): writing volume 1 to the output file cvhdcon3:/dev/rmt/0m
    1: /
    2: /tmp
    3: /tmp/.AgentSockets
    4: /tmp/.AgentSockets/A
    5: /tmp/X11_newfonts.log
    6: /tmp/install.vars
    7: /tmp/llbdbase.dat
    8: /tmp/lost+found
    9: /tmp/portmap.file
   10: /tmp/rpcbind.file
   11: /tmp/sd_ipd_acl.1417
   12: /tmp/services
   13: /tmp/swlist
   14: /tmp/typescript
fbackup(1005): run time: 17 seconds
fbackup(3055): total file blocks read for backup: 141
fbackup(3056): total blocks written to output file
cvhdcon3:/dev/rmt/0m: 0
#
```

It takes a little getting used to running **fbackup** to a remote device even when the device is physically connected to the computer on which you're working. Since the two vPars are different hosts, however, it makes sense that a tape drive used by a different vPar would indeed be remote.

The **fbackup** runs over the network from *cable2* to *cable1*. This is one of the many reasons that you want to have a network card for every vPar on your system.

We would issue other remote backup commands as well. The following example shows running **cpio** on *cable2* to the tape drive on *cable1* (host name *cvhdcon3*:)

```
# find . -print | cpio -oBv | (remsh cvhdcon3 dd of=/dev/rmt/0m)
.
lost+found
.AgentSockets
Socket <.AgentSockets/A> not backed up
typescript
install.vars
swlist
X11_newfonts.log
sd_ipd_acl.1417
services
rpcbind.file
portmap.file
llbdbase.dat
140 blocks

140+0 records in
140+0 records out
#
```

This command performs the backup of **/tmp** on vPar *cable2* by sending the files to the tape drive on system *cvhdcon3* (vPar *cable1*.)

Chapter 7. System Startup and Shutdown Scripts

Introduction

System Startup and Shutdown Mechanism

System Shutdown

System Startup and Shutdown Scripts in Virtual Partitions

Application Startup in Virtual Partitions

Virtual Partition Startup

Manual Pages for Commands Used in Chapter 7



Introduction

A variety of topics related to startup and shutdown scripts are covered in this chapter, including:

- 

The overall organization of the startup and shutdown mechanism in HP-UX
- 

Example of a startup file
- **/etc/inittab** file
- **shutdown** command
- Virtual Partitions (vPars) and startup



System Startup and Shutdown Mechanism

Startup and shutdown scripts for HP-UX 11.x are based on a mechanism that separates the actual startup and shutdown scripts from configuration information. In order to modify the way your system starts or stops, you don't have to modify scripts, which in general is considered somewhat risky; you can instead modify configuration variables. The startup and shutdown sequence is based on an industry standard that is similar to many other UNIX-based systems, so your knowledge of HP-UX applies to many other systems.

Startup and shutdown are going to become increasingly more important to you as your system administration work becomes more sophisticated. As you load and customize more applications, you will need more startup and shutdown knowledge. What I do in this section is give you an overview of startup and shutdown and the commands you can use to control your system.

The following components are in the startup and shutdown model:

Execution Scripts

Execution scripts read variables from configuration variable files and run through the startup or shutdown sequence. These scripts are located in **/sbin/init.d**.

Configuration Variable Scripts

These are the files you would modify to set variables that are used to enable or disable a subsystem or to perform some other function at the time of system startup or shutdown. These are located in **/etc/rc.config.d**.

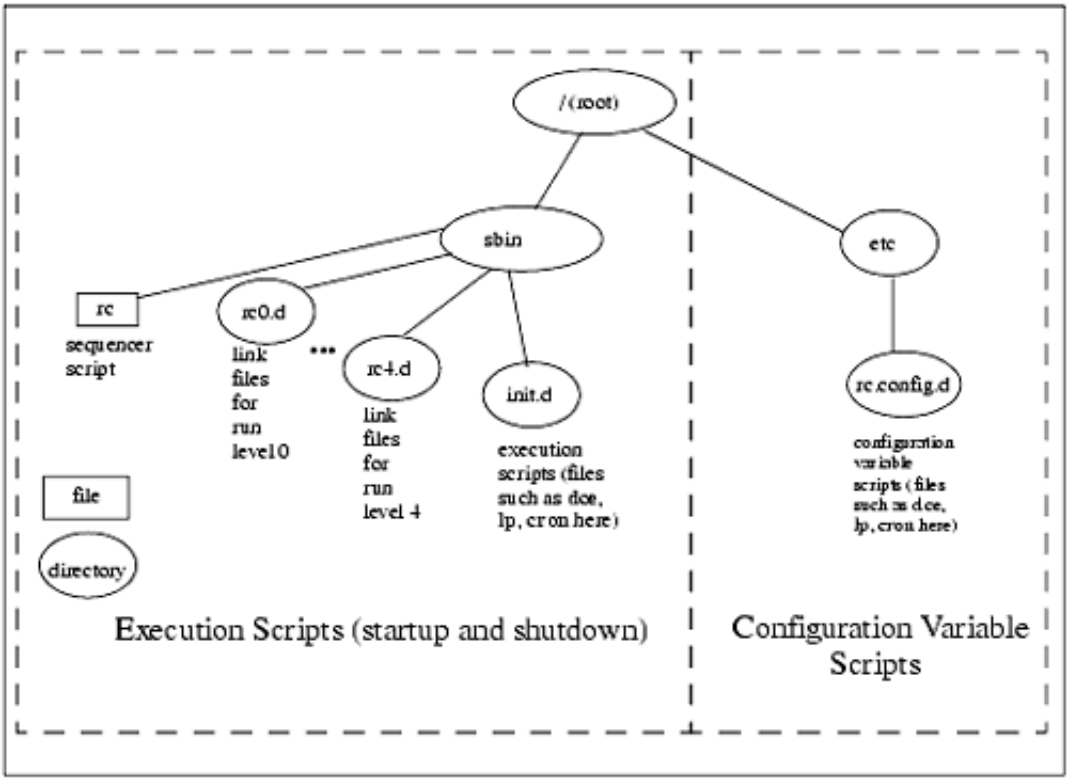
Link Files — These files are used to control the order in which scripts execute. These are actually links to execution scripts to be executed when moving from one run level to another. These files are located in the directory for the appropriate run level, such as **/sbin/rc0.d** for run level 0, **/sbin/rc1.d** for run level 1, and so on.

Sequencer Script

This script invokes execution scripts based on run-level transition. This script is located in **/sbin/rc**.

Figure 7-1 shows the directory structure for startup and shutdown scripts.

Figure 7-1. Organization of Startup and Shutdown Files



Execution scripts perform startup and shutdown tasks. **/sbin/rc** invokes the execution script with the appropriate start or stop arguments, and you can view the appropriate start or stop messages on the console. The messages you see will have one of the three following values:

OK	This indicates that the execution script started or shut down properly.
FAIL	A problem occurred at startup or shutdown.
N/A	The script was not configured to start.

In order to start up a subsystem, you would simply edit the appropriate configuration file in **/etc/rc.config.d**.

Let's take a look at an example startup and shutdown file for an application loaded onto an HP-UX system that is widely used for Internet applications called Cold Fusion. Like many applications loaded on HP-UX systems, Cold Fusion installs startup and shutdown scripts as a standard part of the installation of the product.

As mentioned earlier, the script used as part of the startup and shutdown process is in **/etc/init.d**. In our case, the name of the program is **/etc/init.d/coldfusion** and is shown in the following listing:

```
# cat /sbin/init.d/coldfusion
#!/bin/sh
#
# Start the Cold Fusion servers
#

# set at install
CFHOME=/apps/coldfusion
CFBIN=$CFHOME/bin

export CFHOME

#
# Start/stop processes for Cold Fusion
#

rval=0
```

```
case "$1" in

    start_msg)
        print "Starting ColdFusion Application Server"
        ;;

    stop_msg)
        print "Stopping ColdFusion Application Server"
        ;;

    'start')
        #First, check "on/off switch", to set CF_AUTOSTART, in config.d file.
        RCFILE=/etc/rc.config.d/coldfusion
        if [ -f $RCFILE ] ; then
                . $RCFILE
        else
                print "Warning: $RCFILE defaults file missing."
                print "         Starting ColdFusion by default."
                CF_AUTOSTART=1
        fi
        # Start CF if switch is on.
        if [ "$CF_AUTOSTART" -eq 1 ]; then
            if [ -x $CFBIN/start ]; then
                $CFBIN/start
                rval=$?
            else
                print "Error: ColdFusion startup script $CFBIN/start missing."
                print "       ColdFusion not started."
                rval=1
            fi
        else
            print "Notice: ColdFusion startup disabled in $RCFILE"
            rval=2
        fi
        ;;

    'stop')
        if [ -x $CFBIN/stop ]; then
                $CFBIN/stop -force
        fi
        ;;

    *)
        echo "Usage: $0 { start | stop }"
        rval=1
        ;;
esac

exit $rval

#
```

The startup and shutdown scripts in **/etc/init.d** generally perform both startup and shutdown functions. The startup and shutdown scripts, including the one in our example, recognize the following four arguments:

- *start_msg* - This is an argument passed to scripts so the that script can report a message indicating what the "start" action will do.
- *stop_msg* - This is an argument passed to scripts so that the script can report a message indicating what the "stop" action will do.
- *start* - The script will start the application.
- *stop* - The script will shut down the application.

You may encounter problems during boot with one of the startup programs being hung. If this is the case *Ctrl* | (*control* and *pipe*) keys will normally break out of the script and continue to the next script.

All startup and shutdown scripts, including the one in the previous listing, obtain configuration data from

variables in **/etc/rc.config.d**. Our example script checks the value of the "on/off" switch in **/etc/rc.confi.d/coldfusion**, which is shown in the following listing, to determine if Cold Fusion should be started:

```
# cat /etc/rc.config.d/coldfusion
# ColdFusion Application Server configuration file
#
CF_AUTOSTART=1  #Set to 1 to restart at boot time

#
```

The variable in this file is set to *1,* so the application will start at the time of system boot.

Startup and shutdown scripts are run based on the directory in which a link to the script appears. Our example script should be started at run level *3*. Therefore, a link to the script appears in the directory **/sbin/rc3.d**, shown as the third link in the following listing:

[View full width]

```
# ls -l /sbin/rc3.d
total 0
lrwxr-xr-x   1 root       sys     23 Apr 26 14:32 S100nfs.server -> /sbin/init.d/nfs.
➥ server
lrwxr-xr-x    1 root        sys       19 Apr 26 14:52 S200tps.rc -> /sbin/init.d/tps.rc
lrwxrwxrwx    1 root       sys      20 May 16 20:57 S790coldfusion -> ../init.d/coldfusion
lrwxr-xr-x   1 root       sys     23 Apr 26 14:44 S990dtlogin.rc -> /sbin/init.d/dtlogin.rc
#
```

We'll get to the significance of the naming of the link shortly. For the time being, it is sufficient to know that a link called **/sbin/rc3.d/S790coldfusion** points to our script **/init.d/coldfusion**.

Applications are shut down in the opposite order from which they were started. This means that a link to the startup and shutdown script will appear in a lower-level directory for shutdown. In our example, the startup link appears in **/sbin/rc3.d** but the shutdown link appears in **/etc/rc1.d**, as shown in the following listing:

```
# ls -l /sbin/rc1.d

lrwxr-xr-x    1 root       sys         17 Apr 26 14:52 K220slsd -> /sbin/init.d/slsd
lrwxr-xr-x    1 root       sys         18 Apr 26 14:45 K230audio -> /sbin/init.d/audio
lrwxr-xr-x   1 root      sys       21 Apr 26 14:46 K240auditing -> /sbin/init.d/auditing
lrwxr-xr-x    1 root       sys         17 Apr 26 14:43 K250envd -> /sbin/init.d/envd
lrwxr-xr-x    1 root       sys         17 Apr 26 14:43 K270cron -> /sbin/init.d/cron
lrwxr-xr-x    1 root       sys         15 Apr 26 14:45 K278pd -> /sbin/init.d/pd
lrwxr-xr-x    1 root       sys         15 Apr 26 14:45 K280lp -> /sbin/init.d/lp
lrwxr-xr-x   1 root      sys       21 Apr 26 14:49 K290hparamgr -> /sbin/init.d/hparamgr
lrwxr-xr-x   1 root       sys        20 Apr 26 14:43 K290hparray -> /sbin/init.d/hparray
lrwxrwxrwx   1 root      sys      20 May 16 20:57 K300coldfusion -> ../init.d/coldfusion
                    .
                    .
                    .
```

The link called **/sbin/rc3.d/K300coldfusion** points to our script **/init.d/coldfusion**. Startup for this application takes place at run level 3 and shutdown takes place at run level 1.

There is significance associated with the names of the links shown in the previous two listings. Let's take a look at the startup link in our example:

```
/sbin/rc3.d/S790coldfusion
        |   | |     |
        |   | |     v
        |   | |     script name - coldfusion in ex
        |   | v
        |   | sequence number - 790 in example
        |   v
        |  "S" for startup, "K" for shutdown
        v
        run level number - 3 in example
```

This example is for our Cold Fusion startup script. Startup links begin with an "S" for *startup*. The shutdown script has a similar entry in **/sbin/rc1.d**, it but has a "K" as the first character of the link name to indicate *kill*.

Scripts are executed in lexicographical order. Gaps are left between startup scripts at a given run level and

between shutdown scripts at a given run level, so when additional scripts are added, you don't have to renumber any existing scripts within a run level.

Because applications are shut down in the opposite order in which they are started, shutdown scripts do not usually have the same numbers as their startup counterparts. Two applications that start in a given order due to dependencies will usually be shut down in the opposite order in which they were started. In our example, the startup number is *S790coldfusion* and the shutdown number is *K300coldfusion*.



Scripts are run when there is a change in run level. **/sbin/rc** is a program that is run whenever there is a change in run level. The following listing shows **/etc/inittab**, which invokes **/sbin/rc** on the system used in our example:

[View full width]

```
init:3:initdefault:
ioin::sysinit:/sbin/ioinitrc >/dev/console 2>&1
tape::sysinit:/sbin/mtinit > /dev/console 2>&1
muxi::sysinit:/sbin/dasetup   </dev/console >/dev/console 2>&1 # mux init
stty::sysinit:/sbin/stty 9600 clocal icanon echo opost onlcr ixon icrnl ignpar </dev/
➥ systty
brc1::bootwait:/sbin/bcheckrc </dev/console >/dev/console 2>&1 # fsck, etc.
link::wait:/sbin/sh -c "/sbin/rm -f /dev/syscon; \
                  /sbin/ln /dev/systty /dev/syscon" >/dev/console 2>&1
cprt::bootwait:/sbin/cat /etc/copyright >/dev/syscon           # legal req
sqnc::wait:/sbin/rc </dev/console >/dev/console 2>&1           # system init
#powf::powerwait:/sbin/powerfail >/dev/console 2>&1           # powerfail
cons:123456:respawn:/usr/sbin/getty console console            # system console
#ttp1:234:respawn:/usr/sbin/getty -h tty0p1 9600
#ttp2:234:respawn:/usr/sbin/getty -h tty0p2 9600
#ttp3:234:respawn:/usr/sbin/getty -h tty0p3 9600
#ttp4:234:respawn:/usr/sbin/getty -h tty0p4 9600
#ttp5:234:respawn:/usr/sbin/getty -h tty0p5 9600
#ups::respawn:rtprio 0 /usr/lbin/ups_mond -f /etc/ups_conf
```



The **/sbin/rc** line is always present in the **/etc/inittab** file. There is more information about **/etc/inittab** coming shortly.

If you are booting your system to run level 3, then **/sbin/rc** will run the startup scripts present in **/sbin/rc1.d**, **/sbin/rc2.d**, and **/sbin/rc3.d**.

I have mentioned run levels several times in this discussion. Both the startup and shutdown scripts described here, as well as the **/etc/inittab** file, depend on run levels. In HP-UX 11.x, the following run levels exist:

0	Halted run level.
s	Run level s, also known as single-user mode, is used to ensure that no one else is on the system so that you can proceed with system administration tasks.
1	Run level 1 starts various basic processes.
2	Run level 2 allows users to access the system. This is also known as multi-user mode.
3	Run level 3 is for exporting NFS file systems.
4	Run level 4 starts the graphical manager, including HP Common Desktop Environment (HP CDE).
5 and 6	Not currently used.



/etc/inittab is also used to define a variety of processes that will be run, and it is used by **/sbin/init**. The **/sbin/init** process ID is 1. It is the first process started on your system and it has no parent. The **init** process looks at **/etc/inittab** to determine the run level of the system.

Entries in the **/etc/inittab** file have the following format:

id:run state:action:process

id	The name of the entry. The id is up to four characters long and must be unique in the file. If the line in **/etc/inittab** is preceded by a "#", the entry is treated as a comment.
run state	Specifies the run level at which the command is executed. More than one run level can be specified. The command is executed for every run level specified.
action	Defines which of 11 actions will be taken with this process. The 11 choices for action are: *initdefault*, *sysinit*, *boot*, *bootwait*, *wait*, *respawn*, *once*, *powerfail*, *powerwait*, *ondemand*, and *off*.
process	The shell command to be executed *if* the run level and/or action field so indicates.

Here is an example of an **/etc/inittab** entry:

```
cons:123456:respawn:/usr/sbin/getty  console console
 |   |   |         |
 |   |   |         |> process
 |   |   |> action
 |   |> run state
 |> id
```



This is in the **/etc/inittab** file, as opposed to being defined as a startup script, because the console may be killed and have to be restarted whenever it dies, even if no change has occurred in run level. **respawn** starts a process if it does not exist and restarts the process after it dies. This entry shows all run states, since you want the console to be activated at all times.

Another example is the first line from **/etc/inittab**:

init:3:initdefault:

The default run level of the system is defined as *3*.

The basics of system startup and shutdown described here are important to understand. You will be starting up and shutting down your system and possibly even modifying some of the files described here. Please take a close look at the startup and shutdown files before you begin to modify them.

Now let's take a look at the commands you can issue to shut down your system.

[Team LiB]

PREVIOUS NEXT

System Shutdown



What does it mean to shut down the system? Well, in its simplest form, a shutdown of the system simply means issuing the **/sbin/shutdown** command. The **shutdown** command is used to terminate all processing. It has many options, including the following:

-r	Automatically reboots the system; that is, brings it down and brings it up.
-h	Halts the system completely.
-y	Completes the shutdown without asking you any questions it would normally ask.
grace	Specifies the number of seconds you wish to wait before the system is shut down, in order to give your users time to save files, quit applications, and log out.

Here are some of the things your system does when you issue the **shutdown** command:

- Checks to see whether the user who executed **shutdown** does indeed have permission to execute the command.
- Changes the working directory to root (/).
- Sets *PATH* to **/usr/bin/:/usr/sbin:/sbin**.
- Updates all superblocks.
- Informs users that a **shutdown** has been issued and asks them to log out.
- Executes **/sbin/rc,** which does such things as shut down subsystems via shutdown scripts such as the spooler and CDE, unmount file systems and other such tasks.
- Runs **/sbin/reboot** if the *-r* option is used.

To shut down and automatically reboot the system, you would type:

```
$ shutdown -r
```

To halt the system, you would type:

```
$ shutdown -h
```

You will be asked whether you want to type a message to users informing them of the impending system shutdown. After you type the message, it is immediately sent to all users. After the specified time elapses (60 seconds is the default), the system begins the shutdown process. Once you receive a message that the system is halted, you can power off all your system components.

To shut down the system in two minutes without being asked any questions or sending any message, type:

```
$ shutdown -h -y 120
```



At times, you will need to go into single-user mode with **shutdown** to perform some task such as a backup or to expand a logical volume, and then reboot the system to return it to its original state.

To shut down the system into single-user mode, you would type:

```
$ shutdown
```

The **shutdown** command with no options puts you into single-user mode. On older versions of the operating system, you could go to single-user mode by using the **init** command with the ***s*** option (**init s**). However, this is highly discouraged because this command does not terminate other system activity nor does it log users off; therefore, it does not result in a true single-user state.



If the system is already in single-user mode or you like to live dangerously, you can execute **/usr/sbin/reboot**. I strongly suggest that you issue **shutdown,** which will call **reboot**. The **reboot** command abruptly terminates all processes and then halts or reboots the system. Also, with dynamically loadable kernel modules, **reboot** will not load these modules; only **shutdown** will.

Again, I recommend using the **shutdown** command, not **reboot**.

[Team LiB]

◄ PREVIOUS NEXT ►

System Startup and Shutdown Scripts in Virtual Partitions

Every Virtual Partition has separate startup and shutdown scripts just as you would have separate scripts on individual systems. There isn't any unique activity that takes place from a startup and shutdown standpoint due to Virtual Partitions; however, since each vPar is running its own HP-UX, you can customize the startup and shutdown to meet the needs of the applications you're running in each vPar. In addition, there is a *heartbeat* deamon that is used to synchronize the Virtual Partition database every few seconds.

Let's take a look at an application running in a vPar and the startup and shutdown procedures employed to support this application. In addition we'll view the means by which the *heartbeat* deamon is started and stopped to synchronize the Virtual Partition database.

Application Startup in Virtual Partitions

Each Virtual Partition is like a separate server. The applications running in a vPar run independently of applications and activity running in other vPars. All applications in a vPar require startup scripts to be run as they would on any system (please see the non-vPar-specific section of this chapter if you need information on how HP-UX startup and shutdown operates.)

In the vPar used in this example we have a variety of applications running, including Broadvision. Let's take a look at **/etc/rc.config.d**, where the configuration variable scripts are located, to see what *bv* files (those that relate to Broadvision) exist:

```
# ll | grep bv
-rwxr-xr-x   1 root       sys             9 Sep 26 09:31 bv
# cat bv
BV_CTL=1
#
```

This listing shows the **bv** file and its contents. The *BV_CTL=1* variable indicates that Broadvision is to be started when the system boots.

Starting the *bv* application processes results in several Broadvision-related daemons being started, including those shown in the following listing:

[View full width]

```
# ps -ef | grep bv1
pbcombv  3111 3096  0  Sep 26 ?  0:04 bvconf_srv p_1221_3 -f -install_name bv1to1/
➥ bvconf_srv_a
pbcombv  3404  3096  0  Sep 26  ?  1:19 cntdb p_1221_5 -install_name bv1to1/cntdb_1
pbcombv  3707  3096  0  Sep 26  ?  0:20 genericdb p_1221_8 -install_name bv1to1/
➥ genericdb_1
pbcombv 34793096 0 Sep26 ? 0:05 cntdb p_1221_7 -install_name bv1to1/
                                DiscussionForum_cntdb_1
root  7572  7232  0 13:17:33 pts/0     0:00 grep bv1
#
```

You can see that there are many processes related to the *bv1to1* application running on this system. These are started automatically as part of the startup structure of HP-UX. The following listing shows the link in **/sbin/rc1.d**, for run level one, and **/sbin/rc2.d**, for run level two, for *bv*.

```
# ll /sbin/rc1.d | grep bv
lrwxr-xr-x   1 root       sys            15 Sep 26 10:42 K105bv -> /sbin/init.d/bv
# ll /sbin/rc2.d | grep bv
lrwxr-xr-x   1 root       sys            15 Sep 26 10:41 S930bv -> /sbin/init.d/bv
#
```

The *bv kill* script, as indicated by the "K" preceding the link, is shown in **/sbin/rc1.d**, and the *start* script, as indicated by the "S" preceding the link, is shown in **/sbin/rc2.d**.

The script **/sbin/init.d/bv,** which is shown in the links, runs a variety of commands at both startup and shutdown of the Virtual Partition. The following listing shows the **/sbin/init.d/bv** script:

[View full width]

```
#
# <Insert comment about your script here>
#

# Allowed exit values:
#       0 = success; causes "OK" to show up in checklist.
#       1 = failure; causes "FAIL" to show up in checklist.
#       2 = skip; causes "N/A" to show up in the checklist.
#           Use this value if execution of this script is overridden
#           by the use of a control variable, or if this script is not
#           appropriate to execute for some other reason.
#       3 = reboot; causes the system to be rebooted after execution.
#       4 = background; causes "BG" to show up in the checklist.
#           Use this value if this script starts a process in background mode.
```

```
# Input and output:
#       stdin is redirected from /dev/null
#
#       stdout and stderr are redirected to the /etc/rc.log file
#       during checklist mode, or to the console in raw mode.

PATH=/usr/sbin:/usr/bin:/sbin
export PATH

# NOTE: If your script executes in run state 0 or state 1, then /usr might
#       not be available.  Do not attempt to access commands or files in
#       /usr unless your script executes in run state 2 or greater.  Other
#       file systems typically not mounted until run state 2 include /var
#       and /opt.

rval=0

# Check the exit value of a command run by this script.  If non-zero, the
# exit code is echoed to the log file and the return value of this script
# is set to indicate failure.

set_return() {
        x=$?
        if [ $x -ne 0 ]; then
                echo "EXIT CODE: $x"
                rval=1  # script FAILed
        fi
}

# Kill the named process(es).
# $1=<search pattern for your process>

killproc() {
        pid=`ps -el | awk '( ($NF ~ /'"$1"'/) && ($4 != mypid) && ($5 != mypid)  ){ print
➥ $4 }' mypid=$$ `
        if [ "X$pid" != "X" ]; then
                if kill "$pid"; then
                        echo "$1 stopped"
                else
                        rval=1
                        echo "Unable to stop $1"
                fi
        fi
}


case $1 in
'start_msg')
        # Emit a _short_ message relating to running this script with
        # the "start" argument; this message appears as part of the checklist.
        echo "Starting the BroadVision subsystem"
        ;;

'stop_msg')
        # Emit a _short_ message relating to running this script with
        # the "stop" argument; this message appears as part of the checklist.
        echo "Stopping the BroadVision subsystem"
        ;;

'start')

        # source the system configuration variables
        if [ -f /etc/rc.config.d/bv ] ; then
                . /etc/rc.config.d/bv
        else
                echo "ERROR: /etc/rc.config.d/bv defaults file MISSING"
        fi

        # Check to see if this script is allowed to run...
        if [ "$BV_CTL" != 1 ]; then
                rval=2
        else

        # Execute the commands to start your subsystem
```

```
        su - pbcombv -c "/opt/bv1to1/bin/bvconf execute"
        :
        fi
        ;;

'stop')
        # source the system configuration variables
        if [ -f /etc/rc.config.d/bv ] ; then
                . /etc/rc.config.d/bv
        else
                echo "ERROR: /etc/rc.config.d/bv defaults file MISSING"
        fi

        # Check to see if this script is allowed to run...
        if [ "$BV_CTL" != 1 ]; then
                rval=2
        else
        :
        # Execute the commands to stop your subsystem
        su - pbcombv -c "/opt/bv1to1/bin/bvconf shutdown"
        fi
        ;;

*)
        echo "usage: $0 {start|stop|start_msg|stop_msg}"
        rval=1
        ;;
esac

exit $rval
```

Note that all of the *start* and *stop* information related to Broadvision is present in this file because this script is run for both *kill* (*K*) and *start* (*S*). Note also the line

```
if [ "$BV_CTL" != 1 ]; then
```

which checks to see if the variable we viewed earlier in **/etc/rc.config.d/bv** is equal to one as it was when we checked the file.

The installation and startup of Broadvision is required in all Virtual Partitions in which you want to run this application. What is taking place in vPar used in this example is what you would see in every vPar running Broadvision.

[Team LiB]

PREVIOUS NEXT

Virtual Partition Startup

Let's take a look at **/etc/rc.config.d**, where the configuration variable scripts are located to see what vPar-related scripts exist (please see the non-vPar-specific section of this chapter if need information on how HP-UX startup and shutdown operates.) We'll perform a long listing of this directory and search for *vpar*:

```
# ll /etc/rc.config.d | grep vpar
-r--r--r--   1 bin        bin            291 Aug 21 14:38 vpard
-r--r--r--   1 bin        bin            399 Aug 13 14:31 vparhb
-r--r--r--   1 bin        bin            702 Aug 13 14:30 vparinit
#
```

This listing shows three vPar-related configuration variable scripts. We'll focus on **vparhb**, the *heartbeat* daemon.

The *heartbeat* deamon is used to synchronize the Virtual Partition database every few seconds among the vPars running on a server. Modifications you make related to Virtual Partitions update the Virtual Partition database **/stand/vpdb**. The following listing shows the contents of **/stand** in a Virtual Partition:

```
# ll /stand
total 143232
-rw-r--r--   1 root     sys          19 Jul 13 15:04 bootconf
drwxr-xr-x   4 root     sys        2048 Sep 17 15:47 build
drwxrwxrwx   5 root     root       1024 Sep 17 15:52 dlkm
drwxrwxrwx   5 root     sys        1024 Sep 17 14:10 dlkm.vmunix.prev
-rw-r--r--   1 root     sys        3388 Sep 26 13:01 ioconfig
-r--r--r--   1 root     sys          82 Jul 13 15:34 kernrel
drwxr-xr-x   2 root     sys        1024 Sep 26 13:04 krs
drwxr-xr-x   2 root     root       1024 Sep 26 13:01 krs_lkg
drwxr-xr-x   2 root     root       1024 Sep 26 13:04 krs_tmp
drwxr-xr-x   2 root     root       8192 Jul 13 15:04 lost+found
-rw-------   1 root     root         12 Sep 26 13:01 rootconf
-r--r--r--   1 root     sys        2035 Sep 17 14:09 system
-r--r--r--   1 root     sys         994 Jul 13 15:28 system.01
-r--r--r--   1 root     sys         999 Jul 13 15:56 system.02
-r--r--r--   1 root     sys         994 Jul 13 15:28 system.base
drwxr-xr-x   2 root     sys        1024 Jul 13 15:37 system.d
-r--r--r--   1 root     sys        2030 Sep 17 14:07 system.prev
-rwxr-xr-x   1 root     root   22682568 Sep 17 15:52 vmunix
-rwxr-xr-x   1 root     sys    21916712 Sep 17 14:10 vmunix.prev
-rw-------   1 root     root       8232 Sep 26 13:01 vpdb
-rw-------   1 root     root       8232 Jul 17 14:11 vpdb.OLD
-r-xr-xr-x   1 bin      bin      837616 Aug 31 18:59 vpmon
-rw-------   1 root     root    5078504 Jul 18 11:36 vpmon.dmp
#
#
```

Among the files shown in this listing is **vpdb**. If this **vpdb** were to be modified, or another **vpdb** on this server were to be modified, the *heartbeat* daemon **vphb** would ensure that all of the databases on the server in all of the running vPars would be synchronized. The non-running vPars cannot be synchronized until they are started.

The *heartbeat* daemon **vphb** is shown in the following **ps -ef** listing:

```
# ps -ef | grep vphb
    root   352     1  0 13:01:53 ?         0:00 vphbd -d 10 -p /var/run/vphbd.pid
    root  7289  7232  1 12:18:28 pts/0     0:00 grep vphb
#
```

vphb is started automatically as part of the startup structure of HP-UX. The following listing shows all of the links in **/sbin/rc0.d**, for run level *0*, including the two vPar-related links at the very beginning of this listing:

```
# ll /sbin/rc0.d
total 0
lrwxr-xr-x   1 bin        bin        18 Sep 17 14:11 K425vpard -> /sbin/init.d/vpard
lrwxr-xr-x   1 bin        bin        19 Sep 17 14:11 K431vparhb -> /sbin/init.d/vparhb
lrwxr-xr-x   1 root       root       19 Jul 13 15:08 K480syncer -> /sbin/init.d/syncer
lrwxr-xr-x   1 root       root       15 Jul 13 15:08 K650kl -> /sbin/init.d/kl
```

```
lrwxr-xr-x   1 root       root    20 Jul 13 15:08 K800killall -> /sbin/init.d/killall
lrwxr-xr-x   1 root       root    19 Jul 13 15:08 K888kminit -> /sbin/init.d/kminit
lrwxr-xr-x   1 root       root    20 Jul 13 15:08 K890kmbuild -> /sbin/init.d/kmbuild
lrwxr-xr-x   1 root    root   23 Jul 13 15:08 K900localmount -> /sbin/init.d/localmount
#
```

These are *kill* scripts, as indicated by the "K" preceding each link. The links to the *start* scripts are found in **/sbin/rc1.d**.

We'll take a look at the file **/sbin/init.d/vparhb**, which is shown in the *heartbeat* link and that is run at startup in the following listing:

```
# cat /sbin/init.d/vparhb
#!/sbin/sh

#
# NOTE: This script is not configurable!  Any changes made to this
#       script will be overwritten when you upgrade to the next
#       release of HP-UX.
#

#
# vphbd startup: Startup and kill script for the virtual partition
#                heartbeat daemon
#

PATH=/sbin:/usr/sbin:/usr/bin
export PATH

if [ -r /etc/rc.config.d/vparhb ]
then . /etc/rc.config.d/vparhb
fi

case "$1" in

    'start_msg') echo "Starting Virtual Partition Heartbeat Daemon" ;;

    'start')
        vphbd -d "${VPHBD_DELAY-10}" -p "${VPHBD_PID_FILE-/var/run/vphbd.pid}"
        exit $?
        ;;

    'stop_msg') echo "Stopping Virtual Partition Heartbeat Daemon" ;;

    'stop')
        [ ! -r "${VPHBD_PID_FILE=/var/run/vphbd.pid}" ] && exit 2
        pid=`cat "$VPHBD_PID_FILE"`
        [ "$pid" -le "0" ] && exit 1
        kill "$pid"
        rm -f "$VPHBD_PID_FILE"
        exit 0
        ;;

    *)
        echo "Usage: $0 { start | start_msg | stop | stop_msg }"
        ;;

esac

exit 0
#
```

Note that all of the *start* and *stop* information related to the *heartbeat* daemon is present in this file because this script is run for both *kill* (*K*) and *start* (*S*.)

The vPar-specific startup and shutdown setup, such as the *heartbeat* daemon we just covered, are automatically performed for you when vPars software is installed. Your application-related startup configuration must be performed on each vPar just as it would on separate servers.

[Team LiB]

Manual Pages for Commands Used in Chapter 7

There are manual pages in Appendix A for the following commands used in this chapter:

inittab
shutdown



Chapter 8. Users and Groups

Set Up Users and Groups

Assigning Users to Groups

Virtual Partitions and Users

NIS for Managing Users



Set Up Users and Groups

In this chapter we'll cover the basics of users on HP-UX systems. Topics such as users, groups, permissions, Virtual Partitions (vPars) and users, and NIS will be covered.

You need to make a few basic decisions about users. Where should users' data be located? Who needs to access data from whom, thereby defining "groups" of users? What kind of particular startup is required by users and applications? Is there a shell that your users prefer?

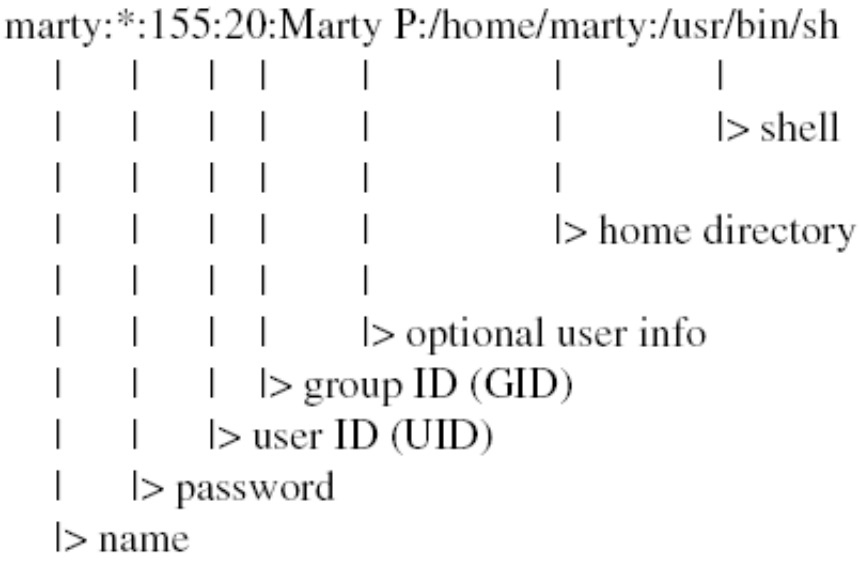
You will want to put some thought into these important user-related questions. I spend a lot of time working with my customers, rearranging user data, for several reasons. It doesn't fit on a whole disk (for this reason, I strongly recommend using Logical Volume Manager or Veritas Volume Manager); users can't freely access one another's data, or even worse, users *can* access one another's data too freely.

We will consider these questions, but first, let's look at the basic steps to adding a user. Here is a list of activities:

- Select a user name to add
- Select a user ID number
- Select a group for the user
- Create an **/etc/passwd** entry
- Assign a user password (including expiration options)
- Select and create a home directory for user
- Select the shell the user will run (I strongly recommend the default POSIX shell)
- Place startup files in the user's home directory
- Test the user account

Most of what you do is entered in the **/etc/passwd** file, where information about all users is stored. You can make these entries to the **/etc/passwd** file with the **/usr/sbin/vipw** command. Figure 8-1 is a sample **/etc/passwd** entry.

Figure 8-1. Sample /etc/passwd Entry



Here is a description of each of these fields:

name. The user name you assign. This name should be easy for the user and other users on the system to remember. When sending electronic mail or copying files from one user to another, the easier it is to remember the user name, the better. If a user has a user name on another system, you may want to assign the same user name on your HP-UX system. Some systems don't permit nice, easy user names, so you may want to break the tie with the old system and start using sensible, easy-to-remember user names on your HP-UX system. Remember, no security is tied to the user name; security is handled through the user's password and the file permissions.

password. This is the user's password in encrypted form. If an asterisk appears in this field, the account can't be used. If it is empty, the user has no password assigned and can log in by typing only his or her user name. I strongly recommend that each user have a password that he or she changes periodically. Every system has different security needs, but at a minimum, every user on every system should have a password. When setting up a new user, you can force the user to create a password at first login by putting **,..** in the password field.

Some features of a good password are:

- Use a minimum of six characters which should include special characters such as a slash (/), a dot (.), or an asterisk (*).
- No words should be used for a password.
- Don't make the password personal such as name, address, favorite sports team, etc.
- Don't use something easy to type such as 123456, or qwerty.
- Some people say that misspelled words are acceptable, but I don't recommend using them. Spell-check programs that match misspelled words to correctly spelled words can be used to guess at words that might be misspelled for a password.
- A password generator that produces an intelligible passwords works the best.

user ID (UID). The identification number of the user. Every user on your system should have a unique UID. I would recommend that you reserve UIDs less than 100 for system-level users.

group ID (GID). The identification number of the group. The members of the group and their GID are in the **/etc/group** file. You can change the GID assigned if you don't like it, but you may also have to change the GID of many files. As a user creates a file, his or her UID is assigned to the file as well as the GID. This means that if you change the GID well after users of the same group have created many files and directories, you may have to change the GID of all these elements. I usually save GIDs of less than 10 for system groups.

optional user info. In this space, you can make entries, such as the user's phone number or full name. You can leave this blank, but if you manage a system or network with many users, you may want to add the user's full name and extension so that if you need to get in touch with him or her, you'll have the information at your fingertips. (This field is sometimes referred to as the GECOs field.)

home directory. The home directory defines the default location for all the users' files and directories. This is the present working directory at the time of login.

shell. This is the startup program the user will run at the time of login. The shell is really a command interpreter for the commands the user issues from the command line. I recommend using the default POSIX shell (**/usr/bin/sh**), but there are also three traditional, popular shells in the HP-UX environment: the C shell (**/usr/bin/csh**); Bourne shell (**/usr/old/bin/sh**); and Korn shell (**/usr/bin/ksh**). Shell programming for the system administrator is covered in Chapter 28.

The location of the user's home directory is another important entry in the **/etc/passwd** file. You have to select a location for the user's "home" directory in the file system where the user's files will be stored. With some of the advanced networking technology that exists, such as NFS, the user's home directory does not even have to be on a disk that is physically connected to the computer that he or she is using! The traditional place to locate a user's home directory on an HP-UX system is the **/home** directory in HP-UX 11.x.

The **/home** directory is typically the most dynamic area in terms of growth. Users create and delete files in their home directory on a regular basis. This means that you have to do more planning related to your user area than in more static areas, such as the root file system and application areas. You would typically load HP-UX and your applications and then perform relatively few accesses to these in terms of adding and deleting files and directories. The user area is continuously updated, making it more difficult to maintain.

[Team LiB]



Assigning Users to Groups

After defining all user-related information, you need to consider groups. Groups are often overlooked in the HP-UX environment until the system administrator finds that all his or her users are in the very same group, even though from an organizational standpoint, they are in different groups. Before I cover the groups in general, let's look at a file belonging to a user and the way access is defined for a file:

```
$ ll
-rwxr-x--x   1 marty     users     120 Jul 26 10:20 sort
```

For every file on the system, HP-UX supports three classes of access:

- User access (u). Access is granted to the owner of the file
- Group access (g). Access is granted to members of the same group as the owner of the file
- Other access (o). Access is granted to everyone else

These access rights are defined by the position of r (read), write (w), and execute (x) when the long listing command is issued. For the long listing (**ll**) above, you see the permissions in Table 8-1.

Table 8-1. Long Listing Permissions

Access	User Access	Group Access	Other
Read	r	r	-
Write	w	-	-
Execute	x	x	x

You can see that access rights are arranged in groups of three. Three groups of permissions exist with three access levels each. The owner, in this case, marty, is allowed read, write, and execute permissions on the file. Anyone in the group users is permitted read and execute access to the file. Others are permitted only execute access to the file.

These permissions are important to consider as you arrange your users into groups. If several users require access to the same files, then you will want to put those users in the same group. The trade-off here is that you can give all users within a group rwx access to files, but then you run the risk of several users editing a file without other users knowing it, thereby causing confusion. On the other hand, you can make several copies of a file so that each user has his or her personal copy, but then you have multiple versions of a file. If possible, assign users to groups based on their work.

The **/etc/group** file contains the group name, an encrypted password (which is rarely used), a group ID, and a list of users in the group. Here is an example of an **/etc/group** file:

```
root::0:root
other::1:root, hpdb
bin::2:root,bin
sys::3:root,uucp
adm::4:root,adm
daemon::5:root,daemon
mail::6:root
lp::7:root,lp
tty::10:
nuucp::11:nuucp
military::25:jhunt,tdolan,vdallesandro
commercial::30:ccascone,jperwinc,devers
nogroup:*:-2:
```

This **/etc/group** file shows two different groups of users. Although all users run the same application, a desktop publishing tool, some work on documents of "commercial" products while others work on only "military" documents. It made sense for the system administrator to create two groups, one for commercial document preparation and the other for military document preparation. All members of a group know what documents are current and respect one another's work and its importance. You will have few problems among group members who know what the other members are doing and you will find that these members don't delete files that shouldn't be deleted. If you put all users into one group, however, you may find that you spend more time restoring files, because users in this broader group don't find files that are owned by other members of their group to be important. Users can change group with the **newgrp** command.

[Team LiB]

PREVIOUS NEXT

Virtual Partitions and Users

Each Virutal Partition is like a separate server. This means that you have separate users in your respective Virtual Partitions. Any user-related information that needs to be shared between Virtual Partitions would be done in the same manner as sharing between physically separate servers. For instance, you may want to have users have the capability to remotely login from one Virtual Partition to another. You would accomplish this by setting up the appropriate network files in each Virtual Partition just as you would with separate servers.

The vPars are isolated from a software perspective, so it is just as if the users exist on different servers. In order to share information among users, such as a database of users, you would implement technology such as Network Information System (NIS). With NIS you could create a database of user information that would be shared among different systems, and in this case, different vPars because they are the same as different servers from a software standpoint.

Let's now create a new user on each of the two vPars. On *cable1* we'll create the user *cable1us* and verify that this user does not exist on *cable2*. On *cable2* we'll create the user *cable2us* and verify that this user does not exist on *cable1*.

We've created the user *cable1us* in vPar *cable1* and it exists only in *cable1*. We'll verify this shortly. We've also created a user *cable2us* in vPar *cable2.*

There should now be one user in *cable1* that is not viewable in *cable2,* and vice versa. The following shows connecting to *cable1* and listing the contents of **/etc/passwd**.

```
cvhdcon3:/ # cat /etc/passwd
root:hiIXKsAzUIFy6:0:3::/:/sbin/sh            cable1 listing
daemon:*:1:5::/:/sbin/sh
bin:*:2:2::/usr/bin:/sbin/sh
sys:*:3:3::/:
adm:*:4:4::/var/adm:/sbin/sh
uucp:*:5:3::/var/spool/uucppublic:/usr/lbin/uucp/uucico
lp:*:9:7::/var/spool/lp:/sbin/sh
nuucp:*:11:11::/var/spool/uucppublic:/usr/lbin/uucp/uucico
hpdb:*:27:1:ALLBASE:/:/sbin/sh
nobody:*:-2:-2::/:
www:*:30:1::/:
webadmin:*:40:1::/usr/obam/server/nologindir:/usr/bin/false
smbnull:*:101:101:DO NOT USE OR DELETE - needed by Samba:
                                      /home/smbnull:/sbin/sh
opc_op:*:777:77:OpC default operator:/home/opc_op:/usr/bin/ksh
cable1us::102:20:,,,:/home/cable1us:/usr/bin/sh     <-- cable1us
cvhdcon3:/ #
```

This listing for *cable1* shows that *cable1us* exists in the **/etc/passwd** file on *cable1*; however, there is no *cable2us* present in this file.

The following shows connecting to *cable2* and listing the contents of **/etc/passwd**.

```
# cat /etc/passwd                        cable2 listing
root:Jx66ARmhj.aBs:0:3::/:/sbin/sh
daemon:*:1:5::/:/sbin/sh
bin:*:2:2::/usr/bin:/sbin/sh
sys:*:3:3::/:
adm:*:4:4::/var/adm:/sbin/sh
uucp:*:5:3::/var/spool/uucppublic:/usr/lbin/uucp/uucico
lp:*:9:7::/var/spool/lp:/sbin/sh
nuucp:*:11:11::/var/spool/uucppublic:/usr/lbin/uucp/uucico
hpdb:*:27:1:ALLBASE:/:/sbin/sh
nobody:*:-2:-2::/:
www:*:30:1::/:
webadmin:*:40:1::/usr/obam/server/nologindir:/usr/bin/false
sam_exec:xxx:0:1::/home/sam_exec:/usr/bin/sh
cable2us::101:20:,,,:/home/cable2us:/usr/bin/sh     <-- cable2us
#
```

This listing for *cable2* shows that *cable2us* exists in the **/etc/passwd** file on *cable2*; however, there is no *cable1us* present in this file

This listing shows that the unique users for the respective vPars exist only on the vPar on which they were created. In addition, each vPar has its own *root* and other system-level users.

Next let's look at setup information of individual users in vPars.

Individual User Setup

As far as setup for users is concerned, you perform setup in a vPars environment just as you would in a non-vPars environment. The prompt, for instance, might include the host name of your system. On a system where the host name and vPar name are different, you still probably want your prompt to include the hostname. You can, however; also include the vPar name at the prompt if you wish. The following is the section of **/.profile** that includes some minor customization for the user *root*:

```
# Added for prompt, history etc.

        HISTFILE=~/.sh_history;                 export HISTFILE
        EDITOR=vi;                              export EDITOR
        PS1="`whoami`@`hostname`"'[${PWD}] > '; export PS1
```

This is the information I typically add for *root* to provide history file information and the prompt. I included the host name here, which would result in a prompt that includes the user's name, host name, and present working directory, as shown below:

```
root@actappd1[/.root] >
```

If your users need to see the vPar name, this can be added to or substituted for the hostname in **.profile**. This is another example of the advantage to having your hostname and vPar name identical. It may be; however, that your hostname is very confusing, and substituting a simple vPar name in the prompt may be advantageous to users. In either case, your user setup does not change with vPars, since vPars are the same as separate systems from a user perspective.

Application Users

Users that are required for specific applications need to be set up on all vPars on which an application is going to run. Since vPars are like individual systems, you'll put the users, including application users, on the vPars where you need them and not on others. On a system running Broadvision and Oracle, for instance, you would set up the users required for those applications. The following shows an **/etc/passwd** output on a system running these two applications:

```
# cat /etc/passwd
root:9Obj3Cya98pFo:0:3::/.root:/sbin/sh
daemon:*:1:5::/:/sbin/sh
bin:*:2:2::/usr/bin:/sbin/sh
sys:*:3:3::/:
adm:*:4:4::/var/adm:/sbin/sh
uucp:*:5:3::/var/spool/uucppublic:/usr/lbin/uucp/uucico
lp:*:9:7::/var/spool/lp:/sbin/sh
nuucp:*:11:11::/var/spool/uucppublic:/usr/lbin/uucp/uucico
hpdb:*:27:1:ALLBASE:/:/sbin/sh
nobody:*:-2:-2::/:
www:*:30:1::/:
webadmin:*:40:1::/usr/obam/server/nologindir:/usr/bin/false
smbnull:*:101:101:DO NOT USE OR DELETE - needed by Samba:/home/smbnull:/sbin/sh
opc_op:*:777:77:OpC default operator:/home/opc_op:/usr/bin/ksh
stssmrp:HFQrTOZx920Fg:1108:20:Martin Paul:/home/stssmrp:/bin/ksh
stssmrpr:pugxhblYiahZI:0:3:Martin Paul:/home/stssmrp:/bin/ksh
stssjtf:VammVKnIwD/T.:1185:20:John Fontanilla:/home/stssjtf:/usr/bin/ksh
pbcombv:UzGN5gDTNfgQM:102:20:PB.COM User:/home/pbcombv:/usr/bin/ksh     <-- BV user
oracle:11Z/oQQfEHzOo:1012:111:Oracle User:/home/oracle:/bin/ksh     <-- Oracle user
#
```

Note the last two users in the file for Broadvision (*pvcombv*) and Oracle (*oracle*) with their respective home directories. With these two applications running on the Virtual Partition, the appropriate users had to be set up.

The remainder of this chapter covers background information related to users.

[Team LiB]

NIS for Managing Users

One of the most popular ways to manage user-related information in a distributed environment is the Network Information System (NIS). NIS provides a method for multiple systems to share a centralized database of password, group, and other optional databases such as services and/or hosts. By doing so, the administration of user accounts is simplified for both end-users and system administrators. You will also often hear NIS referred to as "Yellow Pages" or "YP"; in fact, most of the NIS commands begin with the letters "yp".

NIS and other such technologies are required only in a distributed environment where users have to be managed on many systems. This is because changes in information, such as adding and removing users, must be disseminated to many systems. This is normally not required in a centralized environment, where only a small number of copies of such information must be maintained.

NIS on HP-UX is interoperable with Solaris and Sun-licensed NIS implementations, including Linux. Like most of these implementations, HP's NIS is not implemented in C2 or Trusted System mode. Additionally, Microsoft-based operating systems such as Windows do not use or interoperate with NIS.

What Does NIS Manage?

NIS can manage many different databases. We'll focus on the user password and group information. Table 8-2 is a list of some of the databases that NIS can manage:

Table 8-2. Some NIS Databases That NIS Can Manage

Filename	Information Contained in File
/etc/passwd	Usernames, user IDs primary groups, and encrypted passwords.
/etc/group	User group memberships.
/etc/hosts	Hostnames and IP addresses.
/etc/services	Network port numbers and service names.
/etc/aliases	Aliases and mailing lists for the mail system.
/etc/netgroup	Netgroup definitions.
/etc/rpc	Remote procedure call program numbers.
/etc/protocols	Network protocol names and numbers.
	The following are optional for HP, but not used by HP-UX NIS clients.
/etc/bootparms	Information about diskless nodes.
/etc/ethers	Ethernet numbers (MAC addresses).
/etc/netmasks	Network masks.

NIS also calls each of these databases a *map*. They are called maps because NIS allows you to map a key, such as a user name, to a value field, such as the user's **passwd** entry from the **passwd** map on the NIS Master Server.

How Do I Plan For NIS?

NIS requires one NIS Master Server and typically at least one NIS Slave Server per IP subnet. A NIS Master

or Slave Server answers requests from NIS clients typically seeking user password information when a user login occurs. NIS is designed in a *top-down* or *hierarchical* manner, with all changes being made through the NIS Master Server. When a change is made to the NIS Master, the changes can be made visible to the NIS Slaves by "pushing" the updated database to the Slaves. Generally, an NIS server should satisfy the demands of 25 to 50 NIS clients. NIS Masters and Slaves are typically also NIS clients.

On the NIS Master server, you need to decide where you want to keep the NIS database "source" files. Most typically, the **/etc** files are used on the NIS Master for all databases except passwords, which are generally put into an "alternate" or "private" password file such as **/etc/passwd.nis**. Put only the user password database there and not the "system" users, such as root, sys, bin and so on. The "system" users should always be put in **/etc/passwd** on any NIS server or client.

NIS will provide a small additional network and system load on an NIS Slave and Master server. Most typically, this extra load is encountered when updating a map or database and pushing the changes to the Slave servers.

The design of NIS requires that you first configure the NIS Master Server, then the NIS Slave Servers, and finally, the NIS clients.

How Do I Configure an NIS Master Or Slave Server?

You can configure an NIS Master and Slave Server either by using the management GUI or by performing the process manually. If you use the GUI to perform this process most of the work will take place for you in the background. You enter the pertinent information and the GUI performs the configuration. You perform the steps in the manual procedure. There are many good documents that can help you in this configuration, including HP's *Installing and Administering NFS Services* manual. This manual covers configuring Master and Slave Servers and can be obtained from www.docs.hp.com. HP support likes you to follow these step-by-step procedures so that if you encounter a problem, your execution of the steps can be reviewed.

To configure with SAM, select *Networking/Communications*, then *NIS*. You will first be prompted to specify the NIS domain name. After that, you can proceed to the *Enable NIS Master Server* or *Enable NIS Slave Server* menus.

How Do I Configure an NIS Client?

You can also configure an NIS client by either using SAM or by performing the process manually. Once again, the *Installing and Administering NFS Services* manual provides an excellent step-by-step procedure for configuring the client.

To configure with SAM, select *Networking/Communications*, then *NIS*. You will first be prompted to specify the NIS domain name. After that you can proceed to the *Enable NIS Client* menu.

In either case, I have two additional tips. First, HP supplies the following **/etc/nsswitch.compat** file as a template to copy into **/etc/nsswitch.conf**. This allows you to use the "+" and "-" syntax in **/etc/passwd** and **/etc/group**:

```
# /etc/nsswitch.compat:
#
# An example file that could be copied over to
# /etc/nsswitch.conf; it
# uses NIS (YP) in conjunction with files. #

passwd:       compat
group:        compat
hosts:        nis [NOTFOUND=return] files
networks:     nis [NOTFOUND=return] files
protocols:    nis [NOTFOUND=return] files
rpc:          nis [NOTFOUND=return] files
publickey:    nis [NOTFOUND=return] files
netgroup:     nis [NOTFOUND=return] files
automount:    files nis
aliases:      files nis
services:     files nis
```

Second, you may prefer DNS over NIS to manage the hosts database and use the following "hosts" entry in **/etc/nsswitch.conf**:

```
hosts:        files [NOTFOUND=continue] dns ...
```

or

[View full width]

```
hosts:        dns   [NOTFOUND=continue,UNAVAILABLE=continue] files [NOTFOUND=continue,
➥ UNAVAILABLE=continue] nis
```

How Do I Maintain My NIS Environment?

The most common user activity is changing a user password. A user can use either the **passwd -r nis** or **yppasswd** command to do this. This will prompt for the old NIS password, the new password, make the change on the NIS server, and by default, re-make the NIS map and push it out to all the Slave Servers.

A system administrator can change user passwords either with SAM or with shell scripts or commands such as **passwd -r nis** <*username*> or **yppasswd** <*username*>. You can also use SAM to add new users, or you can do this with shell scripts or commands.

Often, when a user changes a database file by editing with **vi**, the changes need to be compiled into the NIS maps (called "making" a map) and pushing the maps out to the Slave Servers if the "make" does not do this. For example, after modifying some users' home directories in **/etc/passwd.nis**, the system administrator needs to:

```
1. cd /var/yp   # Change to directory of NIS Makefile
2. make passwd  # "make" or compile the passwd map
3. yppush passwd (if step #2 did not push to the NIS slaves)
```

Here is a tip that applies to HP-UX and to any NIS vendor's implementation with group files: There are times when users are members of multiple groups, producing lines in **/etc/group** that are longer than the NIS limitation of 1024 characters per line. To work around this, use different group names for the same GID, for example:

```
102 support:brian,sam,charlie
102 support1:bill,julie,maria
```

Note that you don't need to specify a user's primary group membership in **/etc/group** since that is already specified by their GID in **/etc/passwd**. If users are members of multiple groups, you only need to put their user name in **/etc/group** entries for their secondary groups.

NIS provides a centralized database scheme for managing user password and group information. Administration techniques used for NIS setup and administration are well documented, integrated into SAM, and are generally interoperable with other NIS implementations.

[Team LiB]



Chapter 9. Ignite-UX

Ignite-UX Overview

Set up Server with Depots and Install Clients

Run Ignite-UX GUI

System Recovery with Ignite-UX

System Recovery to Local Tape Drive with make_tape_recovery

Creating Bootable Recovery Archives with make_net_recovery

Using Ignite-UX GUI to Create a vPar Archive

Manual Pages for Commands Used in Chapter 9



Ignite-UX Overview

Ignite-UX is quite a versatile product. It has a great deal of functionality associated with it that would take an entire book to cover. I'll cover what I believe to be the two major functional areas of Ignite-UX that I see used. These are:

- Network installation of client, or target, systems from a server that has on it software depots. These depots are usually created by copying software from media into depots on the Ignite-UX server. This is somewhat the same as the software installation covered in Chapter 2, except that the software is on a server rather than media. There is also a Graphical User Interface that allows you to control this process from the server. This technique would allow you to select the specific software that you want to load on each individual client. You can either "push" software from the server to the client or "pull" software to the client from the server. In addition to creating software depots from media, you can create an operating system archive from a "golden image." The "golden image" is a perfectly running system that you wish to replicate to many systems. The **make_sys_image** command produces a compressed image of the "golden system." In this chapter we'll focus on the software depots created from copying media into depots on the server.

- Creating a bootable system recovery tape using **make_tape_recovery**. This feature allows you to create an image of your root volume on tape. You could later boot off of this tape and restore the root volume group and other volume groups directly from tape.

- Creating a network recovery archive using **make_net_recovery**. This allows you to have a centralized system on which all of your recovery archives are stored. You could later boot off of this network archive and restore your system.

Let's take a look at these three aspects of Ignite-UX in the upcoming sections. Be sure to check www.docs.hp.com to obtain some good documents available on Ignite-UX and www.software.hp.com to obtain the latest Ignite-UX software. Some of these documents have much more detail than I include in my upcoming "how to" sections.

I'll assume that you already have the Ignite-UX product loaded on your server system. This software comes bundled with HP-UX 11i, so it is just a matter of loading it on your server system from media. You can be sure you're getting the latest version of Ignite-UX if you download it from the software.hp.com Web site.

Set up Server with Depots and Install Clients

You can use most any system as your Ignite-UX server. You can have depots on your server for HP-UX 11i, 11.0, and 10.20. I would recommend using an 11.00 or 11i system as an Ignite-UX server because I have seen problems with a 10.20 system acting as a server for loading clients with 11.00 and 11i. I have never seen a problem with an 11.0 or 11i system acting as a server for loading clients with any version of HP-UX. In general, though, the server acts as a host for depots and its operating system is not an integral part of loading software onto clients. Ignite-UX is a versatile product. To demonstrate its flexibility, we'll use an 11.0 Ignite-UX server in this section to load 11.0 software to clients. We can create 11i and 10.20 depots as well. Although we'll use 11.0 in this section, the procedure for 11i is identical. Later in the chapter we'll recover Virtual Paritions (vPars) systems running 11i with Ignite-UX. We'll use a workstation (model J2240) running 11.00 in our examples in this section. From the 11.00 depot we create on this system, we can load a variety of clients. The fact that we're using a workstation rather than a server doesn't matter as far as loading clients is concerned. I find that many system administrators use a workstation as an Ignite-UX server even though they're loading primarily servers, such as N-Class systems.

The software depots you create on your Ignite-UX server consume a lot of disk space. A core operating system with many applications could take up several GBytes of disk space. I like to devote several GBytes of disk space to Ignite servers so I don't have to increase the size of **/var** (the default location for software depots) when additional depots are loaded on a system. My rule of thumb is **/var** at 6 GBytes to support several depots.

Just to give you an idea of the disk space consumed by depots, I have issued the following **bdf** commands upon initial system installation without any software depots and then again after creating a core operating system and application depot. The location of the software depots is **/var** by default; however, you can load the depots in any directory. I would recommend creating a file system such as **/var/depots** as a depot area. In the following example, we'll focus on the capacity of **/var**, since it is used as the destination for depots in upcoming examples.

```
# bdf
Filesystem          kbytes    used   avail %used Mounted on
/dev/vg00/lvol3     204800   19830  173464   10% /
/dev/vg00/lvol1      83733   13654   61705   18% /stand
/dev/vg00/lvol8    6553600  156292 5998347    3% /var
/dev/vg00/lvol7     512000  341251  160127   68% /usr
/dev/vg00/lvol4     102400    1522   94578    2% /tmp
/dev/vg00/lvol6     602112  186976  389238   32% /opt
/dev/vg00/lvol5     204800    1157  190923    1% /home
/dev/dsk/c1t2d0    2457600  2457600       0  100% /var/tmp/cddirAAAa02574
```

The following **bdf**, which was taken immediately after the core operating system depot for 11.0 was created, shows that **/var** availability has been reduced to about 5.667 GBytes from 5.998 GBytes. This represents roughly 330 MBytes of data that were loaded as part of the core operating system.

```
# bdf
Filesystem          kbytes    used   avail %used Mounted on
/dev/vg00/lvol3     204800   19943  173357   10% /
/dev/vg00/lvol1      83733   13654   61705   18% /stand
/dev/vg00/lvol8    6553600  507003 5668779    8% /var
/dev/vg00/lvol7     512000  341255  160123   68% /usr
/dev/vg00/lvol4     102400    3350   92862    3% /tmp
/dev/vg00/lvol6     602112  186976  389238   32% /opt
/dev/vg00/lvol5     204800    1157  190923    1% /home
/dev/dsk/c1t2d0    2457600 2457600       0  100% /iuxcdrom0
#
```

The disk space consumed by the core operating system depot can be confirmed by viewing the location of the depot as shown in the following listing:

```
# cd /var/opt/ignite/depots
# du -s *
688842  Rel_B.11.00
#
```

This listing of the **du** command shows that roughly 330 MBytes (688K blocks divided by 2048 bytes/block) are in the core operating system directory. This 330 MBytes of disk space consumed by the core operating system depot is nothing compared with all of the application software we're going to load into the **depots**

directory. We're going to put both the core operating system and applications in this directory, so substantially more space will be consumed.

To further drive home the point of the amount of disk space consumed, the following **du** command shows the amount of disk space consumed by both the core operating system depot (still about 330 MBytes) and the total applications contained on the three CD-ROMs that came with the release of the operating system for which we're creating our depots. The **/var/opt/ignite/depots/app** directory, which is the depot for the contents of the three application CD-ROMs, is roughly 3.7 GBytes!

```
# cd /var/opt/ignite/depots
# du -s *
688874  Rel_B.11.00
3722622 app
#
```

The space consumed by the applications is over 10 times the space consumed by the core operating system. This 3.7 GBytes consists of every application available for the 11.0 release of the operating system; however, this gives you an idea of the space required when using Ignite-UX to create depots.

[Team LiB]

◀ PREVIOUS NEXT ▶

Run Ignite-UX GUI

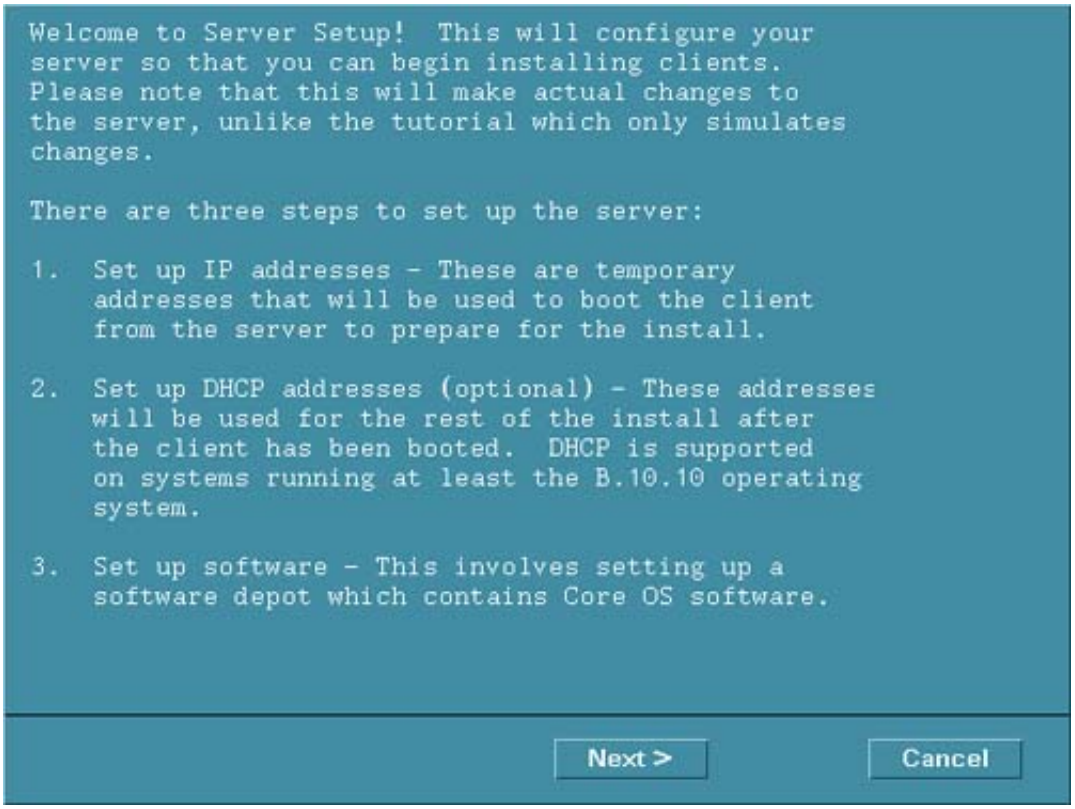
After loading the Ignite-UX tool, you can type **ignite** if you have set up your path for **/opt/ignite/bin**, or you can type the full path **/opt/ignite/bin/ignite** to bring-up the graphical interface for Ingite-UX as shown in Figure 9-1:

Figure 9-1. Ignite-UX Welcome Screen



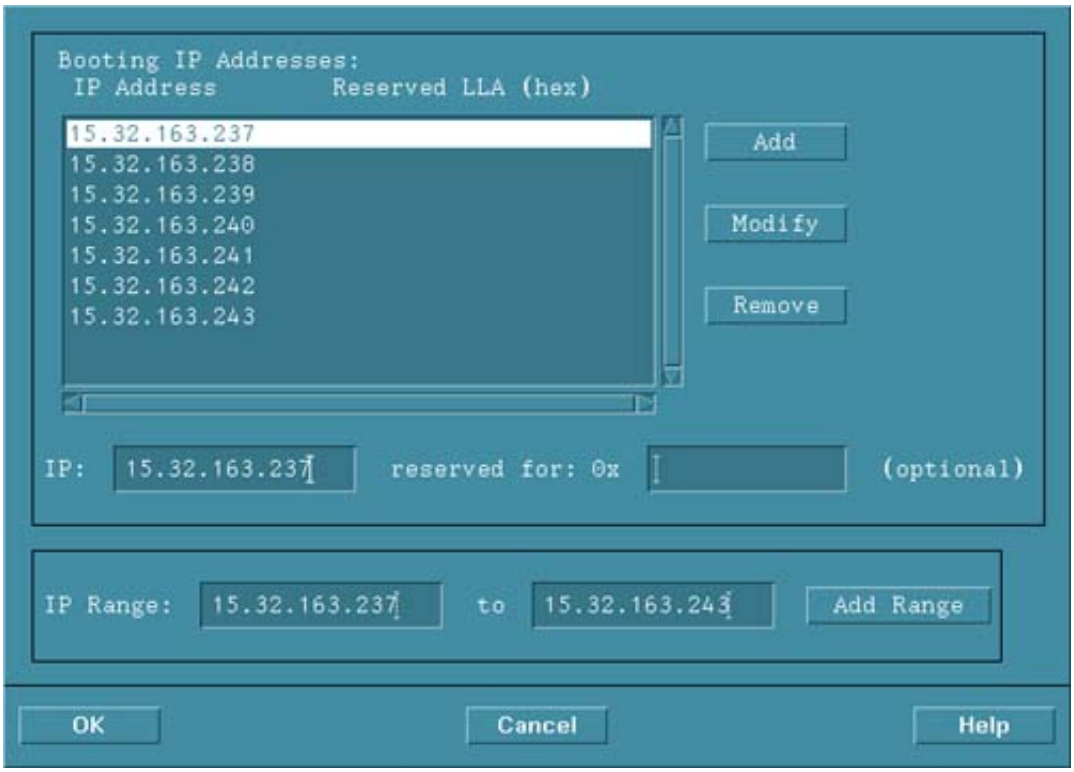
From this window we can make various selections to configure Ignite-UX for our environment. The first configuration we'll perform is *Server Setup.* When we select *Server Setup,* we're shown the screen in Figure 9-2, which allows us to set up IP addresses for the clients we wish to manage from Ignite-UX under option 1.

Figure 9-2. Setup Client IP Address



We'll set up a bank of addresses for several clients we anticipate loading through the Ignite-UX interface as shown in Figure 9-3. Be sure to put entries for these clients in your host database, and the file **/etc/hosts** as well.

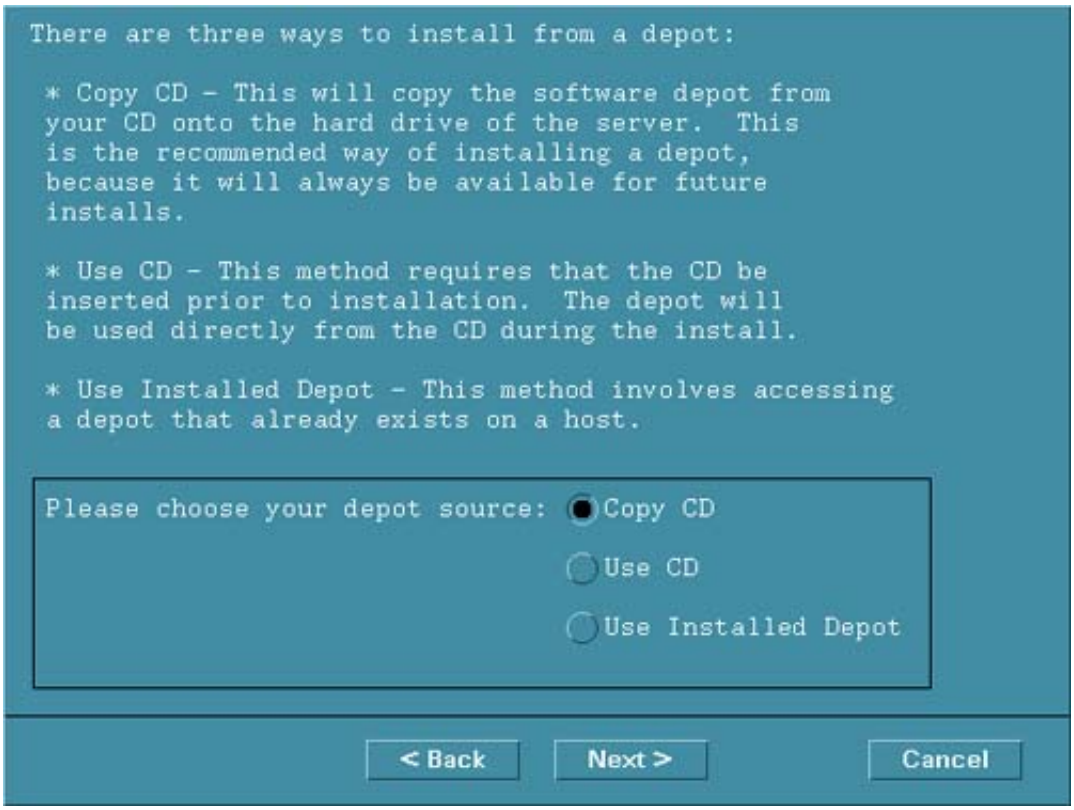
Figure 9-3. A Bank of Client Ip Addresses



We have set up a bank of seven clients that we wish to manage from Ignite-UX.

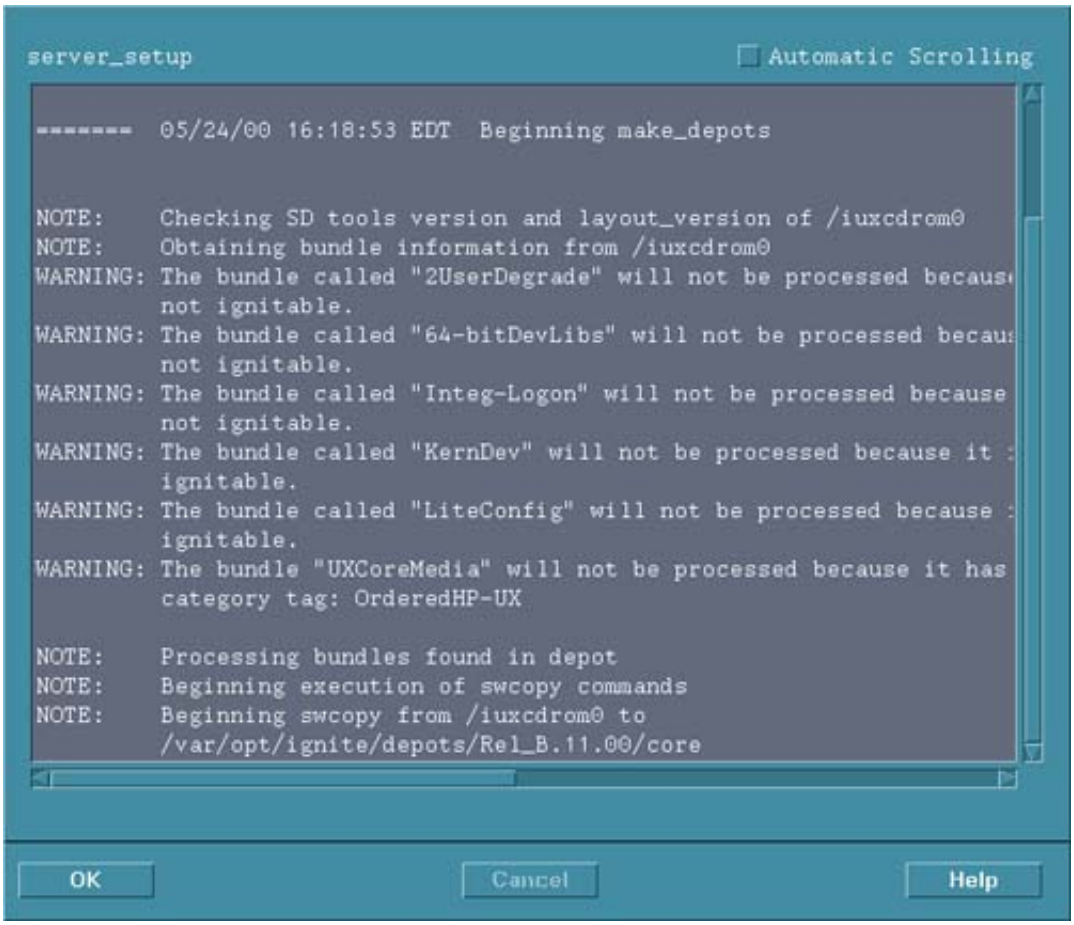
Next, let's create a core operating system depot for HP-UX 11.0 through the graphical interface. We'll copy the core operating system from the orginal CD-ROM, which is the selection we make in Figure 9-4:

Figure 9-4. Install a Software Depot



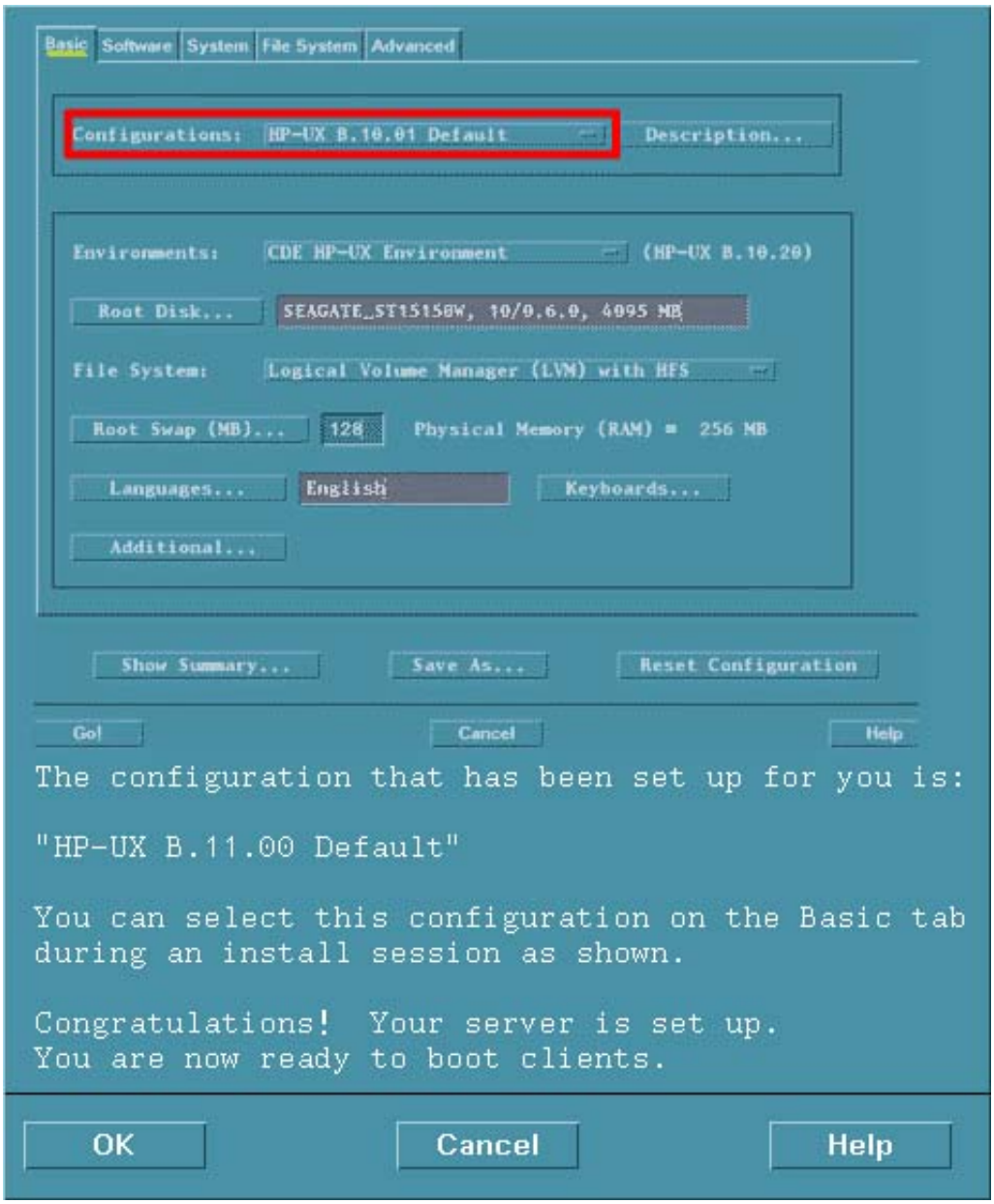
After making the selection to load from the CD-ROM, another window appears, which is not shown in our example. Here, we specify the location in which the depot will be loaded. By default the directory is **/var/opt/ignite/depots** (full path **/var/opt/ignite/depots/Rel_B.11.11/core** as an example for the 11.11 release of HP-UX); however, we can specify any directory location for the depots. If you have a lot of unused disk space in **/localapps** for instance, you could make the full path **/localapps/depots/Rel_B.11.11/core**. Next, we receive a status window showing the progression of the software copy and depot creation, shown in Figure 9-5:

Figure 9-5. Status of Software Depot Creation



Upon completion of the software copy and depot setup, we are given the window shown in Figure 9-6:

Figure 9-6. The Software Depot is Complete



The depot for the core operating system was automatically created under **/var/opt/ignite/depots/Rel_B.11.00**. As described earlier in the chapter, this depot consumes about 330 MBytes of disk space.

At this point, we have created only the core operating system depot and configured the clients all through the graphical interface on the Ignite-UX server. At the time of this writing, there is no way to use the graphical interface to create the applications software depot. We'll use three Ignite-UX commands to create the applications depot. First we'll run the **make_depots** command for all three applications CD-ROMs we wish to load into the depot. Then we'll run the **make_config** command one time for the applications depot we have created. Finally, we'll run the **manage_index** one time for the applications depot we created. Let's take a look at each of these commands in the upcoming listings.

```
# /opt/ignite/bin/make_depots -d /var/opt/ignite/depots/app -s /dev/dsk/c1t2d0
# /opt/ignite/bin/make_depots -d /var/opt/ignite/depots/app -s /dev/dsk/c1t2d0
# /opt/ignite/bin/make_depots -d /var/opt/ignite/depots/app -s /dev/dsk/c1t2d0
```

We add the *-d* option to specify a destination directory called **app** for the contents of the application media. We could also specify an alternate destination directory. Using the earlier example, we could specify the destination with **-d /localapps/depots/app**, for instance. We'll load the entire application media: in this case, three CD-ROMs. Next we run **make_config**:

```
# /opt/ignite/bin/make_config -s /var/opt/ignite/depots/app
               -c /var/opt/ignite/data/Rel_B.11.00/app_cfg

NOTE:    make_config can sometimes take a long time to complete. Please be
         patient!
```

We placed the **app_cfg** file in the directory for our specific release of the operating system. This is a method of organizing the configuration files in such a way that those pertaining to a specific release of the operating system are grouped together. If we had used a different location for the application directory, we would have specified it with **-s /localapps/depots/app**, but left the **app_cfg** file in the same location. You will want to take a look at the configuration files in **/var/opt/ignite/data/<***yourHP-UXrelease***>** to see their contents. These files contain the information about all of the software you have loaded as part of your Ignite-UX depot.

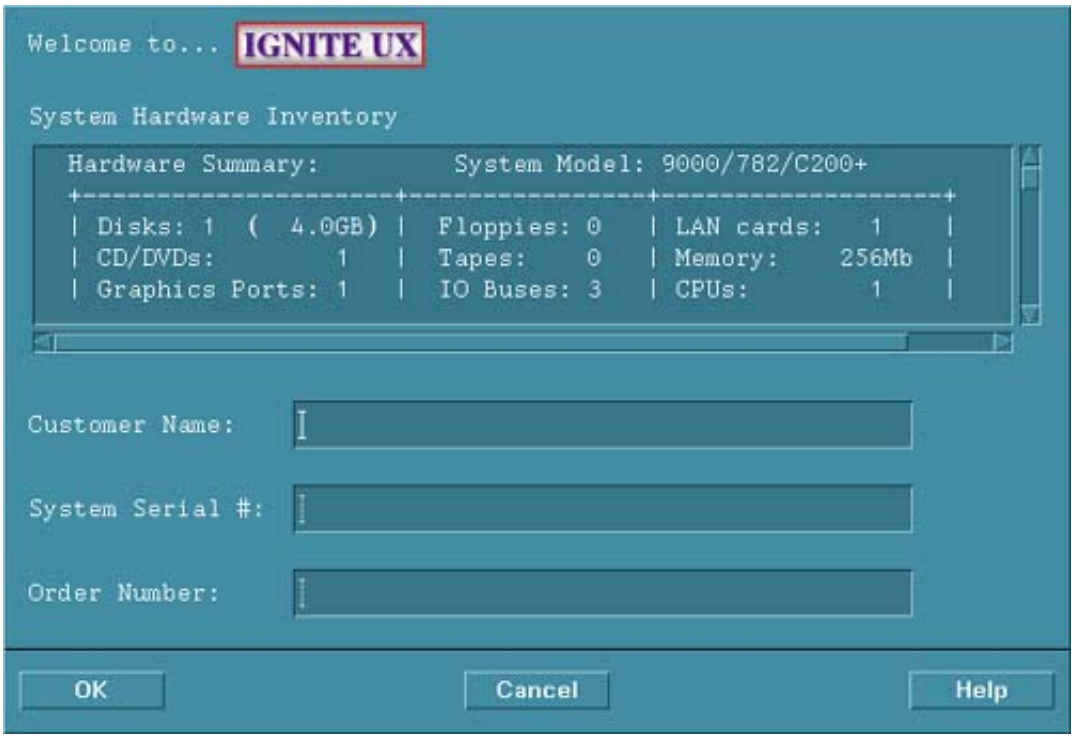
The **manage_index** command is required to manage the index file used by Ignite-UX, as shown in the following example:

```
# /opt/ignite/bin/manage_index -a -f /var/opt/ignite/data/Rel_B.11.00/app_cfg
```

We have now set up the core operating system depot using the Ignite-UX graphical interface and the application depot using the three Ignite-UX commands at the command line. You now also have an entry for your release of the operating system in **/var/opt/ignite/INDEX**. This is a file you will want to view so that you can see the entry made for your release of the operating system.

Let's now get back to using the Ignite-UX graphical interface on the server. We can get information about a specific client through the Ignite-UX interface on the server. The window that appears on the server is shown in Figure 9-7 for the client we'll be booting through Ignite-UX:

Figure 9-7. Information about Client



Now that we've created both a core operating system and an applications depot we can boot and load one of our clients through Ignite-UX on the server.

In addition to controlling the client as we have done using the Ignite-UX graphical interface in "push" fashion from the server, you can also work from the client in "pull" fashion. In order to boot the client from the server while sitting at the server, you would issue the following command from PDC (Processor Dependent Code; see Chapter 1) after interrupting the boot process:

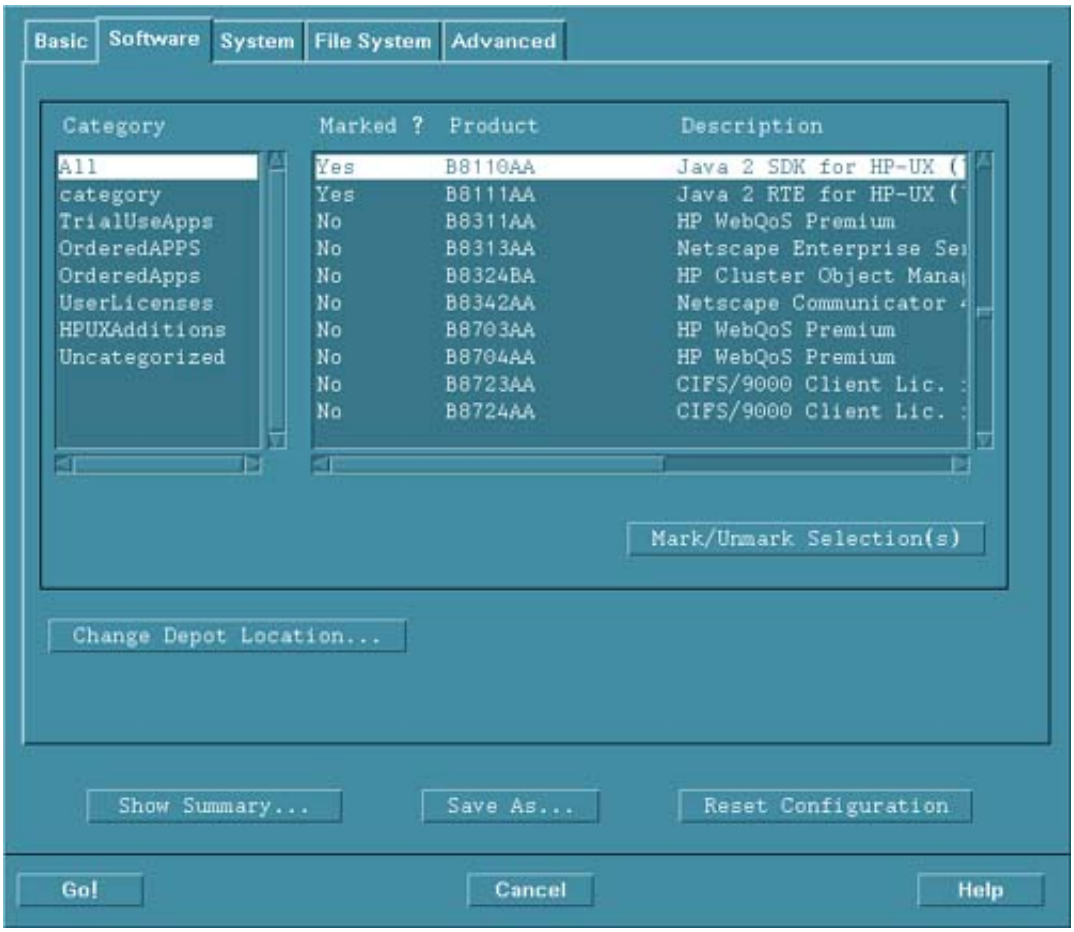



```
BOOT ADMIN> boot lan 15.32.163.26 install
```

The IP address specified is that of the server where we set up the software depot. While sitting at the client, you would see the Ignite-UX interface in character mode and proceed with the installation while sitting at the client using the software depot we set up on the server.

Getting back to the GUI, we select the software we wish to load through the familiar Ignite-UX interface that is part of any initial operating system load as shown in Figure 9-8:

Figure 9-8. Selecting Software to Load on Client

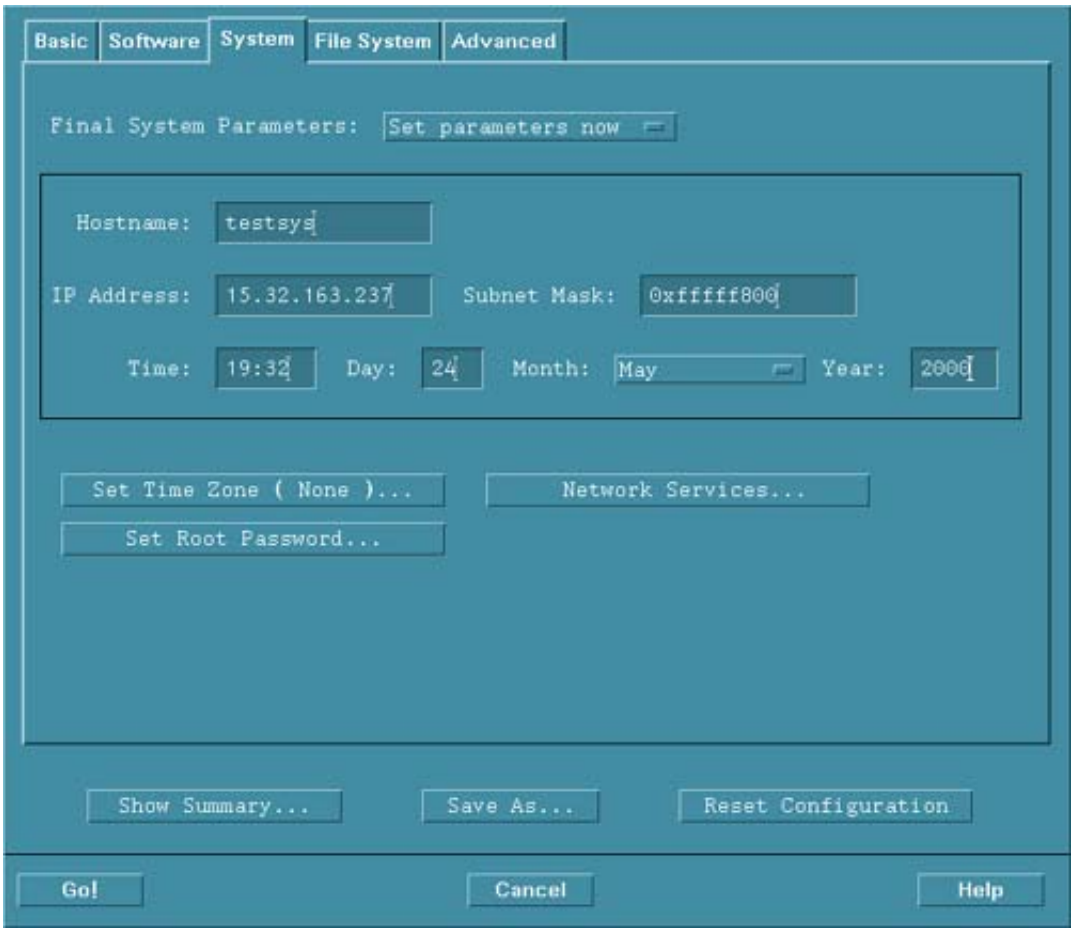


Notice in Figure 9-8 that we are in the *Software* tab at the top of the screen. We'll progress through the *System* and *File System* tabs in two upcoming figures. Remember, all of this work is being done on the Ignite-UX server we set up.

We have selected a Java product that was part of the applications depot we created in Figure 9-8. The core operating system and other software selections were also made in the *Software* window.

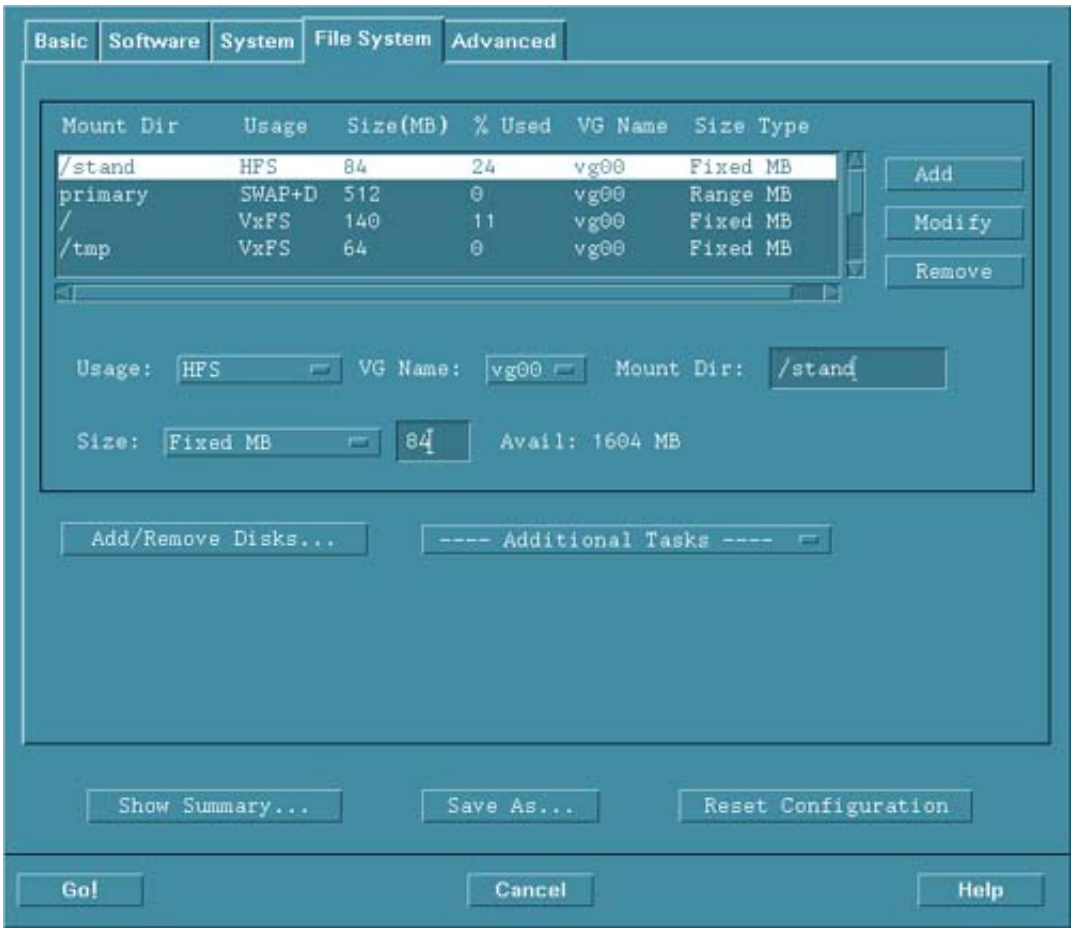
In addition to the software load, we can also perform additional configuration on the server for the client we are loading such as providing the system name under the *System* tab as shown in Figure 9-9:

Figure 9-9. Setup the Client System



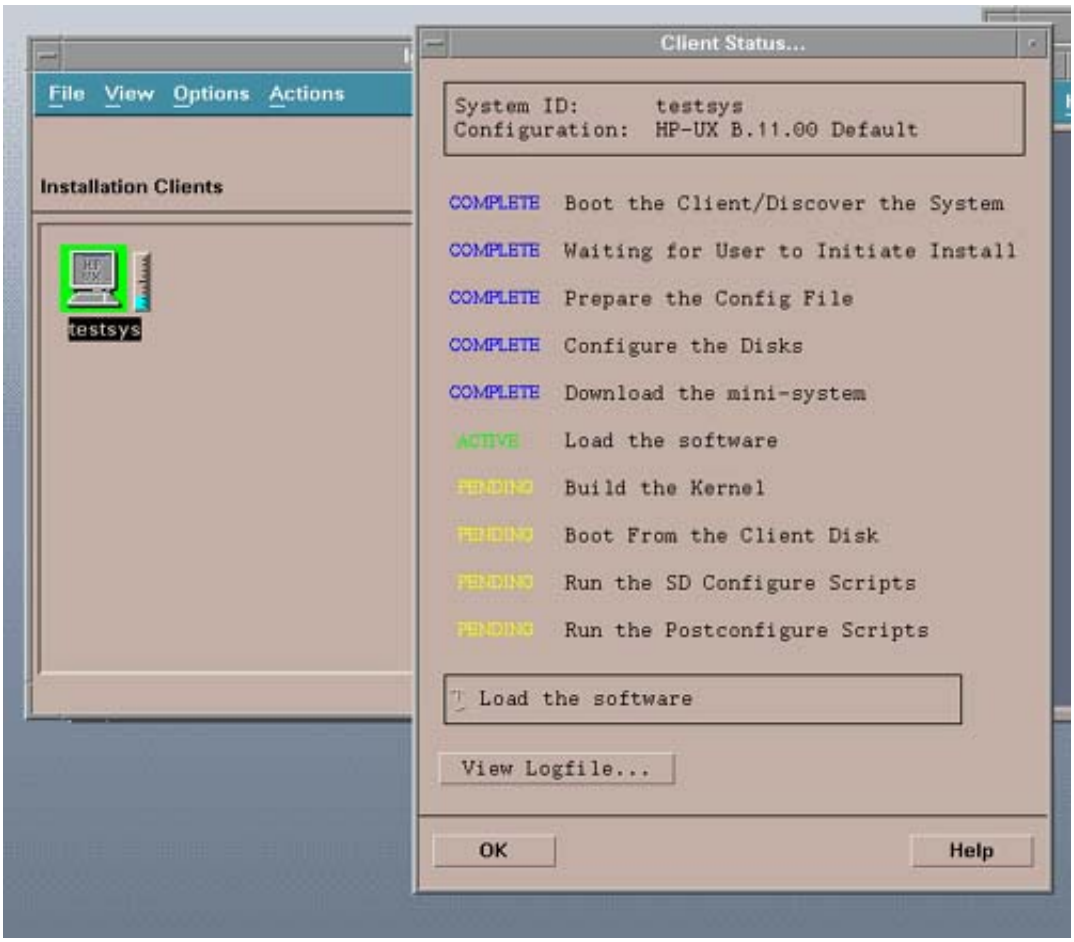
Next we can perform file system-related configuration under the *File System* tab, as shown in Figure 9-10:

Figure 9-10. Setup the Client File Systems



After we have completed all of our initial work in the Ignite-UX for the client while working on the server, we can proceed with the client load. Figure 9-11 shows two Ignite-UX windows that appear on the server when we begin configuring and loading the client:

Figure 9-11. Status of Client Load



The left-most window in Figure 9-11 shows the icon for the system we are working on and the right-most window provides the status of the client load. Since we are working on only one system, only one icon appears in the left-most window. The status of the software load in the right-most window indicates that *Load the software* is an *ACTIVE* process. Serveral process have been *COMPLETE,* and several others are *PENDING*.

Although you can't see the colors in the original screen shot, the status messages are color-coded.

There is substantially more that you can accomplish with Ignite-UX, such as grouping the configuration files into sets that you can select when loading clients, installing patch depots, and performing many other useful tasks. The *Ignite-UX Administrator Guide* at www.docs.hp.com covers many additional topics.

From the procedure we performed in this section, however, you can see how easy it is to create software depots and manage clients.

System Recovery with Ignite-UX

Ignite-UX is a product bundled with HP-UX 11i that provides a process to create a bootable system recovery tape or network archive. The tape or network archive contains a boot area and an operating system archive. Should your root disk fail or corruption of the root disk take place, you can recover using the bootable system recovery tape or archive.

Creating the tape and archive are done in a similar manner. The **make_tape_recovery** command to create a tape and the **make_net_recovery** command to create a network archive use many of the same options, such as those for including and excluding files in the archive. The primary difference is that with **make_tape_recovery** you write to a tape drive, and with **make_net_recovery** you write the archive to a system on the network which acts as a repository for storing archives.

In the upcoming two sections we'll cover writing a tape and writing a network archive. The tape archive will be created using **make_tape_recovery** on a local system, and both volume groups vg00 and vg01 will be included in the archive.

The network archive will be created using **make_net_recovery**, and the Ignite-UX GUI, across a network. This example will be on a system running Virtual Partitions (vPars) software. Since it is unlikely that every vPar on a system will have its own tape drive, the **make_net_recovery** command is ideal for creating archives on vPars-based systems.

System Recovery to Local Tape Drive with make_tape_recovery

Ignite-UX is a product bundled with HP-UX 11i that provides a process to create a bootable system recovery tape. The tape contains a boot area and an operating system archive. Should your root disk fail or corruption of the root disk take place, you can recover using the bootable system recovery tape.

make_tape_recovery is the Ignite-UX tool that is used to create the bootable system recovery tape. You could boot and restore from the system recovery tape and then use your system backup to fully restore the system. If your root disk were to fail, in a non-mirrored environment, you would perform the following steps to recover from the failure:

- Replace the defective root disk.
- Boot from the recovery tape by selecting the tape device.
- Monitor the restoration of the operating system archive from the recovery tape.
- Restore the balance of data on the system with backup information.

The bootable system recovery tape consists of both a Logical Interchange Format (LIF) volume as well as the operating system archive. The LIF volume contains all the components necessary to boot from the tape. The operating system archive contains only the core operating system by default. The following bullet lists describe the components contained in the respective areas of the tape:

Boot and Configruation Files:

- Boot files including *ISL, HPUX, AUTO*
- System configuration information such as *hostname*, *IP address*, *disk layout*, and so on.
- Ignite-UX kernel and user interface.

Archive Information includes one of the following three:

- Minimum Core-OS to boot such as **/sbin**, **/dev**, **/stand**, **/etc**, **/usr/bin**, **/usr/lib**, **/bin**, **/lib**, **/etc**, and so on.
- Entire root volume group.
- Files from both root and non-root volume groups.

You can easily include additional information to be placed in the archive as well as preview what processing will take place. The following is a summary of the **make_tape_recovery** command:

```
/opt/ignite/bin/make_tape_recovery [ -s Ignite-UX_server ]
  [-a tape_drive] [-A] [-b] [-B boot_destination_file]
  [-d description] [-f content_file] [-i|-ib] [-I] [-l lanic_id]
  [-n num_cfg_dirs] [-p] [-P s|w|e] [-r] [-t tape_title_string]
  [-v] [-x content-options] [XToolkit_Options]
```

We'll use some of these options in our upcoming command line example. The following list shows the options you would use to add to or exclude from the archive:

Adding to or excluding from an archive:

- -x include=<*file*|*directory*>

- -x inc_cross=<*file*|*directory*> (this can contain any number of *include*, *exclude*, *inc_entire*, and *inc_cross* options.
- -x exclude=<*file*|*directory*>
- -x inc_entire=<*vgname*>|*/dev/dsk/*<*name*>
- -f <*contents_file*>
- -A (archive all volume groups that contain essential files)

You include information in the archive with *-x include* and exclude with *-x exclude* except for essential files which can't be excluded. *-x inc_cross* includes a file or directory recursively and crosses mount points. *-x inc_entire* will include an entire volume or disk.

Let's now run make_tape_recovery on a system. We'll issue three commands in the following example. The first is **bdf**, so that we can see the two volume groups on the system we wish to include: *vg00* and *vg01*. Next we'll run **make_tape_recovery** in *preview* mode with *-p* and include both volume groups. Lastly we'll *resume* making the tape with the *-r* option with *verbose* output specified with *-v*:

[View full width]

```
# bdf
Filesystem          kbytes    used   avail %used Mounted on
/dev/vg00/lvol3     204800   70942  125523   36% /
/dev/vg00/lvol1     299157   26222  243019   10% /stand
/dev/vg00/lvol8    2609152  140280 2315011    6% /var
/dev/vg00/lvol7    1056768  767093  271616   74% /usr
/dev/vg00/lvol5     204800    1368  190782    1% /tmp
/dev/vg00/lvol6    1056768  691086  342888   67% /opt
/dev/vg00/lvol4     512000    1234  478849    0% /home
/dev/vg01/lvol1    4096000    2104 3838035    0% /backup
# make_tape_recovery -x inc_entire=vg00 -x inc_entire=vg01 -p
       * Creating local directories for configuration files and archive.

=======  03/25/02 10:36:24 EST  Started make_tape_recovery. (Mon Mar 25
         10:36:24 EST 2002)
         @(#) Ignite-UX Revision B.3.5.89
         @(#) net_recovery (opt) $Revision: 10.565 $

       * Testing pax for needed patch
       * Passed pax tests.
       * Creating System Configuration.
       * /opt/ignite/bin/save_config -f
         /var/opt/ignite/recovery/2002-03-25,10:36/system_cfg vg00 vg01
       * Backing Up Volume Group /dev/vg00
       * /usr/sbin/vgcfgbackup /dev/vg00
Volume Group configuration for /dev/vg00 has been saved in /etc/lvmconf/vg00.conf
       * Creating Map Files for Volume Group /dev/vg00
       * /usr/sbin/vgexport -p -m /etc/lvmconf/vg00.mapfile /dev/vg00
vgexport: Volume group "/dev/vg00" is still active.

       * Backing Up Volume Group /dev/vg01
       * /usr/sbin/vgcfgbackup /dev/vg01
Volume Group configuration for /dev/vg01 has been saved in /etc/lvmconf/vg01.conf
       * Creating Map Files for Volume Group /dev/vg01
       * /usr/sbin/vgexport -p -m /etc/lvmconf/vg01.mapfile /dev/vg01
vgexport: Volume group "/dev/vg01" is still active.

       * Creating Control Configuration.
       * Creating Archive File List
       * Creating Archive Configuration
vxvm:vxprint: ERROR: IPC failure: Configuration daemon is not accessible

       * /opt/ignite/bin/make_arch_config -c
         /var/opt/ignite/recovery/2002-03-25,10:36/archive_cfg -g
         /var/opt/ignite/recovery/2002-03-25,10:36/flist -n 2002-03-25,10:36 -r
         64 -d Recovery\ Archive -t -i 1
       * Saving the information about archive to
         /var/opt/ignite/recovery/previews
```

```
       *
       * **** This is just Preview Only.                          ****
       * **** No archive has been created yet.                    ****
       * **** You may modify the configure files                  ****
       * **** and use -r option for resuming to create archive. ****


=======  03/25/02 10:37:02 EST  make_tape_recovery completed successfully!
# make_tape_recovery -r -v
NOTE:    ******* RESUMING *******
NOTE:    archive size: 1841410 kB
NOTE:    last preview is on 2002-03-25,10:36
NOTE:    The archive name is 2002-03-25,10:36
NOTE:    The archive description Recovery Archive
NOTE:    The archive is on local tape drive /dev/rmt/0mn
NOTE:    You want to resume archive creation from the configuration files
         locally on this system.
NOTE:    In Resume Mode.  Do you wish to continue?
Continue?(y/n)y
       * Creating The Tape Archive
       * Checking configuration file for Syntax
       * Writing boot LIF to tape
30000+1 records in
30000+1 records out
       * Complete writing boot LIF onto tape.
       * /opt/ignite/data/scripts/make_sys_image -c n -d /dev/rmt/0mn -t n -s
         local -n 2002-03-25,10:36 -m t -w
         /var/opt/ignite/recovery/latest/recovery.log -u -R -g
        /var/opt/ignite/recovery/latest/flist -a 1841410             * Creation of system
➥ archive complete
       * Preparing to create a system archive
NOTE: Could not read the /etc/resolv.conf file.
     * Archiving contents of mponiatowsk4457 via tar image to local device/dev/rmt/0mn.
=======  03/25/02 10:59:42 EST  make_tape_recovery completed successfully!
#
```

When we *resume* the **make_tape_recovery**, the process of creating the bootable archive tape completes. The LIF area is first written to the tape, then all of the information from the two volume groups is included. You can see that the **make_sys_image** script is called after the LIF area is written to the tape.

Both volume groups were active at the time this tape archive was produced; however, there was minimal activity on the system.

Creating Bootable Recovery Archives with make_net_recovery

In this section we'll create a bootable recovery archive for a Virtual Partition using the Ignite-UX GUI. In addition, we'll take a look at some of what is taking place in the background such as:

- Disk space consumed by a bootable recovery archive on the server.

- Server files created and modified to support creating the client archives (**/etc/exports**, others.)

- Command issued by server on client to produce the archive.

- In the next section we'll restore a client by booting from the archive created on the server (note that the relocatable kernel is different from a standard kernel.)

Using Ignite-UX GUI to Create a vPar Archive

After the Ignite-UX tool is loaded on the server there is some initial client setup that is required and managed through the Ignite-UX Graphical User Interface (GUI). There are also some modifications made to the server. One of these changes is to make entries for the clients to be managed in **/etc/exports**. Ignite-UX automatically included entries in **/etc/exports** for the clients that were set up. The following **/etc/exports** listing shows an entry for two clients on the server *chess2*:

```
root@chess2 > cat /etc/exports

/var/opt/ignite/clients -anon=2
/var/opt/ignite/recovery/archives/actappd1 -anon=2,access=actappd1
/var/opt/ignite/recovery/archives/ecvpard2 -anon=2,access=ecvpard2

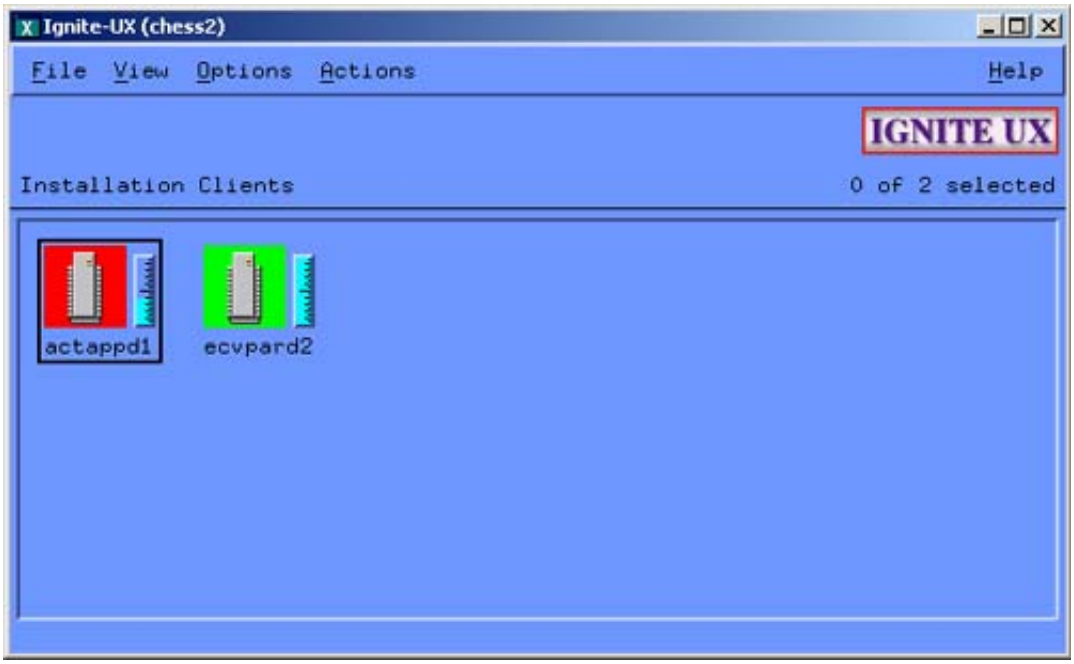
root@chess2 >
```

The three entries include the general *clients* directory used by the Ignite-UX server, and *archives* directories for the two clients that were setup on the Ignite-UX server, called *actappd1* and *ecvpard2*. These two hosts are Virtual Partitions running on the same server. *chess2*, the Ignite-UX server, is a different server than that on which the two vPars are running.

In addition to **/etc/exports** updates that were automatically made when specifying clients, you may want to manually create entries in **/.rhosts** on both the server and client so that you won't have to respond with the client password on the server when you initiate creating the bootable recovery archive. The server is running a program remotely on the client, and in order to do so without issuing a password, the **/.rhosts** file must have the appropriate entries. Keep in mind, though, that including entries in **/.rhosts** reduces the overall security level of your system.

Once this has been completed, you can initiate the process of creating the bootable recovery archive as shown in Figure 9-12 on our server *chess2*:

Figure 9-12. Specifying Client for Which Bootable Recovery Archive Will Be Produced



The two clients shown in the Ignite-UX GUI were those for which we saw entries in **/etc/exports**. Keep in mind that the two systems shown in the Ignite-UX GUI are two Virtual Partitions running on the same

server.

There are several *Actions* from which we can choose once a client has been selected, as shown in Figure 9-13:

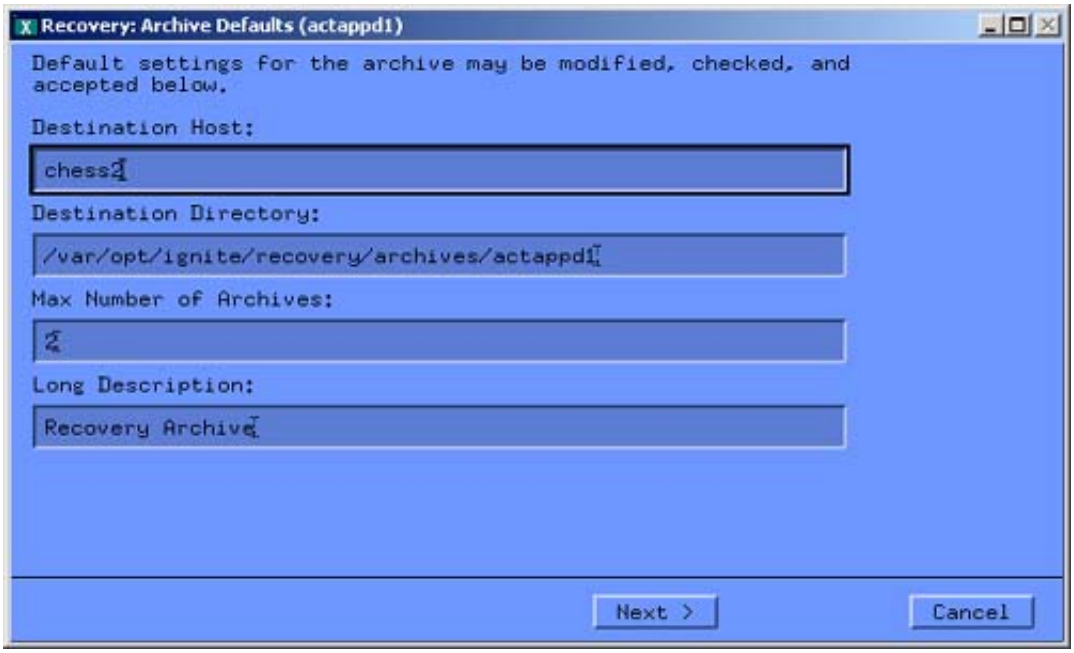
Figure 9-13. *Actions* Menu after Client Has Been Selected



With *actappd1* selected, we'll choose *Create Network Recovery Archive* from the *Actions* menu.

Making this menu selection brings up a window that contains a series of defaults for creating the archive, as shown in Figure 9-14:

Figure 9-14. Defaults for Creating a Bootable Recovery Archive



After we progress through these basic screens, Ignite-UX begins the process of creating the archive.

The server issued the **make_net_recovery** command on the client in order to create the archive. The following is the output of **ps -ef** for only this command on the client (*actappd1*) for which the archive is being produced:

```
root@actappd1 > ps -ef | grep make

/opt/ignite/bin/make_net_recovery -i -b -P s -l 0x00306E0981D4

root@actappd1 >
```

This command is now running in the background on our client, which is a Virtual Partition, and writing an archive across the network to the server. This process takes about three hours on our client and produces a file over 600 MBytes in size.

If you're interested in the many options to **make_net_recovery**, you can view the online man page for the command. In this example the *i* is for *interactive*, *b* for *background*, *-P* for *partial*, *s* for *server*, and *-l* for *lanic id* of the client. You could issue this command from the client as well as from the server. Some system administrators prefer to manage the creation of the archives centrally from the server and others prefer to create **cron** entries on the clients and initiate this process from the client. Both the setup and syntax of the **make_net_recovery** command appear in its man page (at the time of this writing, **make_net_recovery** is the latest command.)

There are many files created as a result of initiating the creation of the recovery archive for the client. The following listing shows some of the more interesting directories and files. The man page for **make_net_recovery** provides an explanation for all of the important directories and files created for clients. Let's take a look at some of the more interesting ones on the Ignite-UX server used in our example in this chapter.

The following is a long listing of the **/var/opt/ignite/clients** directory on the server:

```
root@chess2[/var/opt/ignite/clients] > ll
total 4
drwxrwxrwx   3 bin        bin       1024 Jul 20 12:23 0x001083F7C4FC
drwxr-xr-x   3 bin        bin       1024 Jul 19 17:54 0x00306E0981D4
drwxr-xr-x   2 root       sys         96 Jul 19 14:13 161.228.212.121
lrwxr-xr-x   1 bin        bin         14 Jul 27 14:01 actappd1 -> 0x00306E0981D4
lrwxr-xr-x   1 bin        bin         14 Jul 20 09:53 ecvpard2 -> 0x001083F7C4FC
lrwxrwxrwx   1 root       sys         14 May 25 15:53 sample -> 0x999999999999
```

Notice that there are entries for both of our clients in this directory. There is a link that allows you to specify the client name rather than the ID of the LAN card because the client name is much easier to remember and type.

Next, let's change to the directory of the client for which we are creating our archive and perform a long listing:

```
root@chess2[/var/opt/ignite/clients] > cd actappd1
root@chess2[/var/opt/ignite/clients/actappd1] > ll
total 18
-rw-r--r--   1 bin        bin          987 Jul 24 18:33 CINDEX
-rw-r--r--   1 bin        bin            9 Jul 19 15:15 client_name
-rw-r--r--   1 bin        bin           28 Jul 19 15:15 client_status
-rw-r--r--   1 bin        bin          300 Jul 19 15:15 config.sys
-rw-r--r--   1 bin        bin          336 Jul 19 15:15 host.info
-rw-r--r--   1 bin        bin          940 Jul 19 15:15 hw.info
-rw-r--r--   1 bin        bin          231 Jul 19 15:15 install.log
-rw-r--r--   1 bin        bin            0 Jul 19 15:20 lockfile
drwxr-xr-x   3 bin        bin         1024 Jul 27 14:01 recovery
-rw-r--r--   1 bin        bin          585 Jul 27 13:39 server.state
```

There is a lot of client-related information in this directory. The **recovery** directory has the most interesting files related to our client, so let's change to that directory and perform a long listing:

```
root@chess2[/var/opt/ignite/clients/actappd1] > cd recovery
root@chess2[/var/opt/ignite/clients/actappd1/recovery] > ll
total 8
drwxr-xr-x   2 bin        bin       1024 Jul 27 14:05 2001-07-27,13:55
-rw-r--r--   1 bin        bin        252 Jul 27 14:03 archive_content
-rw-r--r--   1 bin        bin        247 Jul 27 14:05 client_status
-rw-r--r--   1 bin        bin        411 Jul 27 14:03 defaults
lrwxr-xr-x   1 bin        bin         16 Jul 27 14:01 latest -> 2001-07-27,13:55
```

The **archive_content** file has in it the files and directories to be included or excluded. **client_status** is used to communicate with the client to provide the status of the **make_net_recovery** command. A status window is shown in the GUI after the creation of the archive is initiated. **latest** is a link to the directory that has in it the newest recovery files. If we change to this directory and perform a long listing, we see the files in the following listing:

```
root@chess2[/var/opt/ignite/clients/actappd1/recovery] > cd 2001-07-27,13:55
@chess2[/var/opt/ignite/clients/actappd1/recovery/2001-07-27,13:55] > ll
total 3704
-rw-r--r--   1 bin        bin           3089 Jul 27 14:05 archive_cfg
-rw-r--r--   1 bin        bin            252 Jul 27 14:03 archive_content
-rw-r--r--   1 bin        bin            269 Jul 27 14:03 control_cfg
-rw-r--r--   1 bin        bin        1880616 Jul 27 14:05 flist
-rw-r--r--   1 bin        bin           2510 Jul 27 14:05 recovery.log
-rw-r--r--   1 root       sys           5588 Jul 27 13:58 system_cfg
```

The **recovery.log** file contains information about the archive we are currently creating. The following output contains the contents of the file:

```
=======  07/27/01 13:55:54 EDT  Started /opt/ignite/bin/make_net_recovery. (Fri
         Jul 27 13:55:54 EDT 2001)
         @(#) Ignite-UX Revision B.3.4.115
         @(#) net_recovery (opt) $Revision: 10.543 $


       * User interface starting.
NOTE:    Detected entries in the defaults file at
         /var/opt/ignite/recovery/client_mnt/0x00306E0981D4/recovery/defaults.

       * User interface completed successfully.
NOTE:    Detected entries in the defaults file at
         /var/opt/ignite/recovery/client_mnt/0x00306E0981D4/recovery/defaults.

       * Checking Versions of Recovery Tools
       * Creating System Configuration.
       * /opt/ignite/bin/save_config -f /var/opt/ignite/recovery/client_mnt/0x0
         0306E0981D4/recovery/2001-07-27,13:55/system_cfg vg00
       * Backing Up Volume Group /dev/vg00
       * /usr/sbin/vgcfgbackup /dev/vg00
Volume Group configuration for /dev/vg00 has been saved in /etc/lvmconf/vg00.conf
       * Creating Map Files for Volume Group /dev/vg00
       * /usr/sbin/vgexport -p -m /etc/lvmconf/vg00.mapfile /dev/vg00
vgexport: Volume group "/dev/vg00" is still active.

       * Creating Control Configuration.
       * Creating Archive File List
       * Creating Archive Configuration

       * /opt/ignite/bin/make_arch_config -c /var/opt/ignite/recovery/client_mn
         t/0x00306E0981D4/recovery/2001-07-27,13:55/archive_cfg -g /var/opt/ign
         ite/recovery/client_mnt/0x00306E0981D4/recovery/2001-07-27,13:55/flist
         -n 2001-07-27,13:55 -r 64 -d Recovery\ Archive -L
         /var/opt/ignite/recovery/arch_mnt -l
         chess2:/var/opt/ignite/recovery/archives/actappd1 -i 1
       * Saving the information about archive to
         /var/opt/ignite/recovery/previews
       * Creating The Networking Archive

       * /opt/ignite/data/scripts/make_sys_image -d
         /var/opt/ignite/recovery/arch_mnt -t n -s local -n 2001-07-27,13:55 -m
         t -w /var/opt/ignite/recovery/client_mnt/0x00306E0981D4/recovery/2001-
         07-27,13:55/recovery.log -u -R -g /var/opt/ignite/recovery/client_mnt/
         0x00306E0981D4/recovery/2001-07-27,13:55/flist -a 905500

       * Preparing to create a system archive
       * The archive is estimated to reach 452750 kbytes.
       * Free space on /var/opt/ignite/recovery/arch_mnt
         after archive should be about 4066 kbytes.

       * Archiving contents of actappd1 via tar to
         /var/opt/ignite/recovery/arch_mnt/2001-07-27,13:55.
@chess2[/var/opt/ignite/clients/actappd1/recovery/2001-07-27,13:55] > ll
total 3704
-rw-r--r--   1 bin        bin           3089 Jul 27 14:05 archive_cfg
```

```
-rw-r--r--   1 bin        bin            252 Jul 27 14:03 archive_content
-rw-r--r--   1 bin        bin            269 Jul 27 14:03 control_cfg
-rw-r--r--   1 bin        bin        1880616 Jul 27 14:05 flist
-rw-r--r--   1 bin        bin           2510 Jul 27 14:05 recovery.log
-rw-r--r--   1 root       sys           5588 Jul 27 13:58 system_cfg
```

There are many useful files produced for each client; however, the one in which we are most interested is the recovery archive. The following listing shows three archives that have been produced for our client.

```
root@chess2[/var/opt/ignite/recovery/archives/actappd1] > ls -l
total 1432116
-rw-------   1 bin        bin        623288688 Jul 24 18:33 2001-07-24,13:08
-rw-------   1 bin        bin        36152098 Jul 25 16:08 2001-07-25,15:42
-rw-------   1 bin        bin        73728000 Jul 27 14:42 2001-07-27,13:55
root@chess2[/var/opt/ignite/recovery/archives/actappd1] >
```

The first is a complete archive created at an earlier date, the second is a partial archive that was not complete because it was aborted, and the third is the archive we were creating at the time this listing was produced.

This shows that our Virtual Partition consumed over 600 MBytes of disk space for the complete archive. You'll want to consider the number of clients for which you wish to store bootable recovery archives on your server and plan your volume space accordingly. For the archive we are currently producing, there have been about 73 MBytes of what will be a recovery archive of over 600 MByte recovery archive present, and we have been running **make_net_recovery** on the client for about 20 minutes.

After our archive has been fully created we can progress to booting it as a test. I usually create an archive as soon as a Virtual Partition has been produced and test it before I perform a lot of customization on the vPar. This will give you peace of mind that you know the process of booting from the recovery archive.

Virtual Partition Recovery

Now that we've created the bootable recovery archive for our Virtual Partition, we can restore our vPar using it. You can restore one vPar from another using the **vparboot** command. This means that while one vPar is running, you can restore another vPar on the same system using the following **vparboot** command:

```
vparboot -p vp_name -I ignite_kernel
```

This is different from the **vparboot** form used in Chapter 2 when we booted a vPar using a kernel that was located on the same system, as opposed to an Ignite-UX server. That form of the **vparboot** is as follows:

```
vparboot -p vp_name [-b kernel_path] [-o boot_opts] [-B boot_addr]
```

In examples in early chapters we simply specified the name of the Virtual Partition we wanted to boot, such as **vparboot -p cable1**. With this form of **vparboot** we specify a boot kernel on the local system with the *-b* option. The man page for **vparboot** and the other vPar commands are found in Appendix A.

You would normally boot a recovery archive from the PDC prompt, that is, your system is down and you are recovering it. Since we have multiple vPars on our system, we can recover one vPar on a system from a running vPar on the same system using **vparboot**.

We'll issue the following **vparboot** command from our running vPar in order to restore vPar *actappd1* from an Ignite-UX server at the command line:

```
# vparboot -p actappd1 -I 161.228.212.110,/opt/ignite/boot/WINSTALL
```



The IP address of our Ignite-UX server is specified. The **WINSTALL** file used in this **vparboot** command is different from that used for normal HP-UX recovery (without vPars). A relocatable kernel is required for vPars, so the **WINSTALL** used in the **vparboot** command is one that is required for vPars. This **WINSTALL** is supplied with vPars.

The ability to recover a Virtual Partition from a running vPar using **vparboot** means that fewer users are affected. The users in the running vPar are unaffected by the recovery taking place on another vPar. After the recovery of the root volume group (*vg00*) takes place, you can restore the balance of information on your vPar from the latest backup. Be sure to create the bootable recovery archive often so that your entire root volume group can be recovered with the latest data.

Manual Pages for Commands Used in Chapter 9

There are manual pages in Appendix A and B for the following commands used in this chapter:

boot
vparboot

Chapter 10. System Administration Manager (SAM) and ServiceControl Manager (SCM)

SAM and SCM

SAM Overview

Running and Using SAM as Superuser

Author's Disclaimer - SAM Is a Moving Target

Running Restricted SAM Builder

Accounts for Users and Groups

Auditing and Security

Backup and Recovery

Clusters

Disks and File Systems

Kernel Configuration

Networking and Communications

Partition Manager

Performance Monitors

Peripheral Devices

Printers and Plotters

Process Management

Routine Tasks

Run SAM on Remote Systems

Software Management

Time

NFS Diskless Concepts

ServiceControl Manager (SCM) Overview

Manual Pages for Commands Used in Chapter 10



SAM and SCM

SAM and SCM are programs with Graphical User Interfaces (GUI) through which you can perform many system administration tasks. SAM is run on a local or remote system and SCM allows you to replicate tasks over multiple systems. Tasks that have traditionally been performed individually on many systems can now be performed across multiple systems in a cluster managed by SCM. Since these are both useful management tools, I decided to cover them in the same chapter: first SAM and the SCM.

At the time of this writing, development is under way for several web-based system administration tools. One such tool that is now available is **kcweb** for working with the kernel. **kcweb** is covered in both Chapter 4 and this chapter and gives you an indication of the type of Web-based tools under development.



SAM Overview

SAM is a program you can use to automate and perform various system administration tasks. SAM has been refined over many releases of HP-UX and is now a highly functional and reliable tool for performing routine system administration tasks. As you'll see in the upcoming sections, SAM has many *functional areas* in which you can perform system administration tasks. Since there is a lot of "coverage" with SAM, meaning that most system administration tasks can be performed with SAM, combined with its reliability in performing these tasks, I have found more and more system administrators using SAM for performing system administration tasks.

Four features of SAM that make it particularly useful are:

1. It provides a central point from which system administration tasks can be performed. This includes both the built-in tasks that come with SAM as well as those you can add into the SAM menu hierarchy. You can run SAM on a remote system and display it locally so that you truly have a central point of control.

2. It provides an easy way to perform tasks that are difficult, in that you would have to perform many steps. SAM performs these steps for you.

3. It provides a summary of what your system currently looks like for any of the categories of administration tasks you wish to perform. If you want to do something with the disks on your system, SAM first lists the disks you currently have connected. If you want to play with a printer, SAM first lists all your printers and plotters for you. This capability cuts down on mistakes by putting your current configuration right in front of you.

4. You can assign non-root users to perform some of the system administration functions in SAM. If, for instance, you feel comfortable assigning one of your associates to manage users, you can give them permission to perform user-related tasks and give another user permission to perform backups, and so on.

There are some tasks SAM can't perform for you. SAM does most routine tasks for you, but troubleshooting a problem is not considered routine. Troubleshooting a problem gives you a chance to show off and hone your system administration skills.

When SAM is performing routine tasks for you, it isn't doing anything you couldn't do yourself by issuing a series of HP-UX commands. SAM provides a simple user interface that allows you to perform tasks by selecting menu items and entering pertinent information essential to performing the task.

Running and Using SAM as Superuser

To run SAM, log in as root and type:

```
# sam                                    (or sam &)
```

This will invoke SAM. If you have a graphics display, SAM will run with the Motif interface. If you have a character-based display, SAM will run in character mode. You have nearly all the same functionality in both modes, but the Motif environment is much more pleasant to use.

If you have a graphics display and SAM does not come up in a Motif window, you probably don't have your *DISPLAY* variable set for root.

Type the following to set the *DISPLAY* variable for default POSIX, Korn, and Bourne shells:

```
# DISPLAY=system_name:0.0
# export DISPLAY
```

Just substitute the name of your computer for *system_name*. This can be set in your local **.profile** file. If you're running HP CDE, you may want to put these lines in your **.dtprofile** file.

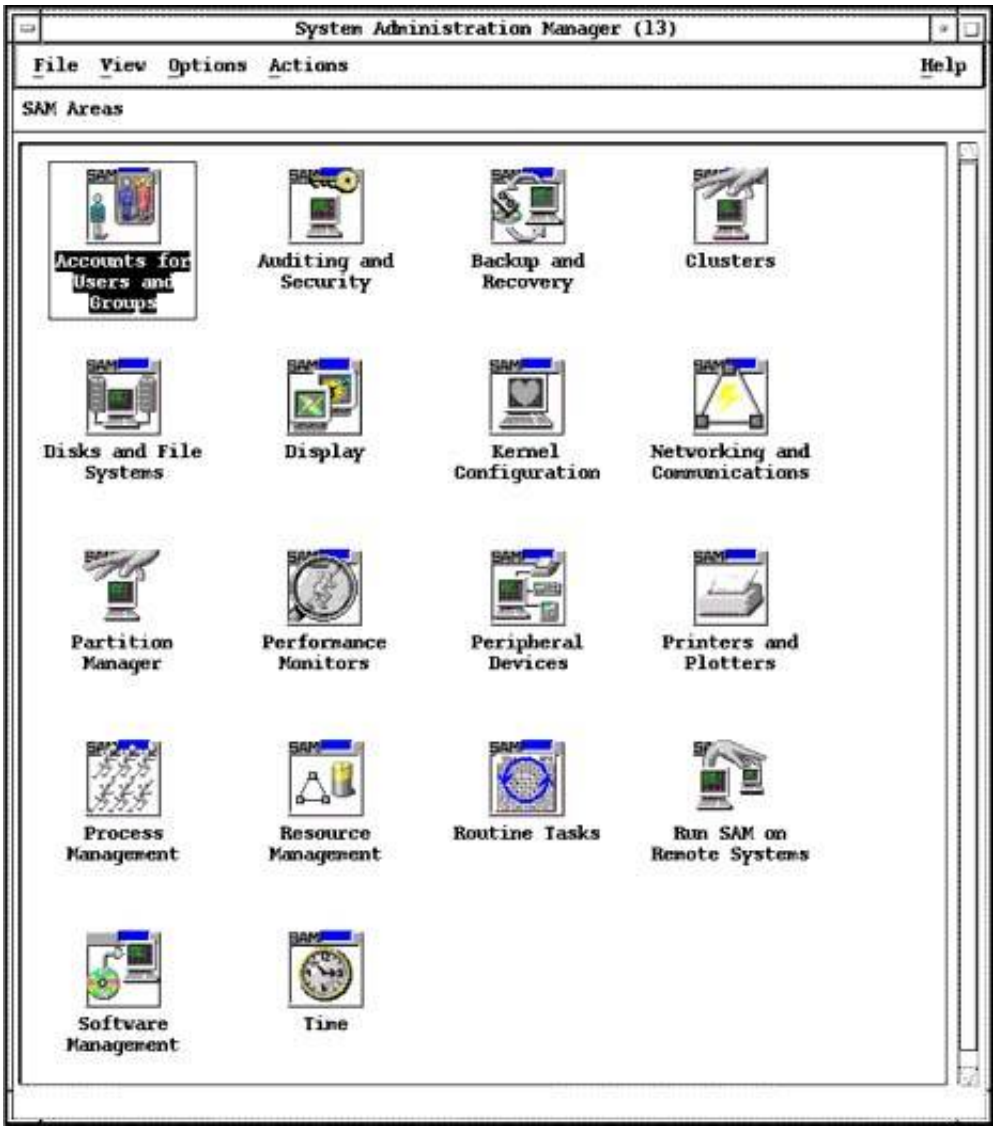
Type the following to set the *DISPLAY* variable for C shell:

```
# setenv DISPLAY system_name:0.0
```

Again, you would substitute the name of your computer for *system_name.* This would typically be done in your **.login** file, but if you're running HP CDE, you may want to put this in your **.dtprofile** file. Most CDE users, however, have **.dtprofile** as the source for **.profile** or **.login**.

Figure 10-1 shows the System Administration Manager running in graphics mode. This is the top-level window of the hierarchical SAM environment called the *Functional Area Launcher* (*FAL*). There are many categories, or areas, of management; shown are the default functional areas managed by SAM. You can select one of these functional areas to be placed in a subarea. Because SAM is hierarchical, you may find yourself working your way down through several levels of the hierarchy before you reach the desired level. I'll cover each of these categories, or areas, in this chapter.

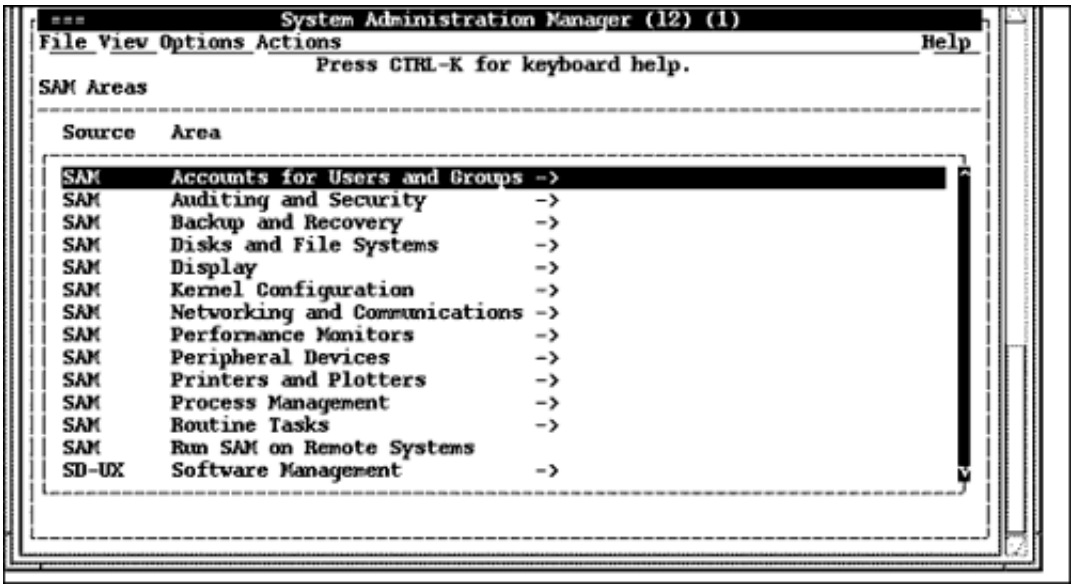
Figure 10-1. HP-UX 11i SAM Startup Window in Graphics Mode

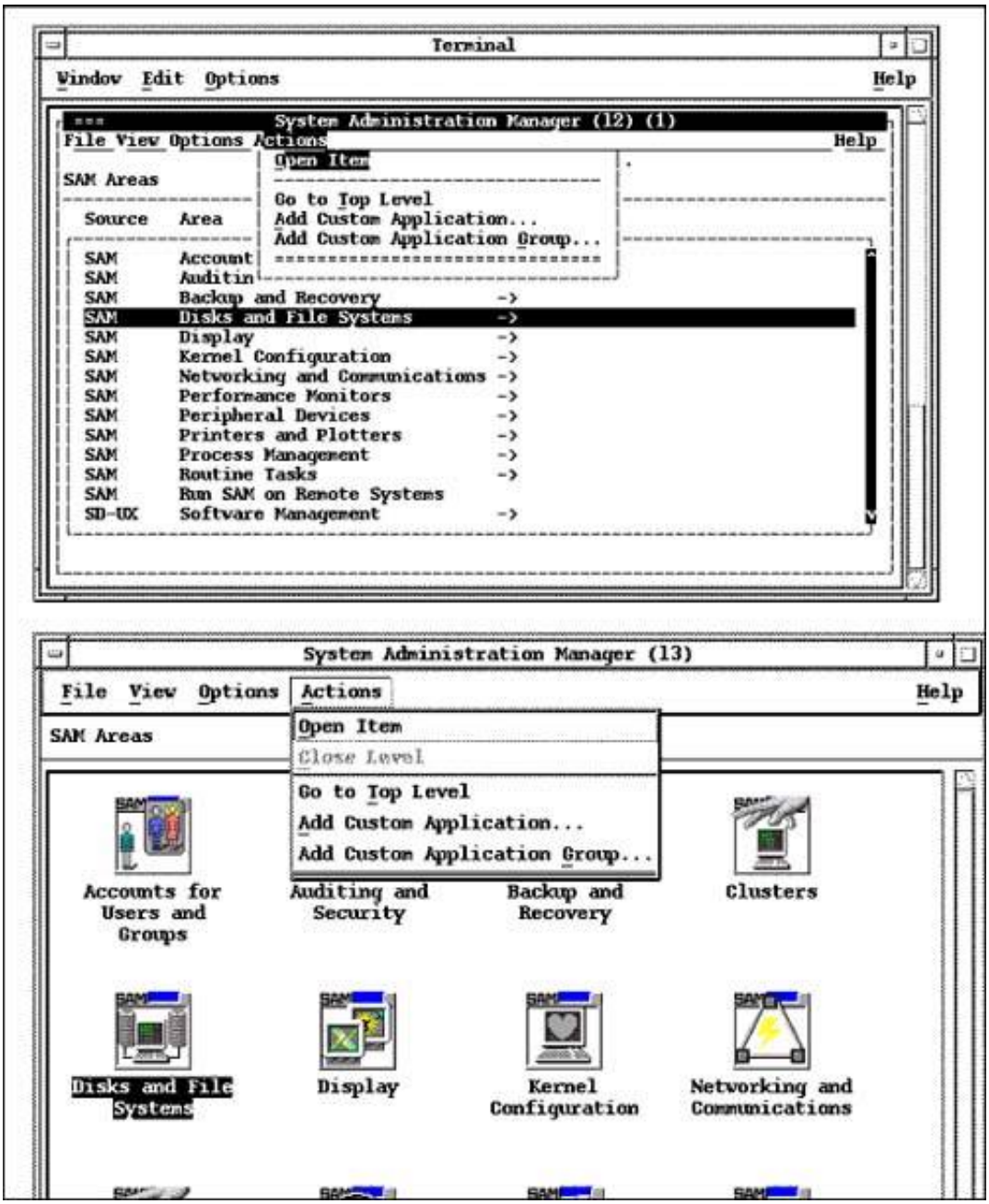


In addition to selecting a functional area, you can select from the pull-down menu bar across the top of the SAM window. I will indicate selections made in SAM and keyboard keys in this chapter with italics. The five selections are *File, View, Options, Actions,* and *Help*. The title line shown in Figure 10-1 reads *SAM Areas*. If you're running Restricted SAM Builder, you will also see a status line with the message "*Privileges for user: <username>*." As you progress down the hierarchy, the title line will change to reflect your level in the SAM hierarchy. You can move into one of the areas shown, such as *Backup and Recovery*, by double-clicking the left mouse button on this functional area. You move back up the hierarchy by double-clicking the ..(*go up)* icon, if available, or by selecting the *Actions-Close Level* menu commands.

You don't need a graphics display to run SAM. You have access to nearly all the same functionality on a text terminal as you do on a graphics terminal. Figure 10-2 is SAM running in character mode with the same functional areas you have in graphics mode, although the bottom selections are not shown in the figure:

Figure 10-2. SAM Startup Window in Character Mode


System Administration Manager (12) (1)
File View Options Actions
Help
Press CTRL-K for keyboard help.
SAM Areas
Source
Area
SAM Accounts for Users and Groups ->
SAM Auditing and Security ->
SAM Backup and Recovery ->
SAM Disks and File Systems ->
SAM Display ->
SAM Kernel Configuration ->
SAM Networking and Communications ->
SAM Performance Monitors ->
SAM Peripheral Devices ->
SAM Printers and Plotters ->
SAM Process Management ->
SAM Routine Tasks ->
SAM Run SAM on Remote Systems
SD-UX Software Management ->
SAM Areas
Source
Area
SAM Accounts for Users and Groups ->
SAM Auditing and Security ->
SAM Backup and Recovery ->
SAM Disks and File Systems ->
SAM Display ->
SAM Kernel Configuration ->
SAM Networking and Communications ->
SAM Performance Monitors ->
SAM Peripheral Devices ->
SAM Printers and Plotters ->
SAM Process Management ->
SAM Routine Tasks ->
SAM Run SAM on Remote Systems
SD-UX Software Management ->
Actions
Add New User Account...
Modify System Defaults...
Modify Password Configuration...
Modify Maximum Number of Users...
Modify Login Time Limits...
Modify Security Policies...
Modify Restricted SU...
Modify Process Accounting...
Modify Audit System...
Modify Audit Users...
Modify Audit Events...
Modify Audit Logs...
Modify Audit Files...
Modify Audit Backup...
Modify Audit Reports...
Modify Audit Alarms...
Modify Audit Compression...
Modify Audit Archive...
Modify Audit Archive Logs...
Modify Audit Archive Files...
Modify Audit Archive Backup...
Modify Audit Archive Reports...
Modify Audit Archive Alarms...
Modify Audit Archive Compression...
Modify Audit Archive Archive...
Modify Audit Archive Archive Logs...
Modify Audit Archive Archive Files...
Modify Audit Archive Archive Backup...
Modify Audit Archive Archive Reports...
Modify Audit Archive Archive Alarms...
Modify Audit Archive Archive Compression...
Modify Audit Archive Archive Archive...
Modify Audit Archive Archive Archive Logs...
Modify Audit Archive Archive Archive Files...
Modify Audit Archive Archive Archive Backup...
Modify Audit Archive Archive Archive Reports...
Modify Audit Archive Archive Archive Alarms...
Modify Audit Archive Archive Archive Compression...
Modify Audit Archive Archive Archive Archive...
Modify Audit Archive Archive Archive Archive Logs...
Modify Audit Archive Archive Archive Archive Files...
Modify Audit Archive Archive Archive Archive Backup...
Modify Audit Archive Archive Archive Archive Reports...
Modify Audit Archive Archive Archive Archive Alarms...
Modify Audit Archive Archive Archive Archive Compression...
Modify Audit Archive Archive Archive Archive Archive...
Modify Audit Archive Archive Archive Archive Archive Logs...
Modify Audit Archive Archive Archive Archive Archive Files...
Modify Audit Archive Archive Archive Archive Archive Backup...
Modify Audit Archive Archive Archive Archive Archive Reports...
Modify Audit Archive Archive Archive Archive Archive Alarms...
Modify Audit Archive Archive Archive Archive Archive Compression...
Modify Audit Archive Archive Archive Archive Archive Archive...
Modify Audit Archive Archive Archive Archive Archive Archive Logs...
Modify Audit Archive Archive Archive Archive Archive Archive Files...
Modify Audit Archive Archive Archive Archive Archive Archive Backup...
Modify Audit Archive Archive Archive Archive Archive Archive Reports...
Modify Audit Archive Archive Archive Archive Archive Archive Alarms...
Modify Audit Archive Archive Archive Archive Archive Archive Compression...
Modify Audit Archive Archive Archive Archive Archive Archive Archive...
Modify Audit Archive Archive Archive Archive Archive Archive Archive Logs...
Modify Audit Archive Archive Archive Archive Archive Archive Archive Files...
Modify Audit Archive Archive Archive Archive Archive Archive Archive Backup...
Modify Audit Archive Archive Archive Archive Archive Archive Archive Reports...
Modify Audit Archive Archive Archive Archive Archive Archive Archive Alarms...
Modify Audit Archive Archive Archive Archive Archive Archive Archive Compression...
Modify Audit Archive Archive Archive Archive Archive Archive Archive Archive...
Modify Audit Archive Archive Archive Archive Archive Archive Archive Archive Logs...
Modify Audit Archive Archive Archive Archive Archive Archive Archive Archive Files...
Modify Audit Archive Archive Archive Archive Archive Archive Archive Archive Backup...
Modify Audit Archive Archive Archive Archive Archive Archive Archive Archive Reports...
Modify Audit Archive Archive Archive Archive Archive Archive Archive Archive Alarms...
Modify Audit Archive Archive Archive Archive Archive Archive Archive Archive Compression...
Modify Audit Archive Archive Archive Archive Archive Archive Archive Archive Archive...
Modify Audit Archive Archive Archive Archive Archive Archive Archive Archive Archive Logs...
Modify Audit Archive Archive Archive Archive Archive Archive Archive Archive Archive Files...
Modify Audit Archive Archive Archive Archive Archive Archive Archive Archive Archive Backup...
Modify Audit Archive Archive Archive Archive Archive Archive Archive Archive Archive Reports...
Modify Audit Archive Archive Archive Archive Archive Archive Archive Archive Archive Alarms...
Modify Audit Archive Archive Archive Archive Archive Archive Archive Archive Archive Compression...
Modify Audit Archive Archive Archive Archive Archive Archive Archive Archive Archive Archive...
Modify Audit Archive Archive Archive Archive Archive Archive Archive Archive Archive Archive Logs...
Modify Audit Archive Archive Archive Archive Archive Archive Archive Archive Archive Archive Files...
Modify Audit Archive Archive Archive Archive Archive Archive Archive Archive Archive Archive Backup...
Modify Audit Archive Archive Archive Archive Archive Archive Archive Archive Archive Archive Reports...
Modify Audit Archive Archive Archive Archive Archive Archive Archive Archive Archive Archive Alarms...
Modify Audit Archive Archive Archive Archive Archive Archive Archive Archive Archive Archive Compression...
Modify Audit Archive Archive Archive Archive Archive Archive Archive Archive Archive Archive Archive...
Modify Audit Archive Archive Archive Archive Archive Archive Archive Archive Archive Archive Archive Logs...
Modify Audit Archive Archive Archive Archive Archive Archive Archive Archive Archive Archive Archive Files...
Modify Audit Archive Archive Archive Archive Archive Archive Archive Archive Archive Archive Archive Backup...
Modify Audit Archive Archive Archive Archive Archive Archive Archive Archive Archive Archive Archive Reports...
Modify Audit Archive Archive Archive Archive Archive Archive Archive Archive Archive Archive Archive Alarms...
Modify Audit Archive Archive Archive Archive Archive Archive Archive Archive Archive Archive Archive Compression...
Modify Audit Archive Archive Archive Archive Archive Archive Archive Archive Archive Archive Archive Archive...
Modify Audit Archive Archive Archive Archive Archive Archive Archive Archive Archive Archive Archive Archive Logs...
Modify Audit Archive Archive Archive Archive Archive Archive Archive Archive Archive Archive Archive Archive Files...
Modify Audit Archive Archive Archive Archive Archive Archive Archive Archive Archive Archive Archive Archive Backup...
Modify Audit Archive Archive Archive Archive Archive Archive Archive Archive Archive Archive Archive Archive Reports...
Modify Audit Archive Archive Archive Archive Archive Archive Archive Archive Archive Archive Archive Archive Alarms...
Modify Audit Archive Archive Archive Archive Archive Archive Archive Archive Archive Archive Archive Archive Compression...
Modify Audit Archive Archive Archive Archive Archive Archive Archive Archive Archive Archive Archive Archive Archive...
Modify Audit Archive Archive Archive Archive Archive Archive Archive Archive Archive Archive Archive Archive Archive Logs...
Modify Audit Archive Archive Archive Archive Archive Archive Archive Archive Archive Archive Archive Archive Archive Files...
Modify Audit Archive Archive Archive Archive Archive Archive Archive Archive Archive Archive Archive Archive Archive Backup...
Modify Audit Archive Archive Archive Archive Archive Archive Archive Archive Archive Archive Archive Archive Archive Reports...
Modify Audit Archive Archive Archive Archive Archive Archive Archive Archive Archive Archive Archive Archive Archive Alarms...
Modify Audit Archive Archive Archive Archive Archive Archive Archive Archive Archive Archive Archive Archive Archive Compression...
Modify Audit Archive Archive Archive Archive Archive Archive Archive Archive Archive Archive Archive Archive Archive Archive...
Modify Audit Archive Archive Archive Archive Archive Archive Archive Archive Archive Archive Archive Archive Archive Archive Logs...
Modify Audit Archive Archive Archive Archive Archive Archive Archive Archive Archive Archive Archive Archive Archive Archive Files...
Modify Audit Archive Archive Archive Archive Archive Archive Archive Archive Archive Archive Archive Archive Archive Archive Backup...
Modify Audit Archive Archive Archive Archive Archive Archive Archive Archive Archive Archive Archive Archive Archive Archive Reports...
Modify Audit Archive Archive Archive Archive Archive Archive Archive Archive Archive Archive Archive Archive Archive Archive Alarms...
Modify Audit Archive Archive Archive Archive Archive Archive Archive Archive Archive Archive Archive Archive Archive Archive Compression...
Modify Audit Archive Archive Archive Archive Archive Archive Archive Archive Archive Archive Archive Archive Archive Archive Archive...
Modify Audit Archive Archive Archive Archive Archive Archive Archive Archive Archive Archive Archive Archive Archive Archive Archive Logs...
Modify Audit Archive Archive Archive Archive Archive Archive Archive Archive Archive Archive Archive Archive Archive Archive Archive Files...
Modify Audit Archive Archive Archive Archive Archive Archive Archive Archive Archive Archive Archive Archive Archive Archive Archive Backup...
Modify Audit Archive Archive Archive Archive Archive Archive Archive Archive Archive Archive Archive Archive Archive Archive Archive Reports...
Modify Audit Archive Archive Archive Archive Archive Archive Archive Archive Archive Archive Archive Archive Archive Archive Archive Alarms...
Modify Audit Archive Archive Archive Archive Archive Archive Archive Archive Archive Archive Archive Archive Archive Archive Archive Compression...
Modify Audit Archive Archive Archive Archive Archive Archive Archive Archive Archive Archive Archive Archive Archive Archive Archive Archive...
Modify Audit Archive Archive Archive Archive Archive Archive Archive Archive Archive Archive Archive Archive Archive Archive Archive Archive Logs...
Modify Audit Archive Archive Archive Archive Archive Archive Archive Archive Archive Archive Archive Archive Archive Archive Archive Archive Files...
Modify Audit Archive Archive Archive Archive Archive Archive Archive Archive Archive Archive Archive Archive Archive Archive Archive Archive Backup...
Modify Audit Archive Archive Archive Archive Archive Archive Archive Archive Archive Archive Archive Archive Archive Archive Archive Archive Reports...
Modify Audit Archive Archive Archive Archive Archive Archive Archive Archive Archive Archive Archive Archive Archive Archive Archive Archive Alarms...
Modify Audit Archive Archive Archive Archive Archive Archive Archive Archive Archive Archive Archive Archive Archive Archive Archive Archive Compression...
Modify Audit Archive Archive Archive Archive Archive Archive Archive Archive Archive Archive Archive Archive Archive Archive Archive Archive Archive...
Modify Audit Archive Archive Archive Archive Archive Archive Archive Archive Archive Archive Archive Archive Archive Archive Archive Archive Archive Logs...
Modify Audit Archive Archive Archive Archive Archive Archive Archive Archive Archive Archive Archive Archive Archive Archive Archive Archive Archive Files...
Modify Audit Archive Archive Archive Archive Archive Archive Archive Archive Archive Archive Archive Archive Archive Archive Archive Archive Archive Backup...
Modify Audit Archive Archive Archive Archive Archive Archive Archive Archive Archive Archive Archive Archive Archive Archive Archive Archive Archive Reports...
Modify Audit Archive Archive Archive Archive Archive Archive Archive Archive Archive Archive Archive Archive Archive Archive Archive Archive Archive Alarms...
Modify Audit Archive Archive Archive Archive Archive Archive Archive Archive Archive Archive Archive Archive Archive Archive Archive Archive Archive Compression...
Modify Audit Archive Archive Archive Archive Archive Archive Archive Archive Archive Archive Archive Archive Archive Archive Archive Archive Archive Archive...
Modify Audit Archive Archive Archive Archive Archive Archive Archive Archive Archive Archive Archive Archive Archive Archive Archive Archive Archive Archive Logs...
Modify Audit Archive Archive Archive Archive Archive Archive Archive Archive Archive Archive Archive Archive Archive Archive Archive Archive Archive Archive Files...
Modify Audit Archive Archive Archive Archive Archive Archive Archive Archive Archive Archive Archive Archive Archive Archive Archive Archive Archive Archive Backup...
Modify Audit Archive Archive Archive Archive Archive Archive Archive Archive Archive Archive Archive Archive Archive Archive Archive Archive Archive Archive Reports...
Modify Audit Archive Archive Archive Archive Archive Archive Archive Archive Archive Archive Archive Archive Archive Archive Archive Archive Archive Archive Alarms...
Modify Audit Archive Archive Archive Archive Archive Archive Archive Archive Archive Archive Archive Archive Archive Archive Archive Archive Archive Archive Compression...
Modify Audit Archive Archive Archive Archive Archive Archive Archive Archive Archive Archive Archive Archive Archive Archive Archive Archive Archive Archive Archive...
Modify Audit Archive Archive Archive Archive Archive Archive Archive Archive Archive Archive Archive Archive Archive Archive Archive Archive Archive Archive Archive Logs...
Modify Audit Archive Archive Archive Archive Archive Archive Archive Archive Archive Archive Archive Archive Archive Archive Archive Archive Archive Archive Archive Files...
Modify Audit Archive Archive Archive Archive Archive Archive Archive Archive Archive Archive Archive Archive Archive Archive Archive Archive Archive Archive Archive Backup...
Modify Audit Archive Archive Archive Archive Archive Archive Archive Archive Archive Archive Archive Archive Archive Archive Archive Archive Archive Archive Archive Reports...
Modify Audit Archive Archive Archive Archive Archive Archive Archive Archive Archive Archive Archive Archive Archive Archive Archive Archive Archive Archive Archive Alarms...
Modify Audit Archive Archive Archive Archive Archive Archive Archive Archive Archive Archive Archive Archive Archive Archive Archive Archive Archive Archive Archive Compression...
Modify Audit Archive...
Modify Audit Archive Logs...
Modify Audit Archive Files...
Modify Audit Archive Backup...
Modify Audit Archive Reports...
Modify Audit Archive Alarms...
Modify Audit Archive Compression...
Modify Audit Archive...
Modify Audit Archive Logs...
Modify Audit Archive Files...
Modify Audit Archive Backup...
Modify Audit Archive Reports...
Modify Audit Archive Alarms...
Modify Audit Archive Compression...
Modify Audit Archive...
Modify Audit Archive Logs...
Modify Audit Archive Files...
Modify Audit Archive Backup...
Modify Audit Archive Reports...
Modify Audit Archive Alarms...
Modify Audit Archive Compression...
Modify Audit Archive...
Modify Audit Archive Logs...
Modify Audit Archive Files...
Modify Audit Archive Backup...
Modify Audit Archive Reports...
Modify Audit Archive Alarms...
Modify Audit Archive Compression...
Modify Audit Archive...
Modify Audit Archive Logs...
Modify Audit Archive Files...
Modify Audit Archive Backup...
Modify Audit Archive Reports...
Modify Audit Archive Alarms...
Modify Audit Archive Compression...
Modify Audit Archive...
Modify Audit Archive Logs...
Modify Audit Archive Files...
Modify Audit Archive Backup...
Modify Audit Archive Reports...
Modify Audit Archive Alarms...
Modify Audit Archive Compression...
Modify Audit Archive...
Modify Audit Archive Logs...
Modify Audit Archive Files...
Modify Audit Archive Backup...
Modify Audit Archive Reports...
Modify Audit Archive Alarms...
Modify Audit Archive Compression...
Modify Audit Archive...
Modify Audit Archive Logs...
Modify Audit Archive Files...
Modify Audit Archive Backup...
Modify Audit Archive Reports...
Modify Audit Archive Alarms...
Modify Audit Archive Compression...
Modify Audit Archive...
Modify Audit Archive Logs...
Modify Audit Archive Files...
Modify Audit Archive Backup...
Modify Audit Archive Reports...
Modify Audit Archive Alarms...
Modify Audit Archive Compression...
Modify Audit Archive...
Modify Audit Archive Logs...
Modify Audit Archive Files...
Modify Audit Archive Backup...
Modify Audit Archive Reports...
Modify Audit Archive Alarms...
Modify Audit Archive Compression...
Modify Audit Archive...
Modify Audit Archive Logs...
Modify Audit Archive Files...
Modify Audit Archive Backup...
Modify Audit Archive Reports...
Modify Audit Archive Alarms...
Modify Audit Archive Compression...
Modify Audit Archive...
Modify Audit Archive Logs...
Modify Audit Archive Files...
Modify Audit Archive Backup...
Modify Audit Archive Reports...
Modify Audit Archive Alarms...
Modify Audit Archive Compression...
Modify Audit Archive...
Modify Audit Archive Logs...
Modify Audit Archive Files...
Modify Audit Archive Backup...
Modify Audit Archive Reports...
Modify Audit Archive Alarms...
Modify Audit Archive Compression...
Modify Audit Archive...
Modify Audit Archive Logs...
Modify Audit Archive Files...
Modify Audit Archive Backup...
Modify Audit Archive Reports...
Modify Audit Archive Alarms...
Modify Audit Archive Compression...
Modify Audit Archive...
Modify Audit Archive Logs...
Modify Audit Archive Files...
Modify Audit Archive Backup...
Modify Audit Archive Reports...
Modify Audit Archive Alarms...
Modify Audit Archive Compression...
Modify Audit Archive...
Modify Audit Archive Logs...
Modify Audit Archive



Of particular interest on the pull-down menu are *Add Custom Application* and *Add Custom Application Group*. When you use *Add Custom Application Group,* you are prompted for a *Label* and optional *Help File* for the group. After you enter this information, a new icon appears, if you have a graphics display, with the name of your application group. You can then go into this application group and *Add Custom Applications*. This means that you can customize SAM to meet your specific administration needs by adding functionality to SAM. After you familiarize yourself with the aspects of system administration SAM can help you with, you'll want to test adding your own application to SAM. Adding a simple application such as opening a log file or issuing the **/usr/bin/find** command will take you only seconds to create.

You can also create users who have restricted access to SAM. You can specify areas within SAM to which specific users can have access. You may have users to whom you would like to give access to backup and restore, or managing users, or handling the print spooler. Invoking SAM with the *-r* option will allow you to select a user to whom you want to give access to a SAM area and then select the specific area(s) to which you want to enable that user to have access. You can also give a user partial access to some areas, such as providing access to backup and recovery, but not providing access to handling automated backups. As you progress through the detailed descriptions of SAM areas in this chapter, you'll want to think about which of these areas may be appropriate for some of your users to access.

Author's Disclaimer - SAM Is a Moving Target

SAM is improving all the time. The SAM you are using on your HP-UX 11i system may have been enhanced and may therefore differ in some ways from what I cover in this chapter. The HP product line also improves requiring SAM to be enhanced. I earlier mentioned the Web-based enhancements being made to system administration tools in general. The Partition Manager that is now part of SAM was recently included to support many of the nPartition-based systems that are now part of the product line. SAM and all of the system administration tools are improving rapidly, so I'll have to keep updating this book and this chapter in particular.



Running Restricted SAM Builder

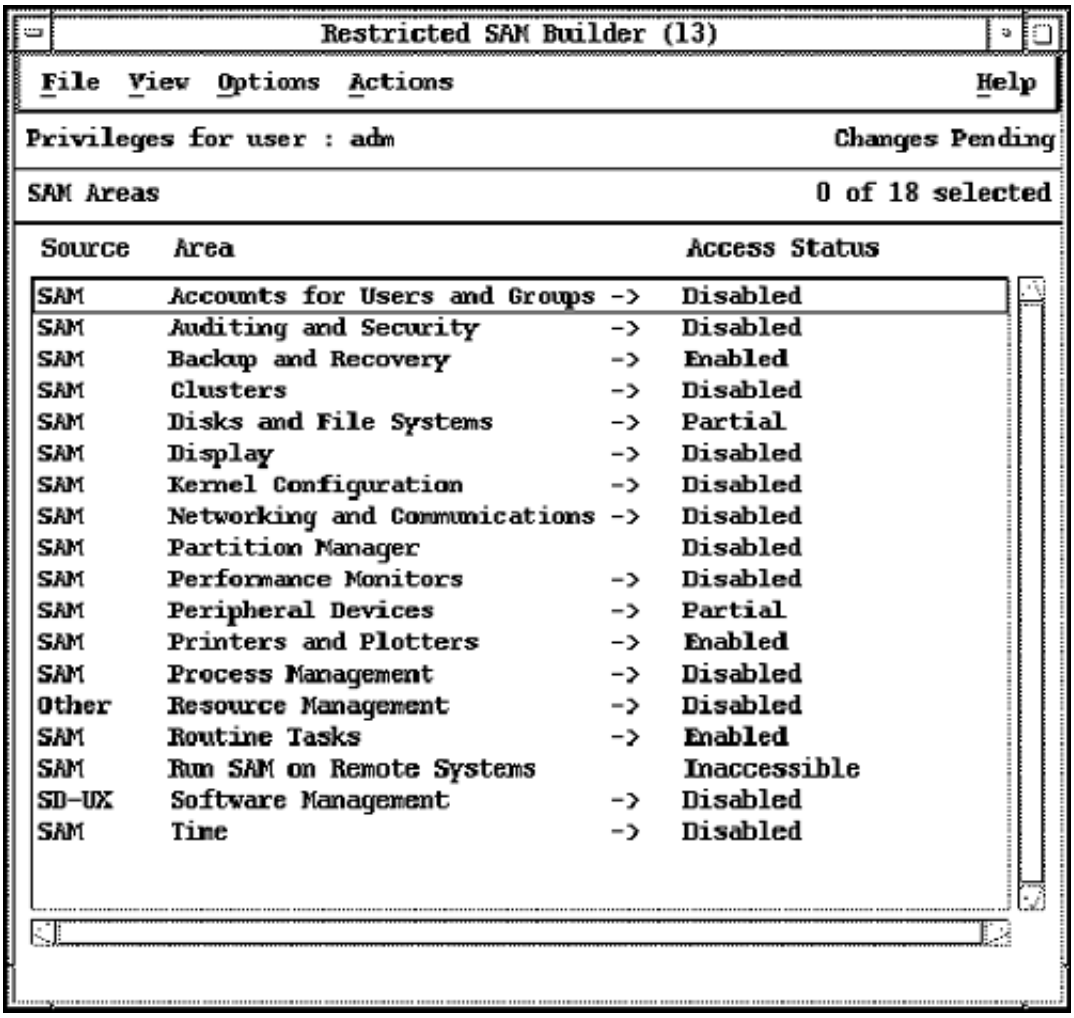
SAM can be configured to provide a subset of its overall functionality to specified users such as operators. You may, for instance, wish to give a user the ability to start a backup, but not the ability to manage disks and file systems. With the Restricted SAM Builder, you have control of the functional areas to which specified users have access.

When specifying the functionality you wish to give a user, you invoke SAM with the *-r* option, initiating a Restricted SAM Builder session. After you have set up a user with specific functionality, you can then invoke SAM with both the *-r* and *-f* options with the login name of a user you wish to test. The functionality of the user can be tested using these two options along with the login name.

Initially Setting User Privileges

When you invoke SAM with the *-r* option, you are first asked to select the user to whom you want to assign privileges. You will then be shown a list of default privileges for a new restricted SAM user. Figure 10-4 shows the default privileges SAM recommends for a new restricted user; note that custom SAM functional areas are disabled by default.

Figure 10-4. Restricted SAM Builder Screen



You can select from the *Actions* shown in Figure 10-4 to control access to functional areas. Of particular interest is the ability to save the privileges, which you may later use as a template for other users with *Load User Privileges* from the *Actions* menu.

The user will now have only access to the functional areas specified, and therefore will be able to execute tasks in only the areas to which they have access.

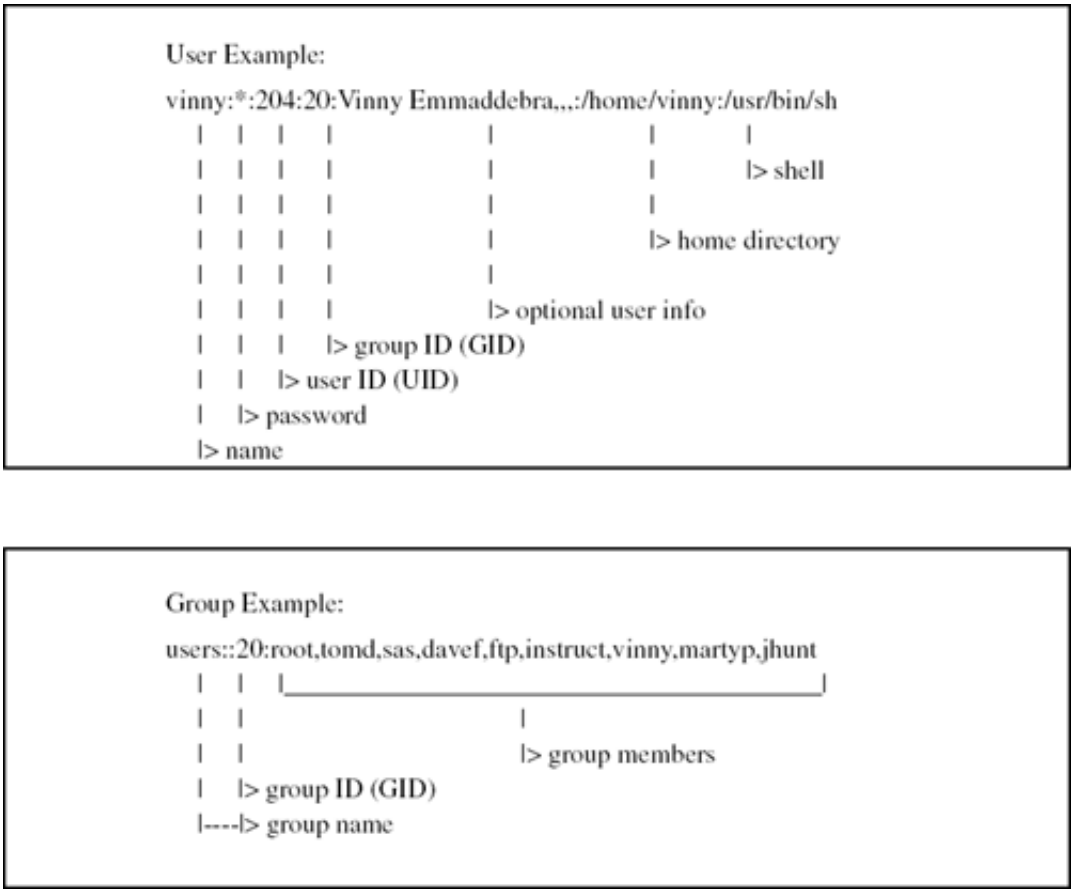
[Team LiB]

PREVIOUS NEXT

Accounts for Users and Groups

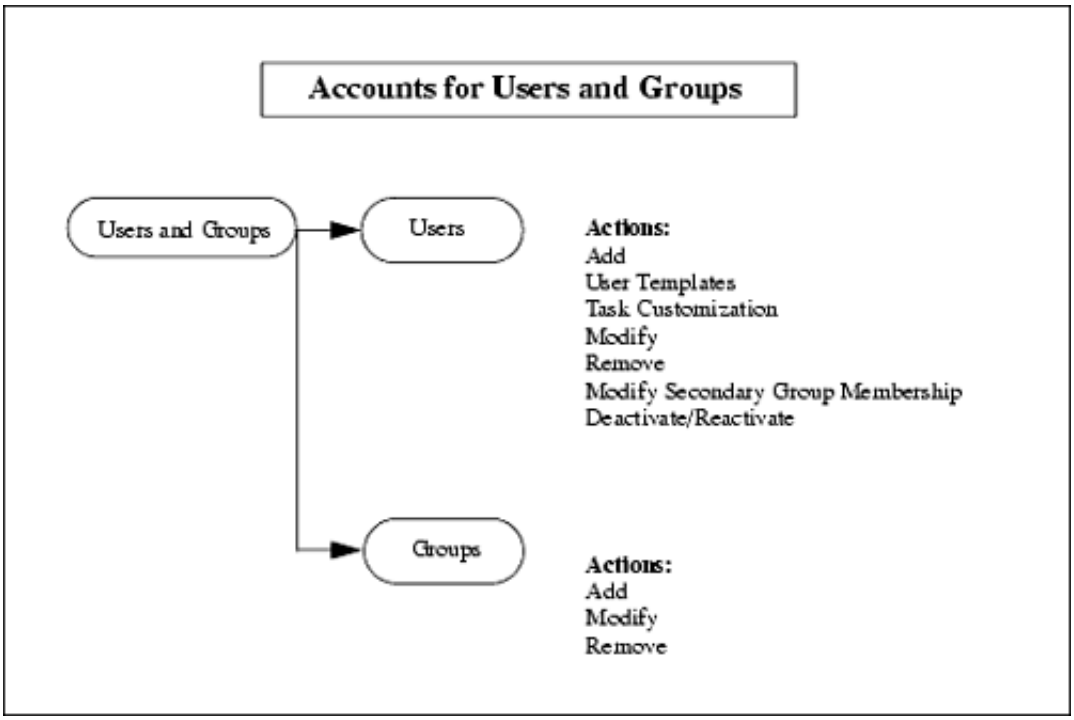
In Chapter 5, I explained the information that is associated with each user and group. There is an entry in the **/etc/passwd** file for each user and an entry in **/etc/group** for each group. To save you the trouble of flipping back to Chapter 5 Figure 10-5 is an example of a user entry from **/etc/passwd** and an example of a group entry from **/etc/group:**

Figure 10-5. Sample /etc/passwd and /etc/group Entry



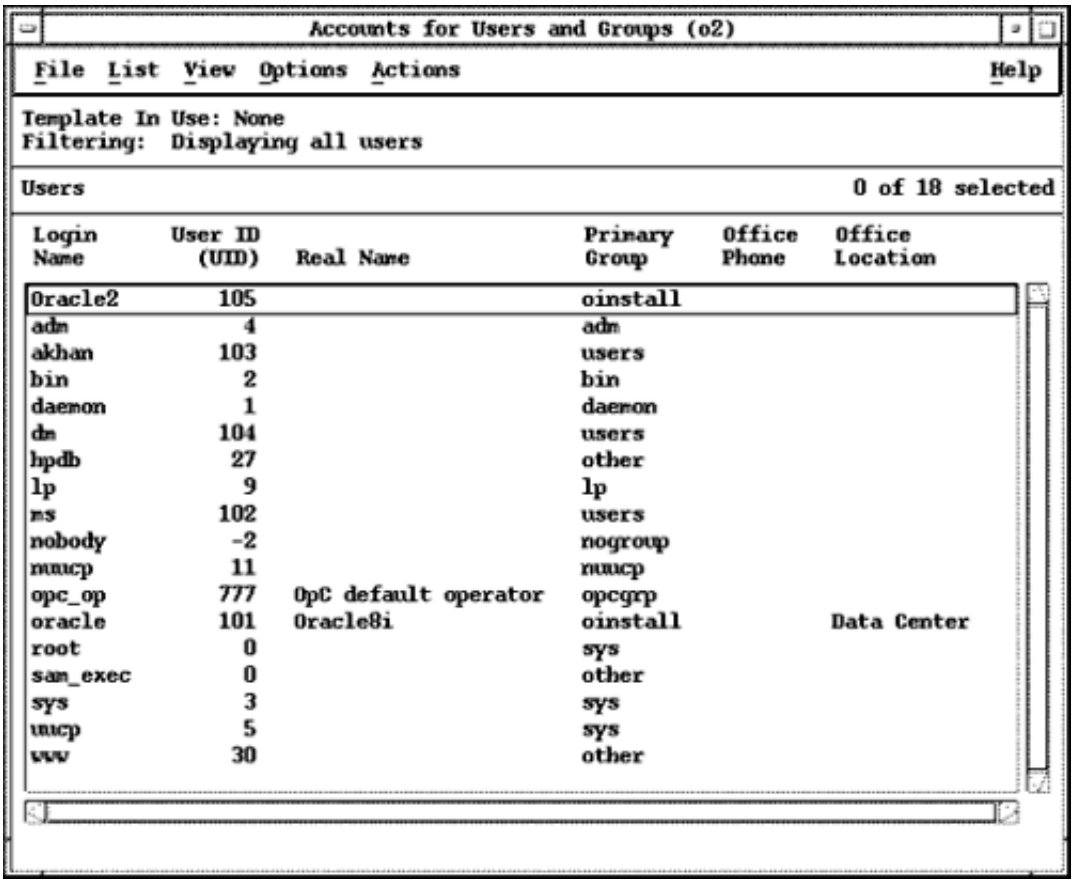
The *Accounts for Users and Groups* top-level SAM category, or area, has beneath it only two picks: *Groups* and *Users*. The menu hierarchy for "Users and Groups" is shown in Figure 10-6.

Figure 10-6. Accounts for Users and Groups



When you select *Accounts for Users and Groups* and then *Users* from the SAM menu, you are provided a list of all the users on your system. Figure 10-7 is a list of users provided by SAM for a system including Oracle users:

Figure 10-7. List of Users

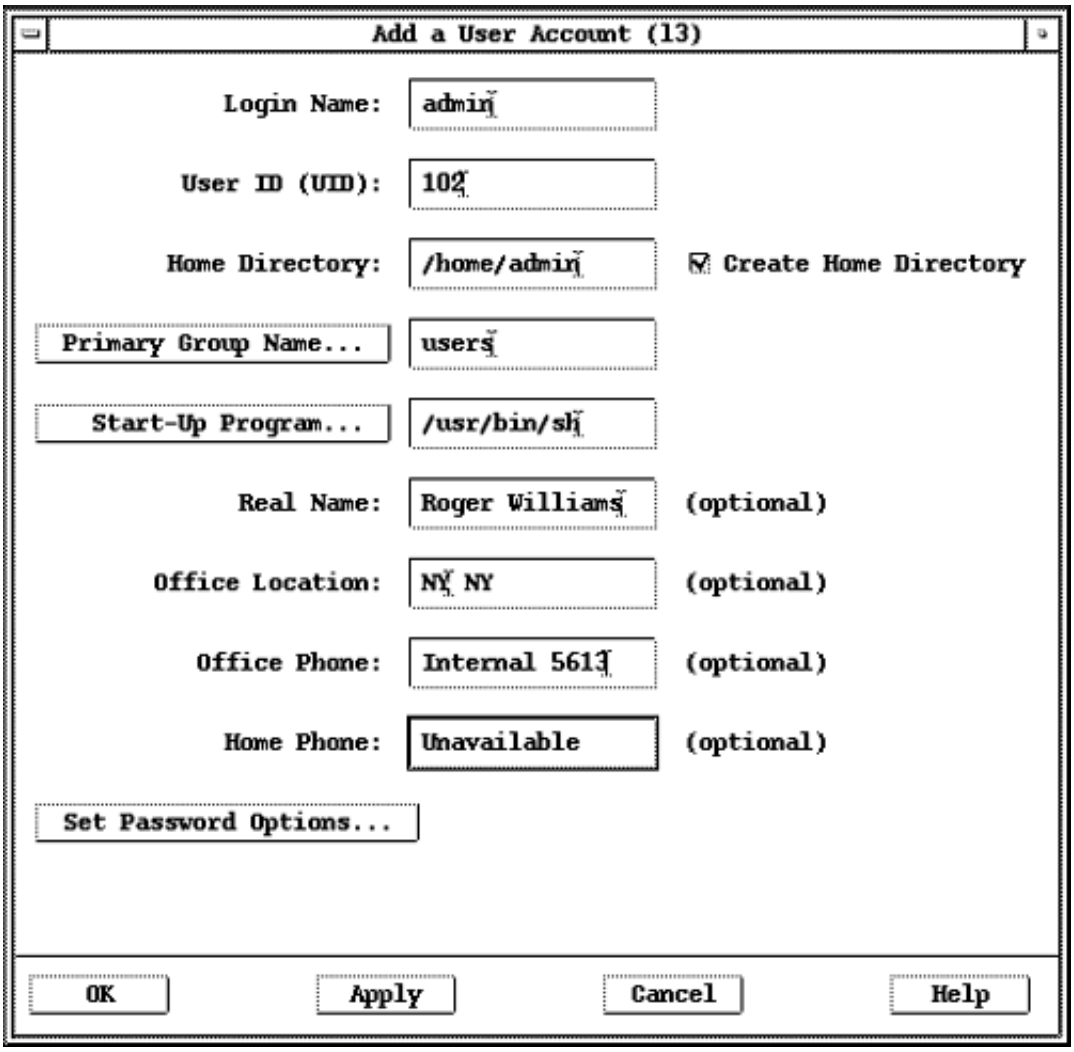


Adding a User

SAM is ideal for performing administration tasks related to users and groups. These are routine tasks that are not complex but require you to edit the **/etc/passwd** and **/etc/group** files, make directories, and copy default files, all of which SAM performs for you. Finally, take a minute to check what SAM has done for you, especially if you modify an existing user or group.

To add an additional user, you would select *Add* from the *Actions* menu under *Users* and then fill in the information as shown in Figure 10-8:

Figure 10-8. Example of Adding a New User



There are some restrictions when entering this information. For instance, a comma and colon are not permitted in the *Office Location* field.

When I tried to enter a comma, SAM informed me that this was not permitted.

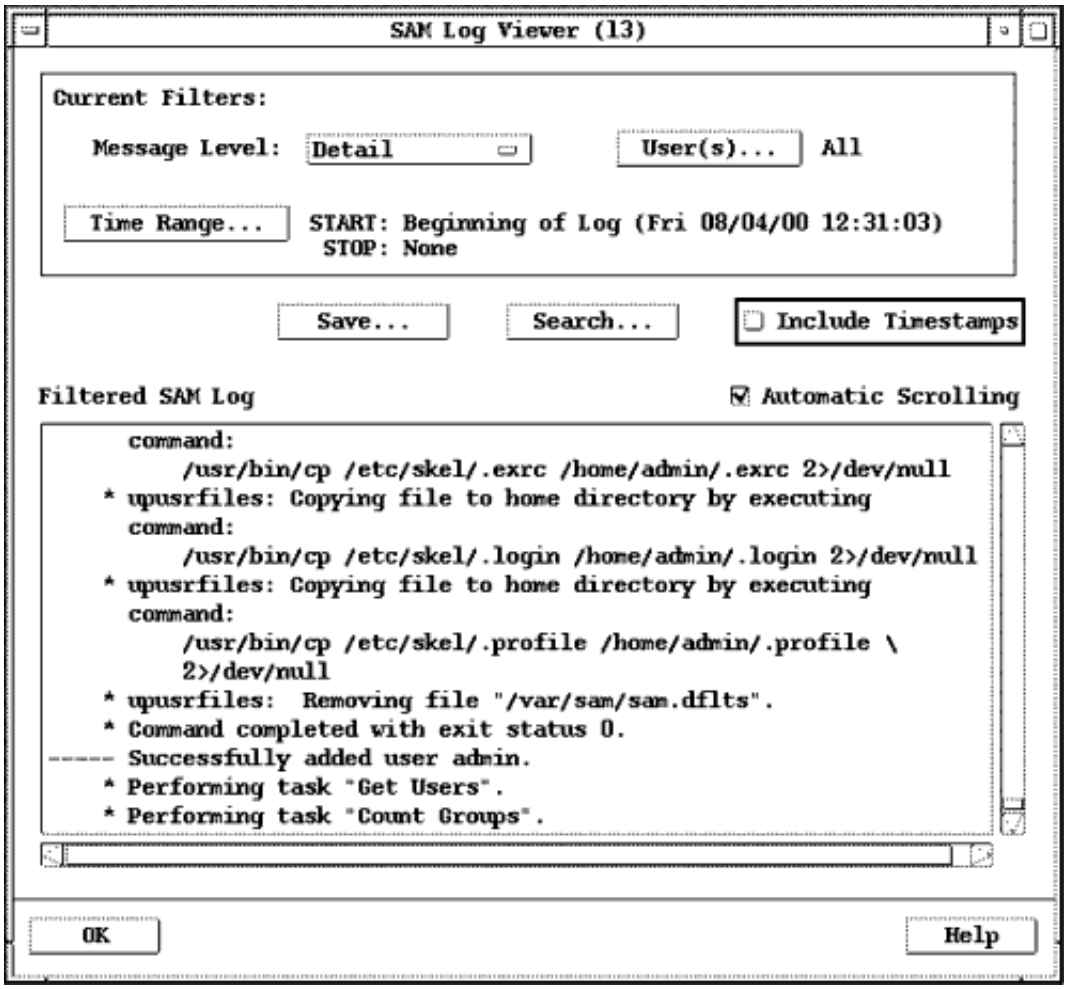
To see the **/etc/passwd** entry that was made I issued the following **cat** and **grep** commands that showed two users who had *admin* in their names:

```
# cat /etc/passwd | grep admin
webadmin:*:40:1::/usr/obam/server/nologindir:/usr/bin/false
admin::102:20:Roger  Williams,NY NY,Internal  5613,Unavailable:/home/admin:/usr/bin/sh
#
```

This user admin has no password entry, as indicated by the **::**, which is a poor practice. You want to make sure that there are passwords for all users.

You can view the log file that SAM produces with the menu selection *Options - View SAM Log.* Figure 10-9 show the end of the log file indicating that our user was successfully added:

Figure 10-9. SAM Log Viewer for Adding a User

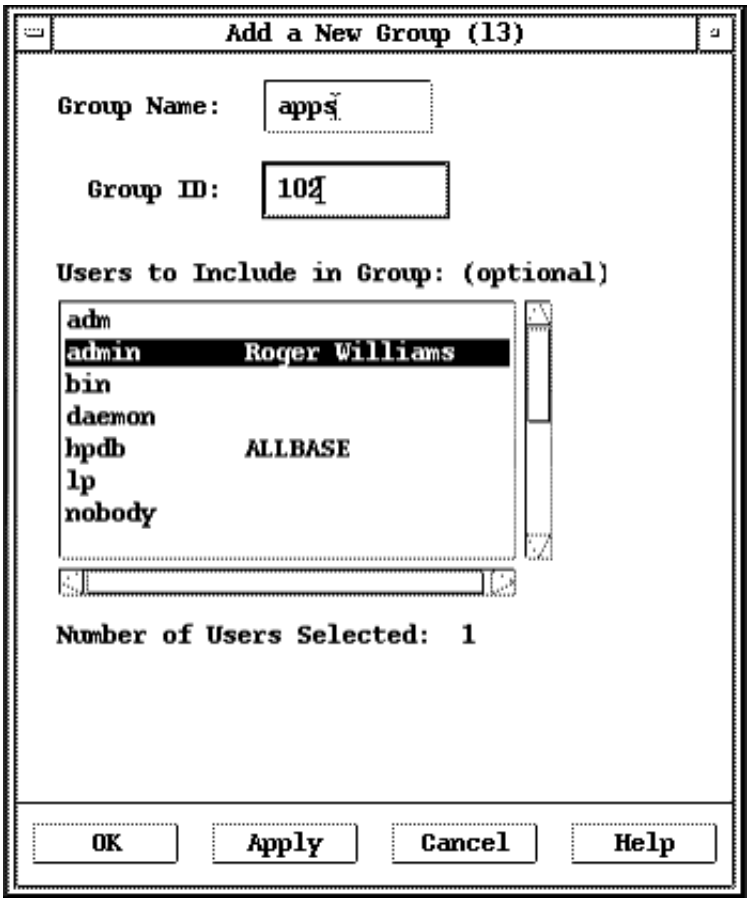


The scroll bar on the right-hand side of the SAM Log Viewer allows you to scroll to any point in the log file. We are viewing only the part of the log file that pertains to adding the user *Roger Williams*. You can select the level of detail you wish to view with the log file. The four levels are *Summary, Detail, Verbose,* and *Commands Only*. The level shown in Figure 10-9 is *Detail*. I like this level because you can see what has taken place without getting mired down in too much detail. When you view the log file, you typically see the calls to SAM scripts that SAM has made as well as other commands that SAM is issuing.

Adding a Group

Adding an additional group is similar to adding a new user. To add an additional group, you would select *Add* from the *Actions* menu under *Groups*. Figure 10-10 shows the Add a New Group window:

Figure 10-10. Example of Adding a New Group



In Figure 10-10, I added a new group called *apps* with a group ID of *102*, and into that group I added the user *admin* that we had earlier created.

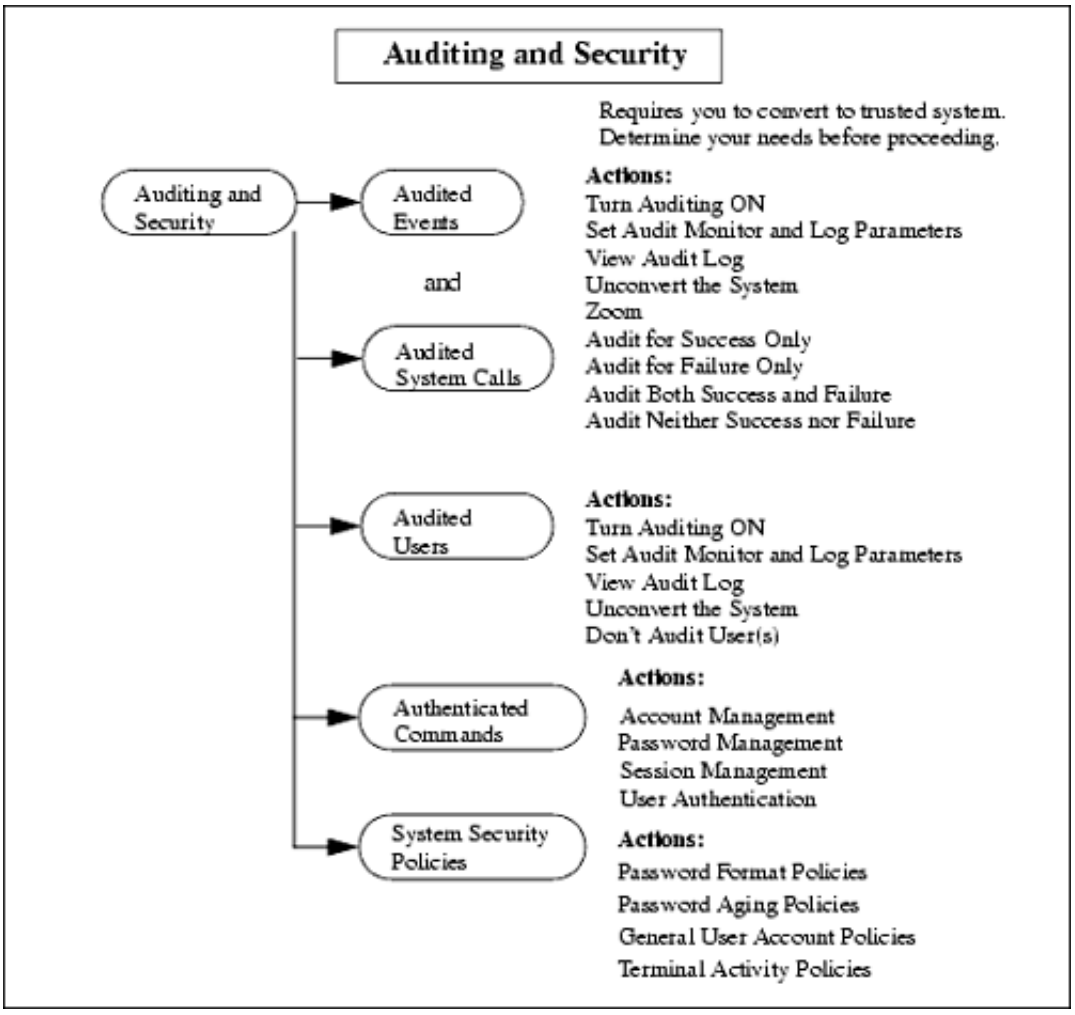
[Team LiB]

PREVIOUS NEXT

Auditing and Security

Under *Auditing and Security,* you manage the security of your system. This is becoming an increasingly important aspect of system management. Some installations care very little about security because of well-known, limited groups of users who will access a system. Other installations, such as those connected to the Internet, may go to great pains to make their systems into fortresses, with firewalls checking each and every user who attempts to access a system. I suggest that you take a close look at all the ramifications of security, and specifically a trusted system, before you enable security. You'll want to review the "Managing System Security" section of the "Administering A System" chapter of the *Managing Systems and Workgroups* manual, which replaces the 10.x manual *HP-UX System Administration Tasks Manual*. Although SAM makes creating and maintaining a trusted system easy, a lot of files are created for security management, which takes place under the umbrella of Auditing and Security. Among the modifications that will be made to your system, should you choose to convert to a trusted system, is the **/etc/rc.config.d/auditing** file, which will be updated by SAM. In addition, passwords in the **/etc/passwd** file will be replaced with "*," and the encrypted passwords are moved to a password database. All users are also given audit ID numbers. Figure 10-11 shows the menu hierarchy of *Auditing and Security*.

Figure 10-11. *Auditing and Security* Menu Structure



One choice to observe in Figure 10-11 is an *Actions* menu choice to *Unconvert the System*. This means to reverse the trusted system environment. I have tried this on various systems and it seems to work fine, but

you should have a good idea of what a trusted system can do for and to you before you make the conversion.

I hope that I have given you a reasonably good overview of *Auditing and Security*, because to investigate it yourself, you must first convert to a trusted system. Before you do, please read this section to get an idea of the functionality this will provide and then convert to a trusted system if you think there is adequate benefit.

Audited Events and Audited System Calls

Under *Audited Events,* you can select the particular events you wish to analyze and detect which may cause security breaches. Under *Audited System Calls,* you can monitor system calls. This option is a function of the trusted system to which you must convert in order to perform auditing. You may have in mind particular events and system calls that are most vital to your system's security that you wish to audit, and not bother with the balance. There are a number of events and system calls that you may wish to keep track of for security reasons.

Auditing these events gives you a detailed report of each event. The same is true of system calls. SAM uses the auditing commands of HP-UX such as **audsys, audusr, audevent, audomon**, and **audisp**, to perform auditing.

Audited Users

Under *Audited Users,* you can use the *Actions* menu to turn auditing on and off for specific users. Since the audit log files, which you can also control and view through the *Actions* menu, grow large very quickly, you may want to select specific users to monitor to better understand the type of user audit information that is created.

Authenticated Commands

This security feature is the ability to perform authentication based on user, password, session, or account. This industry-standard authentication framework is known as the Pluggable Authentication Module, or PAM. The PAM framework allows for authentication modules to be implemented without modifying any applications. Authentication is currently provided for CDE components, HP-UX standard commands, trusted systems, and DCE (the Distributed Computing Environment), as well as third-party modules.

System Security Policies

The most important part of HP-UX security is the policies you put in place. If, for instance, you choose to audit each and every system call, but don't impose any restrictions on user passwords, then you are potentially opening up your system to any user. You would be much better off restricting users and not worrying so much about what they're doing. Being proactive is more important in security than being reactive.

Password Aging Policies, when enabled, allows you to set:

- Time between Password Changes
- Password Expiration Time
- Password Expiration Warning Time
- Password Life Time
- Expire All User Passwords Immediately

General User Account Policies, when enabled, allows you to specify the time at which an account will become inactive and lock it. In addition, you can specify the number of unsuccessful login tries that are

permitted.

Terminal Security Policies allows you to set:

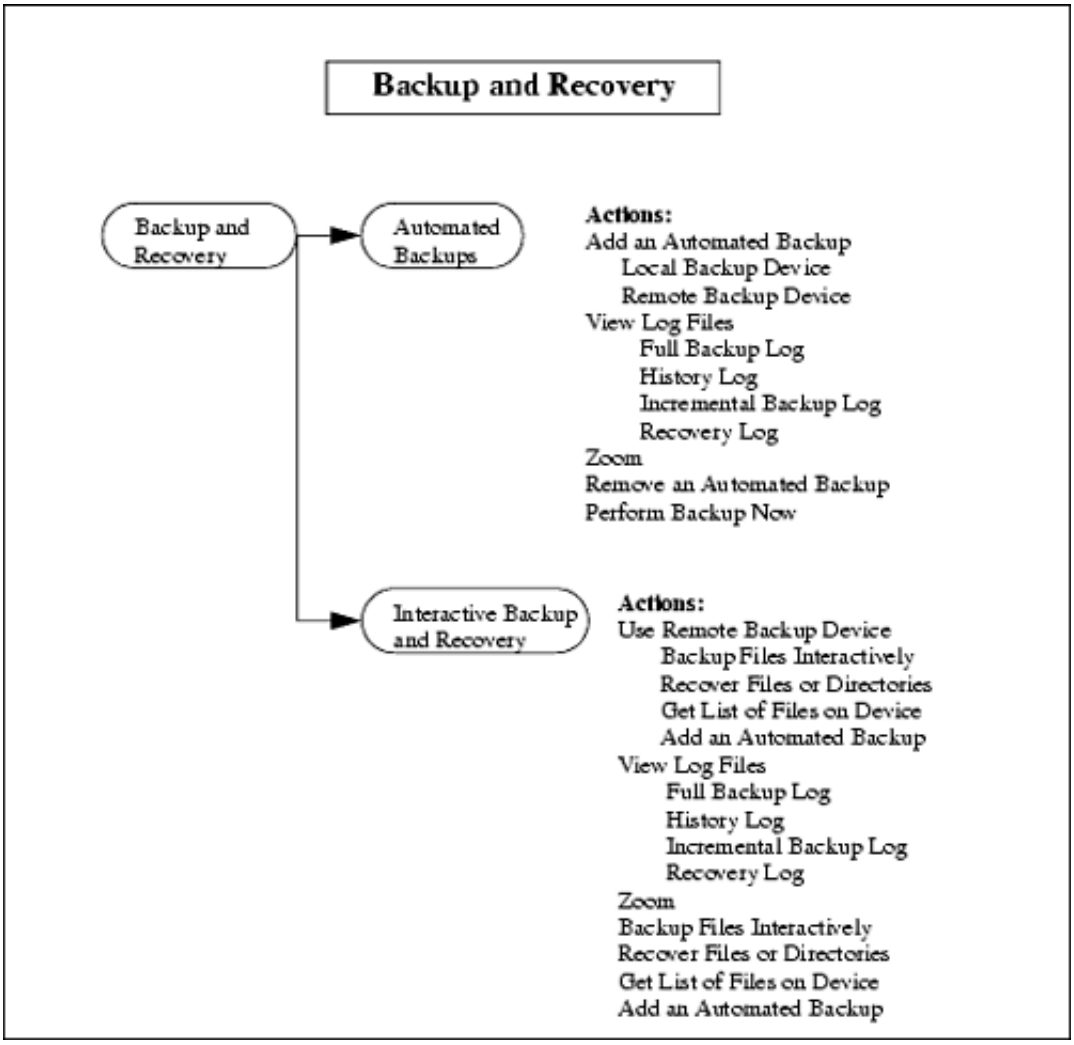
- Number of unsuccessful Login Tries Allowed
- Delay between Login Tries
- Login Timeout Value in Seconds
- Required Login upon Boot to Single-User State

[Team LiB]

Backup and Recovery

The most important activities you'll perform as a system administrator are system backup and recovery. The SAM team put a lot of thought into giving you all the options you need to ensure the integrity of your system through backup and recovery. You may also want to review Chapter 6, which covers various backup commands available on HP-UX. Figure 10-12 shows the hierarchy of the *Backup and Recovery* SAM menu:

Figure 10-12. *Backup and Recovery* Menu Structure



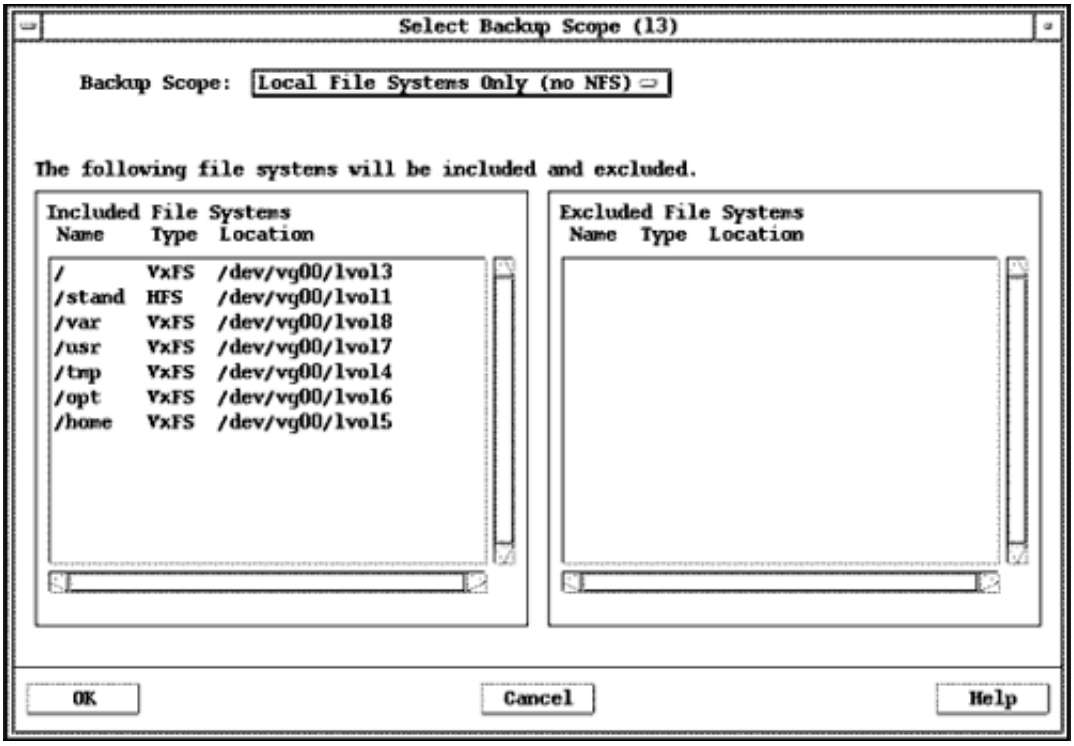
Scheduling a Backup

The first step is to enter the *Automated Backups* subarea. You won't see any automated backups appear in the list until you have specified one. Using the *Actions* menu and selecting *Add an Automated Backup,* you can specify all the information about your automated backup. When you select *Add an Automated Backup,* you have to specify whether your backup will be to a local or a remote backup device. You will have to enter information pertaining to the backup scope, backup device, backup time, and additional parameters.

Select Backup Scope

You can view the backup scope as the files that will be included and excluded from the backup. This can include Network File System (NFS)-mounted file systems as well. Figure 10-13 shows the window used to specify files to be included and excluded from a backup:

Figure 10-13. Selecting the Backup Scope



In the selections shown in Figure 10-13 are several directories specified under *Included Files*. These were selected with the *Local File Systems Only (no NFS)* option. We could easily have excluded files and directories from the backup scope as well.

Select Backup Device

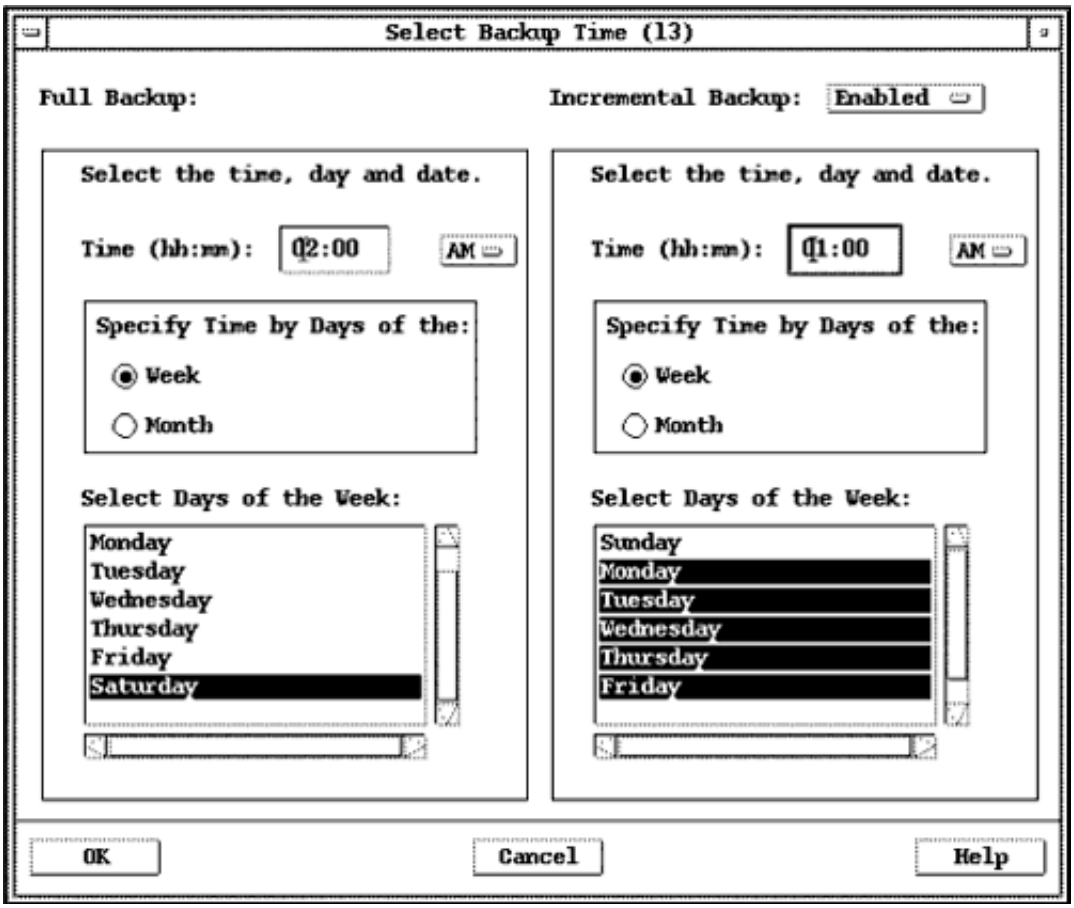
If you plan to back up to a local backup device, then those attached to your system will be listed and you select the desired device from the list.

If you plan to use a remote backup device, then you will be asked to specify the remote system name and device file.

Select Backup Time

As with the backup scope, you are provided with a window in which you can enter all the information about backup time for both full and incremental backups, as shown in Figure 10-14. If *Incremental Backup* is *Enabled,* then you must provide all pertinent information about both the full and incremental backup, as shown in the figure.

Figure 10-14. Selecting the Backup Time



A key point to keep in mind here is that the incremental backup that SAM creates for you includes files that have been changed *since the last full backup*. This means that you need only the full backup and last incremental backup to recover your system; that is, you do not need to restore the full backup and each incremental backup.

Set Additional Parameters



You can also specify additional parameters such as whether or not to create an index log, which I strongly suggest that you do, and to whom to mail the results of a backup. After specifying backup information, you can view the **crontab** entry that SAM has made for root for your backups. The **crontab** file is used to schedule jobs that are automatically executed by **cron. crontab** files are in the **/var/spool/cron/crontabs** directory. **cron** is a program that runs other programs at the specified time. **cron** reads files that specify the operation to be performed and the date and time it is to be performed. Since we want to perform backups on a regular basis, SAM will activate **cron**.

The format of entries in the **crontab** file is as follows:

minute hour monthday month weekday user name command

minute - the minute of the hour, from 0-59
hour - the hour of the day, from 0-23
monthday - the day of the month, from 1-31
month - the month of the year, from 1-12
weekday - the day of the week, from 0 (Sunday) - 6 (Saturday)
user name - the user who will run the command if necessary
(not used in the example)
command - specifies the command line or script file to run

You have many options in the **crontab** for specifying the *minute, hour, monthday, month,* and *weekday* to perform a task. You could list one entry in a field and then a space, several entries in any field separated by a comma, two entries separated by a dash indicating a range, or an asterisk, which corresponds to all possible entries for the field.

To list the contents of the **crontab** file, you would issue the command **crontab -l**. following *command*. SAM will create a **crontab** entry for any backups you specify.



You will see various *crontab* **commands** when you use the *SAM Log Viewer* to see what SAM has done for you to create the **crontab** files. For instance, if you change your backup plan, SAM will remove the old **crontab** file with the command:

```
$ crontab -r
```

This will remove the **crontab** file for the user from the **/var/spool/cron/crontabs** directory.

To place a file in the **crontab** directory, you would simply issue the **crontab** command and the name of the **crontab** file:

```
$ crontab crontabfile
```

You can schedule cron jobs using SAM. The section in this chapter covering Process Management has a section called Scheduling Cron Jobs.

Interactive Backup and Recovery

The *Interactive Backup and Recovery* subarea is used to perform a backup interactively or to restore information that was part of an earlier backup. When you enter this area, you are asked to select a backup device from a list that is produced, in the same way that you are asked to select a backup device when you first enter the *Automated Backups* subarea.

After selecting a device from the list, you may select an item from the *Actions* menu shown earlier. If you decide to use *Backup Files Interactively,* you are again provided a window from which you can specify files to be included and excluded from the backup. You are asked to *Select Backup Scope, Specify Tape Device Options*, and *Set Additional Parameters*. You are not, however, asked to *Select Backup Time,* since the backup is taking place interactively.

The steps in this area will vary, depending on the tape devices you have selected.

The index files can be reviewed from the *Actions* menu. These are stored in the **/var/sam/log** directory. The following shows the very top and bottom of an index file for an interactive backup:

```
#  1 /
#  1 /.profile
#  1 /.rhosts
#  1 /.sh_history
#  1 /.sw
#  1 /.sw/sessions
#  1 /.sw/sessions/swinstall.last
#  1 /.sw/sessions/swlist.last
#  1 /.sw/sessions/swmodify.last
#  1 /.sw/sessions/swreg.last
#  1 /.dt
#  1 /.dt/Desktop
#  1 /.dt/Desktop/Two                    TOP
#  1 /.dt/Desktop/Four
#  1 /.dt/Desktop/One
#  1 /.dt/Desktop/Three

                -
                -
                -

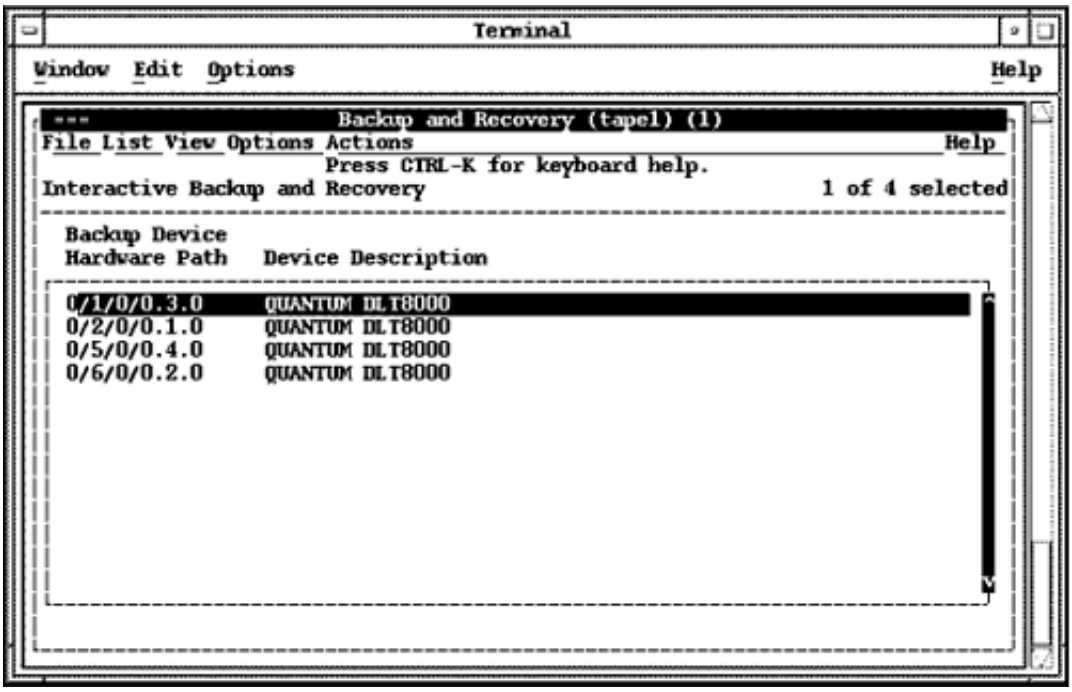
#  1 /var/uucp/.Log/uucico
#  1 /var/uucp/.Log/uucp
#  1 /var/uucp/.Log/uux
#  1 /var/uucp/.Log/uuxqt
```

```
#  1 /var/uucp/.Old
#  1 /var/uucp/.Status
#  1 /var/varspool/sw
#  1 /var/varspool/sw/catalog/dfiles
#  1 /var/varspool/sw/catalog/swlock
#  1 /var/varspool/sw/swagent.log
#  1 /var/yp
#  1 /var/yp/Makefile              BOTTOM
#  1 /var/yp/binding
#  1 /var/yp/securenets
#  1 /var/yp/secureservers
#  1 /var/yp/updaters
#  1 /var/yp/ypmake
#  1 /var/yp/ypxfr_1perday
#  1 /var/yp/ypxfr_1perhour
#  1 /var/yp/ypxfr_2perday
```

Performing a Restore

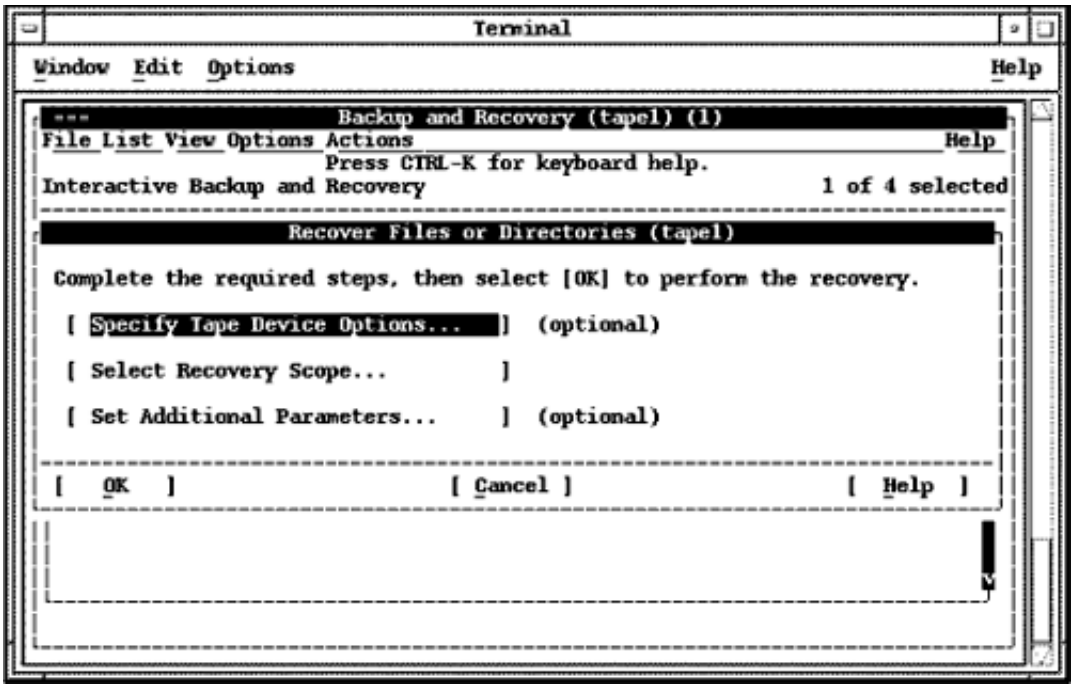
A full or incremental backup, however, is only as good as the files it restores. To retrieve a file from the backup tape, you specify a backup device and then many options related to the backup. Figure 10-15 shows one device selected from among four DLT units connected to a system:

Figure 10-15. Selecting From Among Four DLT Units



After selecting a device, there are a number of options from which you can select, including those shown in Figure 10-16:

Figure 10-16. Recover Files Interactively



In this window you supply information in three areas: *Select Recovery Scope; Specify Tape Device Options*; and *Set Additional Parameters*. The device options you specify will depend on the tape device you are using.

Select Recovery Scope allows you to either enter a file name that contains the files to be recovered or manually list the files to be included in the recovery. You can optionally list files to be excluded from the recovery as well.

A list of tape device files is provided in *Specify Tape Device Options,* from which you can select the tape device. In this step, you may select the tape device file; in other cases, you might make selections such as a magneto-optical surface, or you may have nothing to select at all.

Under *Set Additional Parameters,* you can select any of the following options:

Overwrite Newer Files

Preserve Original File Ownership

Recover Files Using Full Path Name

Place Files in Non-Root Directory

After you make all the desired selections, the recovery operation begins. If a file has been inadvertently deleted and you wish to restore it from the recovery tape, you would select the Preserve Original File Ownership and Recover Files Using Full Path Name options. You will be given the status of the recovery as it takes place and may also View Recovery Log from the *Actions* menu after the recovery has completed. If you View Recovery Log, you will receive a window that provides the name of the index log and the names of the files recovered.

[Team LiB]

PREVIOUS NEXT



Clusters

High availability clusters can be managed through SAM. Such tasks as adding applications and adding application groups can be performed using SAM. Since high availability is a highly customized aspect of a computing environment, I won't cover this area of SAM. If, however, you have to manage clusters of systems, you will want to review this SAM area to see if it would help in your cluster-related work. See Chapter 19 for a discussion of MC/ServiceGuard.



Disks and File Systems

Disks and File Systems helps you manage disk devices, file systems, logical volumes, swap, and volume groups (you may also manage other HP disk devices such as XP and disk arrays through SAM if you have these installed on your system). There is no reason to manually work with these, since SAM does such a good job of managing these for you. Figures 10-17 and 10-18 show the hierarchy of *Disks and File Systems*:

Figure 10-17. *Disks and File Systems* Menu Structure

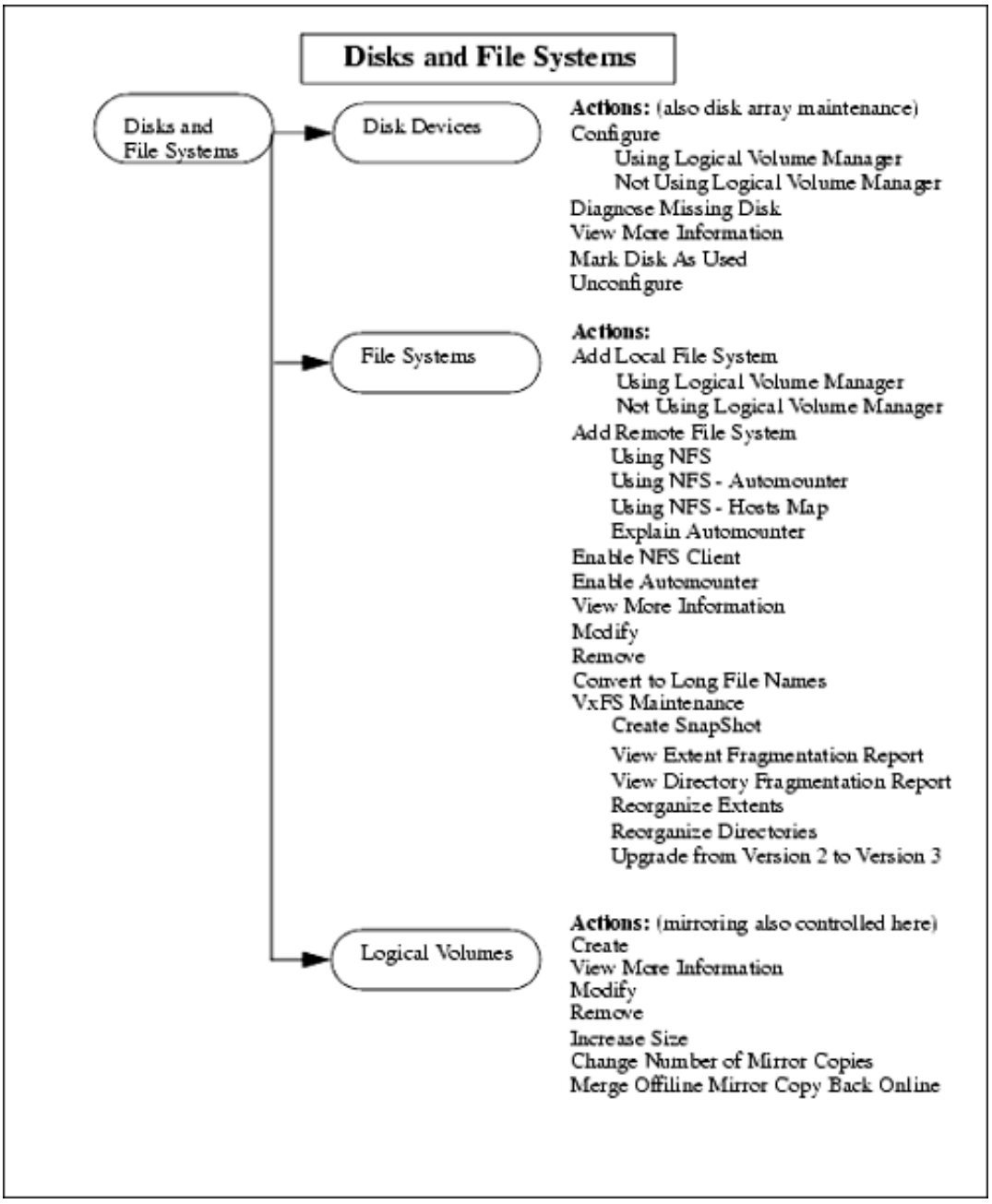
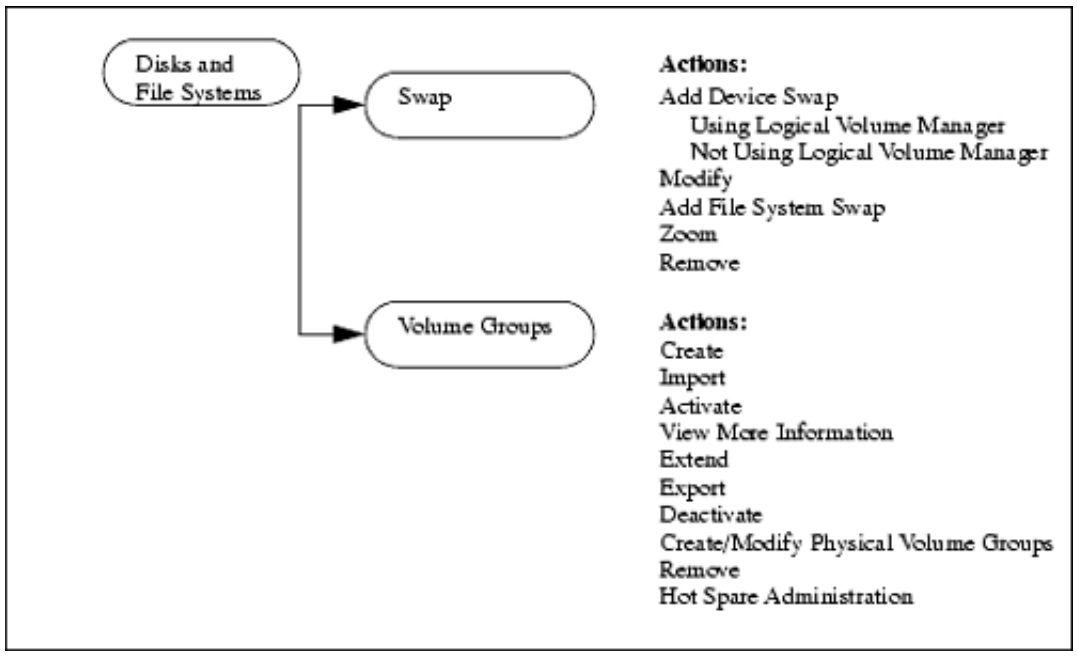


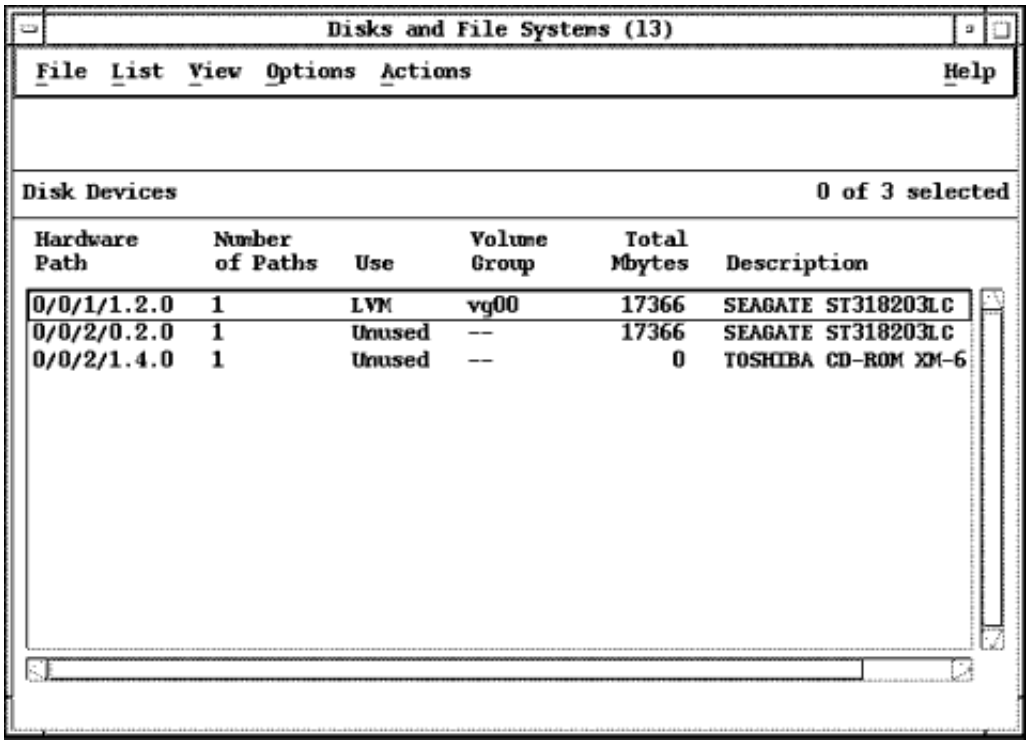
Figure 10-18. *Disks and File Systems* Menu Structure (continued)



Disk Devices

When you enter this subarea, SAM shows you the disk devices connected to your system. Figure 10-19 shows a listing of the disks for a Series 800 unit:

Figure 10-19. *Disk Devices* Window



The first two entries refer to internal disks in an L-Class system. The third entry is a CD-ROM drive. Let's compare this output to what we would see when we run **ioscan**:

```
$ /usr/sbin/ioscan -funC disk

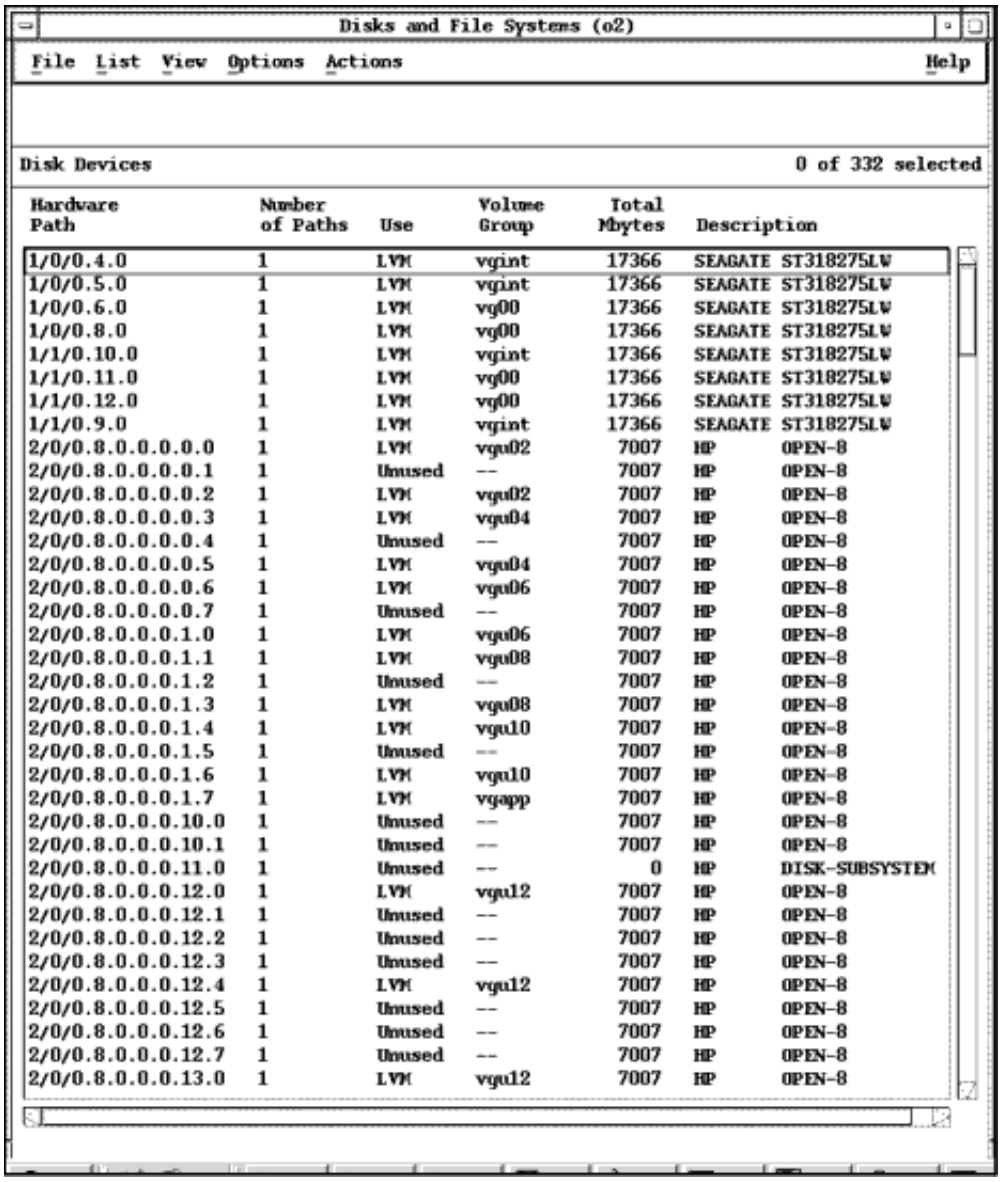
Class     I  H/W Path     Driver S/W State   H/W Type     Description
===================================================================
disk      1  0/0/1/1.2.0  sdisk CLAIMED     DEVICE       SEAGATE ST318203LC
                         /dev/dsk/c1t2d0   /dev/rdsk/c1t2d0
disk      2  0/0/2/0.2.0  sdisk CLAIMED     DEVICE       SEAGATE ST318203LC
                         /dev/dsk/c2t2d0   /dev/rdsk/c2t2d0
disk      3  0/0/2/1.4.0  sdisk CLAIMED     DEVICE      TOSHIBA CD-ROM XM-6201TA
                         /dev/dsk/c3t4d0   /dev/rdsk/c3t4d0
```

The **ioscan** output jives with the disk information produced by SAM. The SAM output also indicates that the first disk is used and the second unused, which **ioscan** does not show.

We can now add one of the unused disks in SAM by selecting *Add* from the *Actions* menu. Using Logical Volume Manager, we can create a new volume group or select the volume group to which we wish to add the new disk. We would then select the new logical volumes we wanted on the volume group or extend the size of existing logical volumes. Other information such as the mount directory and size of the logical volume would be entered as well.

Most disks connected to your system can be managed through SAM. Figure 10-20 shows a SAM screen shot of a V-Class system with an XP-256 attached to it:

Figure 10-20. *Disk Devices* Window with XP-256 Disks Shown

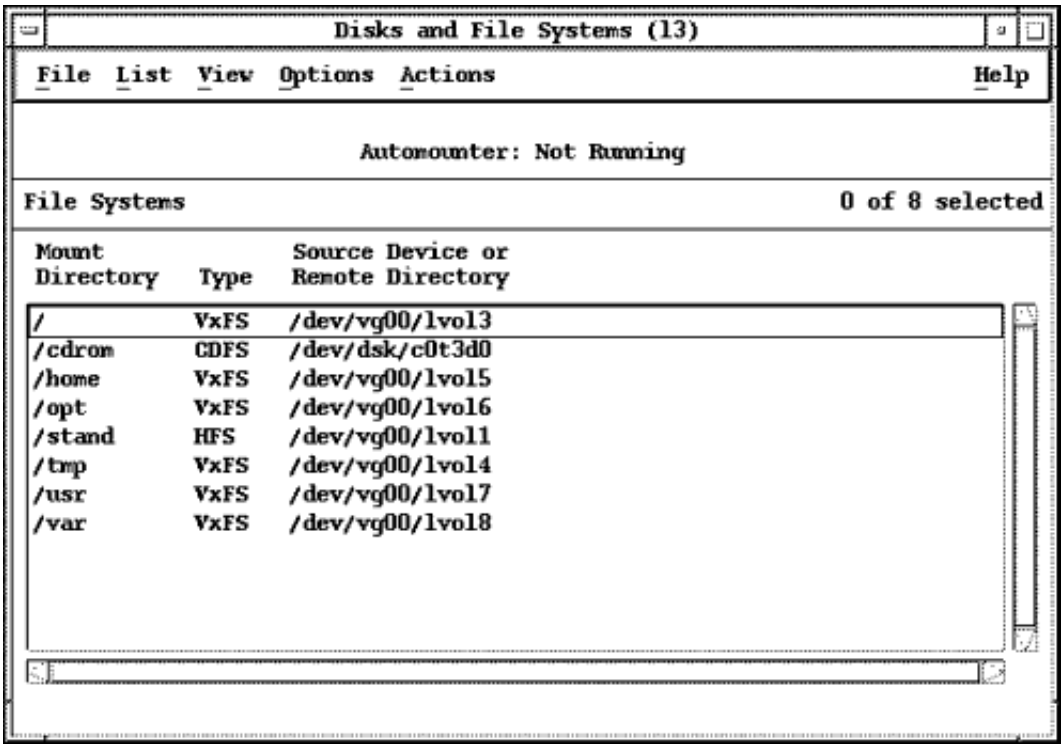


A small subset of the total disks attached is shown in the screen shot. The first eight disks are internal to the V-Class and the remainder are XP disks. All of the devices shown in this figure can be managed through SAM.

File Systems

File Systems shows the *Mount Directory, Type* of file system, and *Source Device or Remote Directory*. Figure 10-21 shows the information you see when you enter *File Systems* for the L-Class system used in earlier examples:

Figure 10-21. *File Systems* Window



At this level, you can perform such tasks as *Add Local File System* and *Add Remote File System,* and you can perform *VxFS Maintenance* from the *Actions* menu.

Several types of file systems may be listed under the Type column. The most common are:

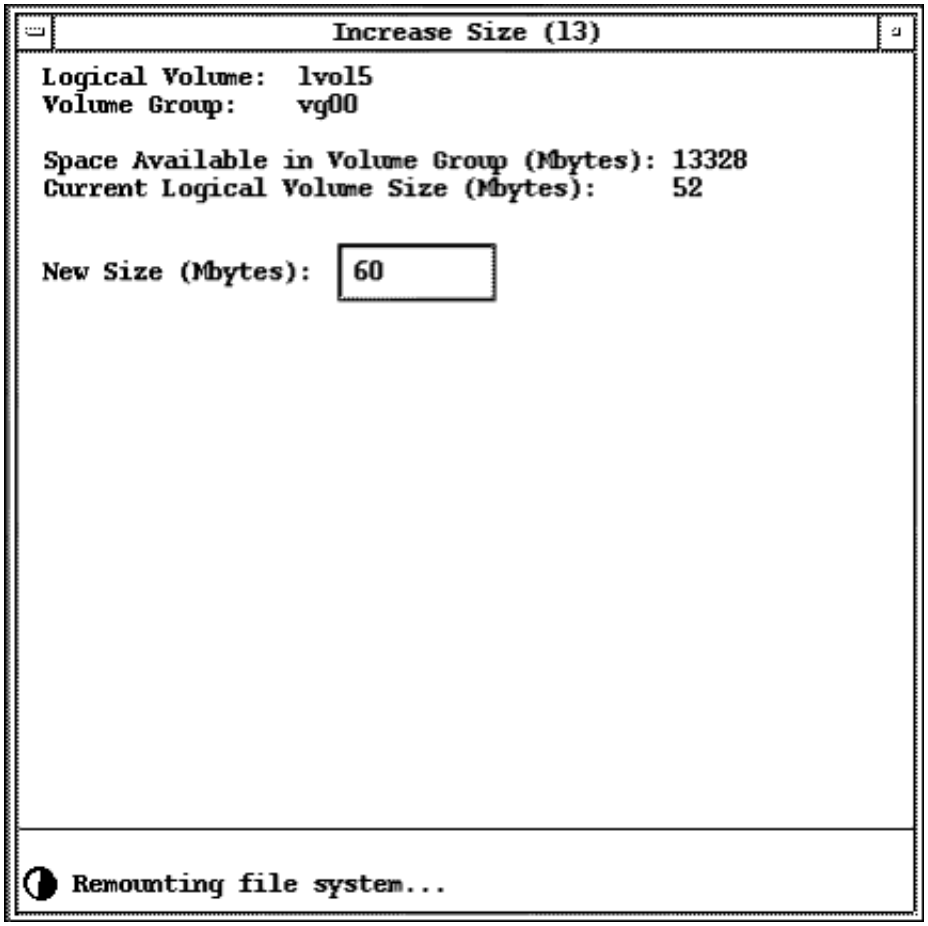
Auto-Indirect	Directory containing auto-mountable remote NFS file systems. You may see the **/net** directory here if you have auto-mounter running.
Auto-Mount	Auto-mountable remote NFS file system.
CDFS	CD-ROM file system if it is currently mounted. If, for instance, you have a CD-ROM mounted as /SD_CDROM, you will see this as type CDFS in the list.
HFS	Local HFS file system. These are local HFS file systems that are part of your system. HP's version of the UNIX File System. This was the most common file system under earlier versions of HP-UX.
NFS	Remote NFS file system that is currently mounted.
LOFS	Loopback file system that allows you to have the same file system in multiple places.
VxFS	Local Journaled File System (JFS). This is the HP-UX implementation of the Veritas journaled file system (VxFS), which supports fast file system recovery. JFS is the default HP-UX file system.

Add Local File System allows you to mount an unmounted, local file system. *Add Remote File System* gives you the ability to mount a file system from another host. The *VxFS Maintenance* subarea is where you can perform some file system maintenance tasks on your JFS file system. Here, you can create reports on extent and directory fragmentation. Once these are reviewed and you find that you do indeed need to perform maintenance, you can choose the option to reorganize either the extents or the directory.

Logical Volumes

You can perform several functions related to logical volume manipulation in SAM. Such tasks as *Create, Modify, Remove,* and *Increase Size* can be performed in SAM. Figure 10-22 shows increasing the size of lvol5 (**/home**) from 52 MBytes to 6 MBytes:

Figure 10-22. *Increase Size* Window



SAM will increase the size of the logical volume only if it can be unmounted. Viewing the log file after this task has been completed shows that SAM ran such commands as **/sbin/lvextend** and **/sbin/extendfs** to extend the size of the logical volume and file system, and **/usr/sbin/umount** and **/usr/sbin/mount** to unmount and mount the file system.

See the Logical Volume Manager detail in Chapter 3 for definitions of Logical Volume Manager terms. There is also a description of some Logical Volume Manager commands.

Increasing the Size of a Logical Volume in SAM



SAM may create a unique set of problems when you attempt to increase the size of a logical volume. Problems may be encountered when increasing the size of a logical volume if it can't be unmounted. If, for instance, you wanted to increase the size of the **/opt** logical volume, it would first have to be unmounted by SAM. If SAM can't **umount /opt**, you will receive a message from SAM indicating that the device is busy. You can go into single-user state, but you will have to have some logical volumes mounted, such as **/usr** and **/var,** in order to get SAM to run. You would then need to reboot your system with `shutdown -r` after you have completed your work. This works for directories such as **/opt**, which SAM does not need in order to run.

Alternatively, you could exit SAM and kill any processes accessing the logical volume you wish to extend the size of, and then manually unmount that logical volume. You could then use SAM to increase the size of the logical volume.

The HP OnLineJFS add-on product allows you to perform many of these LVM functions without going into single-user mode. For example, with OnLineJFS, logical volumes and file systems are simply expanded with

the system up and running and no interruption to users or processes.

Swap

Both device swap and file system swap are listed when you enter *Swap*. Listed for you are the *Device File/Mount Directory, Type, Mbytes Available*, and *Enabled*. You can get more information about an item by highlighting it and selecting *Zoom* from the *Actions* menu.

Volume Groups

Listed for you when you enter volume groups are *Name, Mbytes Available, Physical Volumes*, and *Logical Volumes*. If you have an unused disk on your system, you can extend an existing volume group or create a new volume group. This window is useful to see how much disk space within a volume group has not been allocated yet. Another function here is the ability to import volume groups from other systems or ready a volume group for export to a remote system. You would use this when moving a volume group contained on an entire disk drive or set of disk drives from one system to another.

Display

In this SAM area, you can perform work related to the graphics display(s) on your system. This work is self-explanatory in SAM, so I won't cover this functional area of SAM.

[Team LiB]



Kernel Configuration

Your HP-UX kernel is a vitally important part of your HP-UX system that is often overlooked by HP-UX administrators. Perhaps this is because administrators are reluctant to tinker with such a critical and sensitive part of their system. Your HP-UX kernel, however, can have a big impact on system performance, so you want to be sure that you know how it is configured. This doesn't mean that you have to make a lot of experimental changes, but you should know how your kernel is currently configured so that you can assess the possible impact that changes to the kernel may have on your system.

Be sure to look at **kcweb** in the kernel chapter. **kcweb** is a web-based tool for working wtih the kernel. There are plans to make many of the system administration tools Web-based, and **kcweb** gives you a good idea of the type of tools being developed.

SAM allows you to view and modify the four basic elements of your HP-UX kernel. There is a great deal of confusion among new HP-UX system administrators regarding these four elements. Before I get into the details of each of these four areas, I'll first give you a brief description of each.

- *Configurable Parameters* - These are parameters that have a value associated with them. When you change the value, there is a strong possibility that you will affect the performance of your system. An example of a *Configurable Parameter* is **nfile**, which is the maximum number of open files on the system. Many configurable parameters in HP-UX 11i can be modified and included in the kernel without a reboot required. We'll cover this in one of the upcoming examples.

- *Drivers* - Drivers are used to control the hardware on your system. You have a driver called **CentIF** for the parallel interface on your system, one called **sdisk** for your SCSI disks, and so on.

- *Dump Devices* - A dump device is used to store the contents of main memory in the event that a serious kernel problem is encountered. If no dump device is configured, then the contents of main memory are saved on the primary swap device, and this information is copied into one of the directories (usually **/var/adm/crash**) when the system is booted. It is not essential that you have a dump device, but the system will boot faster after a crash if you have a dump device because the contents of main memory don't need to be copied to a file after a crash. A dump device is different from a swap device.

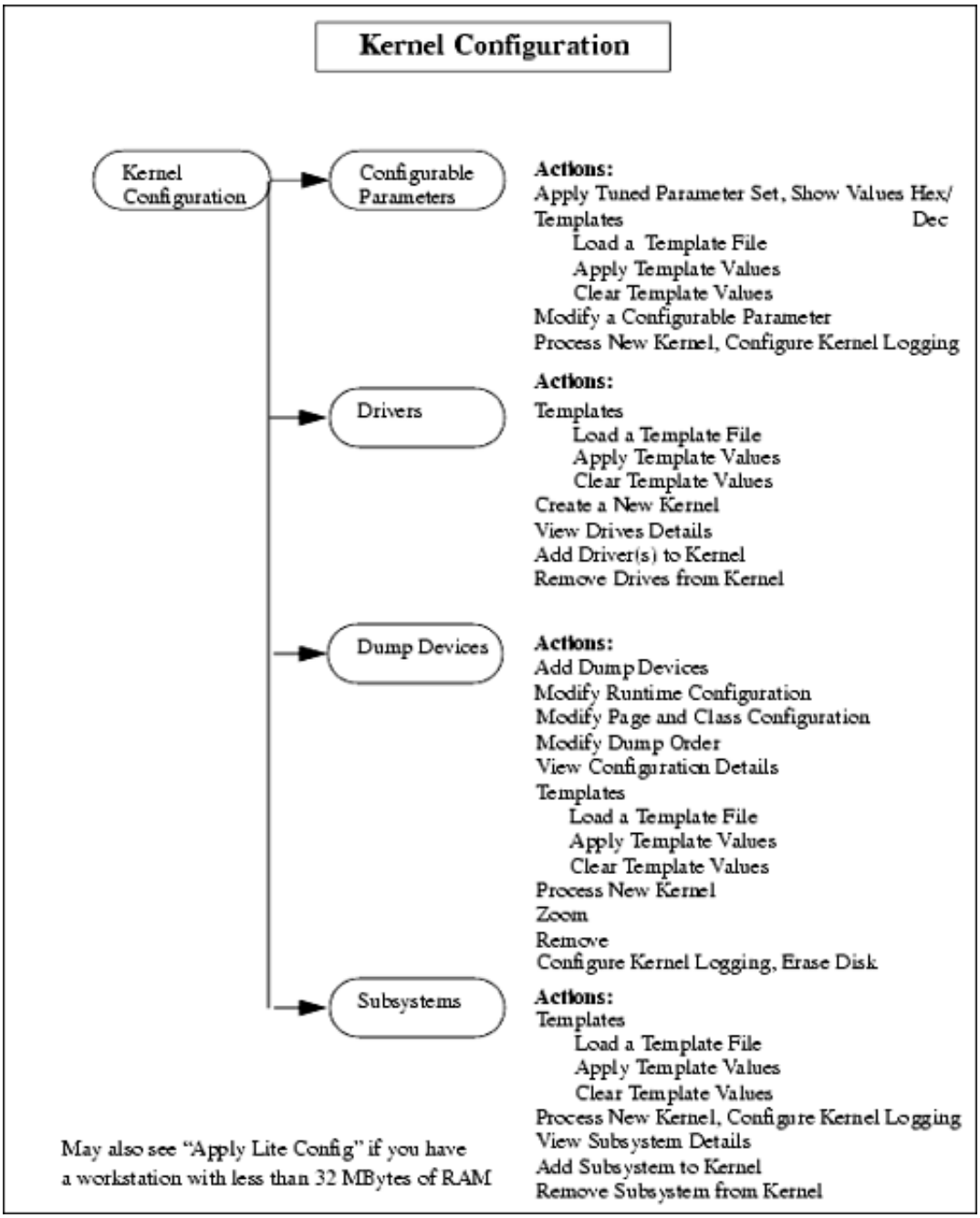
- *Subsystems* - A subsystem is different from a driver. A subsystem is an area of functionality or support on your system such as **CD-ROM/9000**, which is CD-ROM file system support; **LVM,** which is Logical Volume Manager support; and so on.

When you go into one of the four subareas described above, the configuration of your system for the respective subarea is listed for you. The first thing you should do when entering *Kernel Configuration* is to go into each of the subareas and review the list of information about your system in each.

In *Kernel Configuration* is a *current* kernel and *pending* kernel. The *current* kernel is the one you are now running, and the *pending* kernel is the one for which you are making changes.

Figure 10-23 shows the SAM menu hierarchy for *Kernel Configuration*.

Figure 10-23. *Kernel Configuration* Menu Structure



Configurable Parameters

Selecting *Configurable Parameters* lists all your configurable kernel parameters. For each configurable parameter, the following information is listed:

- Name - Name of the parameter.
- Current Value - Value of the parameter in **/stand/vmunix**.
- Pending Value - Value of the parameter in the kernel to be built.

- Type - Shows whether parameters are part of the static kernel or a loadable module.
- Associated Module - Lists module parameter is part of if type is loadable.
- Description - A few words describing the parameter.

You can then take a number of *Actions,* including the following:

Apply Tuned Parameter Set

: Several sets of configurable parameters have been tuned for various environments. When you select this from the Actions menu, the tuned parameter sets on your system, such as a database server system, are listed for you and you can select from among them.

Templates
: You can select a kernel template to load that is basically a different kernel configuration from the one you are currently running.

Process New Kernel

: After making whatever changes you like to the *Pending Value* of a configurable parameter, you can have SAM create a new kernel for you.

Modify Configurable Parameter

: You can change the value of a parameter in the *pending* kernel. You simply highlight a parameter and select this from the *Actions* menu.

Modifying a configurable parameter is made much easier by SAM. But although the logistics of changing the parameter are easier, determining the value of the parameter is still the most important part of this process.

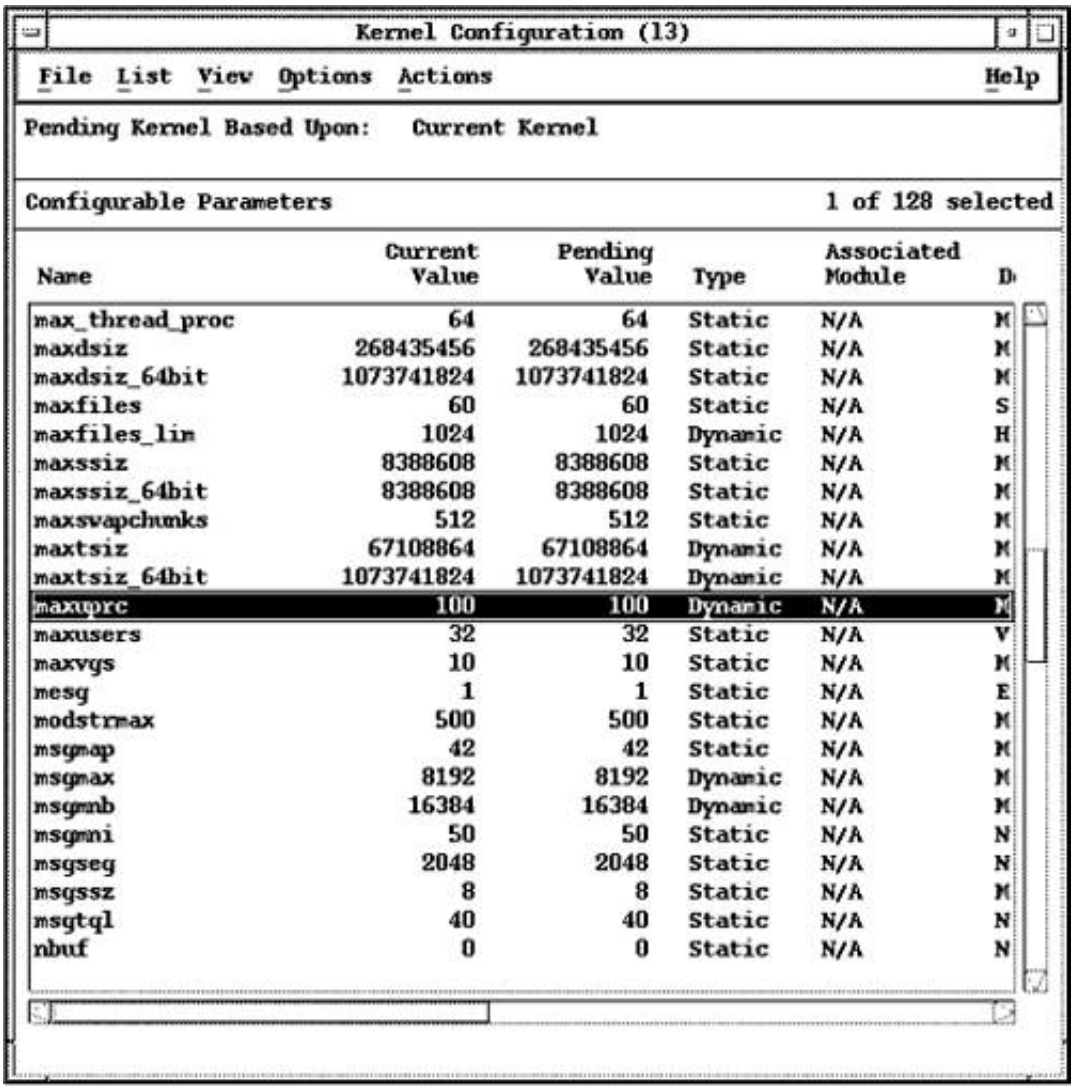
Many applications recommend modifying one or more of these parameters for optimal performance of the application. Keep in mind, though, that many of these parameters are related; modifying one may adversely affect another parameter. Many applications will request that you change the *maxuprc* to support more processes. Keep in mind that if you have more processes running, you may end up with more open files and also you may have to change the *maxfiles* per process. If you have a system primarily used for a single application, you can feel more comfortable in modifying these. But if you run many applications, make sure that you don't improve the performance of one application at the expense of another.

When you do decide to modify the value of a configurable parameter, be careful. The range on some of these values is broad. The *maxuprc* (maximum number of user processes) can be reduced as low as three processes. I can't imagine what a system could be used for with this low a value, but SAM ensures that the parameter is set within supported HP-UX ranges for the parameter. "Let the administrator beware" when changing these values. You may find that you'll want to undo some of your changes. Here are some tips: Keep careful notes of the values you change, in case you have to undo a change. In addition, change as few values at a time as possible. That way, if you're not happy with the results, you know which configurable parameter caused the problem.

In Chapter 4, when covering the kernel, we manually modified the *maxuprc* kernel parameter. Since this parameter is dynamic, there was not a reboot required in order for it to take effect. Let's now use SAM to modify this parameter and again demonstrate that since it is dynamic, no reboot is required if it is modified with SAM.

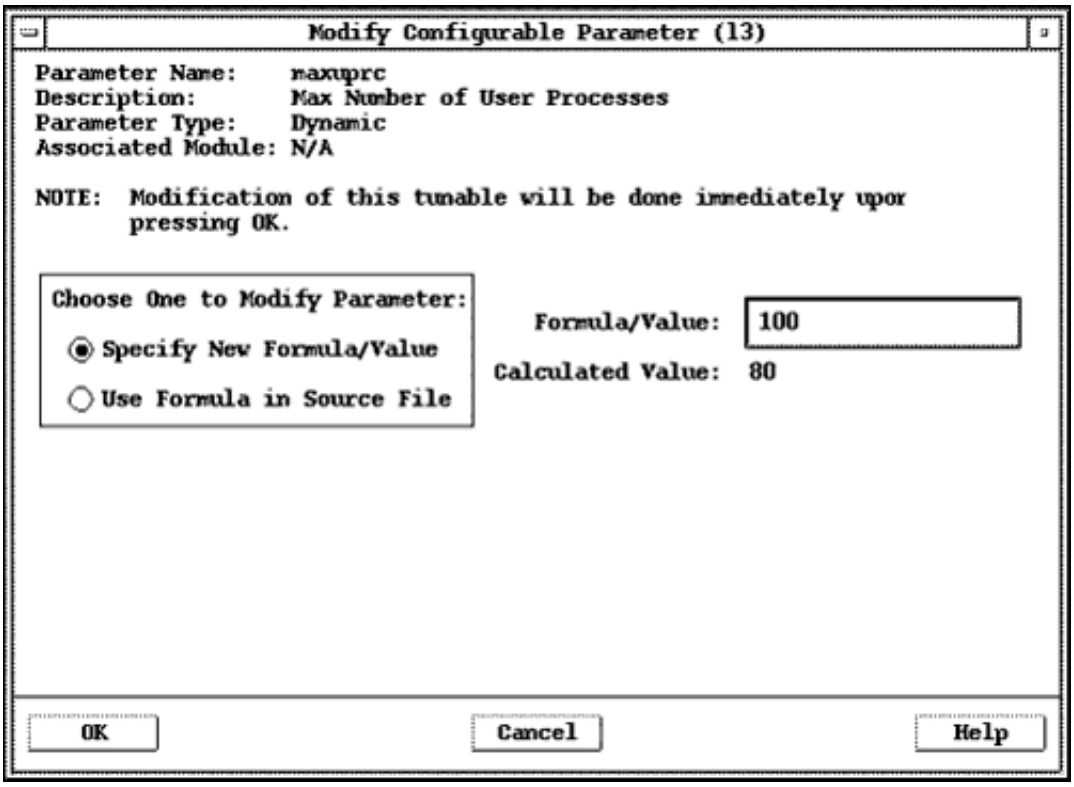
Figure 10-24 shows *maxuprc* selected in a SAM screen shot:

Figure 10-24. *maxuprc* Selected in SAM Window



Let's now increase the value of *maxuprc* from *80* to *100* as shown in Figure 10-25:

Figure 10-25. *maxuprc* from *80* to *100*



Clicking *OK* results in changing *maxuprc* from *80* to *100* on the system immediately, without a reboot required, as shown in Figure 10-26:

Figure 10-26. *maxuprc* Increase Takes Effect Immediately

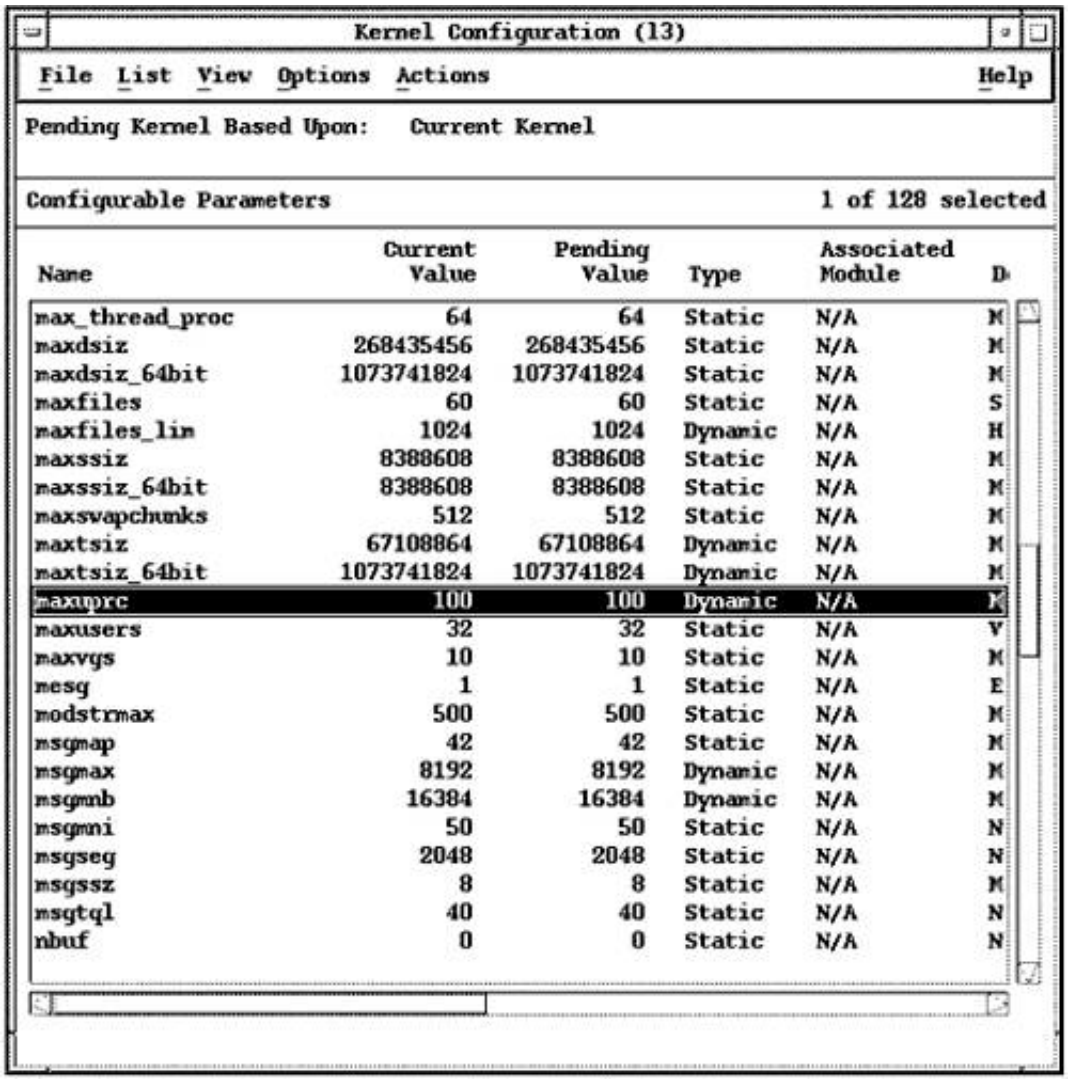


Figure 10-26 shows that both the *Current Value* and *Pending Value* of *maxuprc* are *100,* without a reboot required. Dynamic configurable parameters and dynamically loadable kernel modules can be updated in this fashion without a reboot required.

Drivers

When you select *Drivers*, the drivers for your current kernel, the template file on which your current kernel is based, and the pending kernel are listed. You'll know that the drivers displayed are for more than your current kernel because you'll see that some of the drivers listed are *Out* of both your current and pending kernels. The following information is listed for you when you enter the *Drivers* subarea:

- Name - Name of the driver.
- Current State - Lists whether the driver is *In* or *Out* of **/stand/vmunix**.
- Pending State - Lists whether the driver is *In* or *Out* of the pending kernel to be built.
- Class - Identifies element as a driver or module.
- Type - Identifies driver as part of the static kernel or a loadable module.

- Load Module at Boot? - Lists whether driver is automatically loaded at boot time if it is a loadable module.

- Description - A few words describing the driver.

The *Current State* indicates whether or not the driver selected is in **/stand/vmunix.**

The *Pending State* indicates whether or not you have selected this driver to be added to or removed from the kernel. *In* means that the driver is part of the kernel or is pending to be part of the kernel. *Out* means that the driver is not part of the kernel or is pending to be removed from the kernel.

Typically, drivers are added statically. In other words, they are added to the kernel and left there. However, if the driver is a specially created module for a particular purpose, then it may be configured as loadable. This means that it can be loaded and unloaded from the kernel without rebooting the system. This advanced feature is discussed in detail in the "Managing Dynamically Loadable Kernel Modules" chapter of the *Managing Systems and Workgroups* book from Hewlett-Packard.

Using the *Actions* menu, you can select one of the drivers and add or remove it. You can also pick *View Driver Details* from the *Actions* menu after you select one of the drivers. You can select *Process New Kernel* from the *Actions* menu. If you have indeed modified this screen by adding or removing drivers, you want to recreate the kernel. SAM asks whether you're sure that you want to rebuild the kernel before it does this for you. The only recommendation I can make here is to be sure that you have made your selections carefully before you rebuild the kernel.

Dump Devices

When you enter this subarea, both the *Current Dump Devices* and *Pending Dump Devices* are listed for you. A dump device is used when a serious kernel problem occurs with your system, and main memory is written to disk. This information is a core dump that can later be read from disk and used to help diagnose the kernel problem.

Prior to HP-UX 11.x, memory dumps contained the entire image of physical memory. As a result, in order to get a full memory dump, you needed to create a dump area at least as large as main memory. With systems with memory size into the gigabits, a large amount of disk space was wasted just waiting around for a system panic to occur. With HP-UX 11.x comes a new, fast dump feature. This allows you to pick and choose what to dump. The fast dump feature not only prevents such things as unused memory pages and user text pages from being dumped, but it also allows you to configure what memory page classes to dump.

The sizes of the dump areas can be configured somewhat smaller than main memory in your system. You can specify a disk or logical volume as a dump device (you can also specify a disk section, but I don't recommend that you use disk sections at all). The entire disk or logical volume is then reserved as a dump device.

If no dump device is specified or if the size of the dump area is less than what is configured to be dumped, then the core dump is written to primary swap. At the time of system boot, the core dump is written out to a core file, usually in **/var/adm/crash**. This is the way most systems I have worked on operate; that is, there is no specific dump device specified and core dumps are written to primary swap and then to **/var/adm/crash**. This approach has sometimes been a point of confusion; that is, primary swap may indeed be used as a dump device, but a dump device is used specifically for core dump purposes whereas primary swap fills this role in the event there is no dump device specified. As long as you don't mind the additional time it takes at boot to write the core dump in primary swap to a file, you may want to forgo adding a specific dump device to your system.

Since you probably won't be allocating an entire disk as a dump device, you may be using a logical volume. You must select a logical volume in the root volume group that is unused or is used for non-file-system swap. This is done by selecting *Add* from the *Actions* menu to add a disk or logical volume to the list of dump devices.

You will want to get acquainted with the *View Dump Configuration Details* subarea of *Dump Devices*. It is here where you can see what the current dump configuration is, what the current and pending kernel dump configurations are, and what the current runtime dump configuration is.

Subsystems

Selecting *Subsystems* lists all of your subsystems. For each subsystem, the following information is listed:

- *Name* - Name of the subsystem.
- *Current Value* - Lists whether the subsystem is *In* or *Out* of **/stand/vmunix**.
- *Pending Value* - Lists whether the subsystem is *In* or *Out* of the pending kernel.
- *Description* - A few words describing the parameter.

You can then take a number of *Actions*, including the following:

Templates You can select a kernel template to load that is basically a different kernel configuration from the one you are currently running.

Process New Kernel

After making whatever changes you like to the *Pending State* of a subsystem, you can have SAM create a new kernel for you.

View Subsystem Details

You get a little more information about the subsystem when you select this.

Add Subsystem to Kernel

When you highlight one of the subsystems and select this from the menu, the *Pending State* is changed to *In* and the subsystem will be added to the kernel when you rebuild the kernel.

Remove Subsystem from Kernel

When you highlight one of the subsystems and select this from the menu, the *Pending State* is changed to *Out* and the subsystem will be removed from the kernel when you rebuild the kernel.

After making selections, you can rebuild the kernel to include your pending changes or back out of them without making the changes.

[Team LiB]

Networking and Communications

The menu hierarchy for *Networking and Communications* is shown in Figures 10-27 through 10-29. This area contains many advanced networking features. Because there are so many networking areas to cover in SAM, I'll go over just a few so you can get a feel for working in this area. The bubble diagram shows the many areas related to networking configuration for which you can use SAM so you can refer back to it if you have a question about whether or not some specific networking can be configured using SAM.

Figure 10-27. *Networking and Communications* Menu Structure

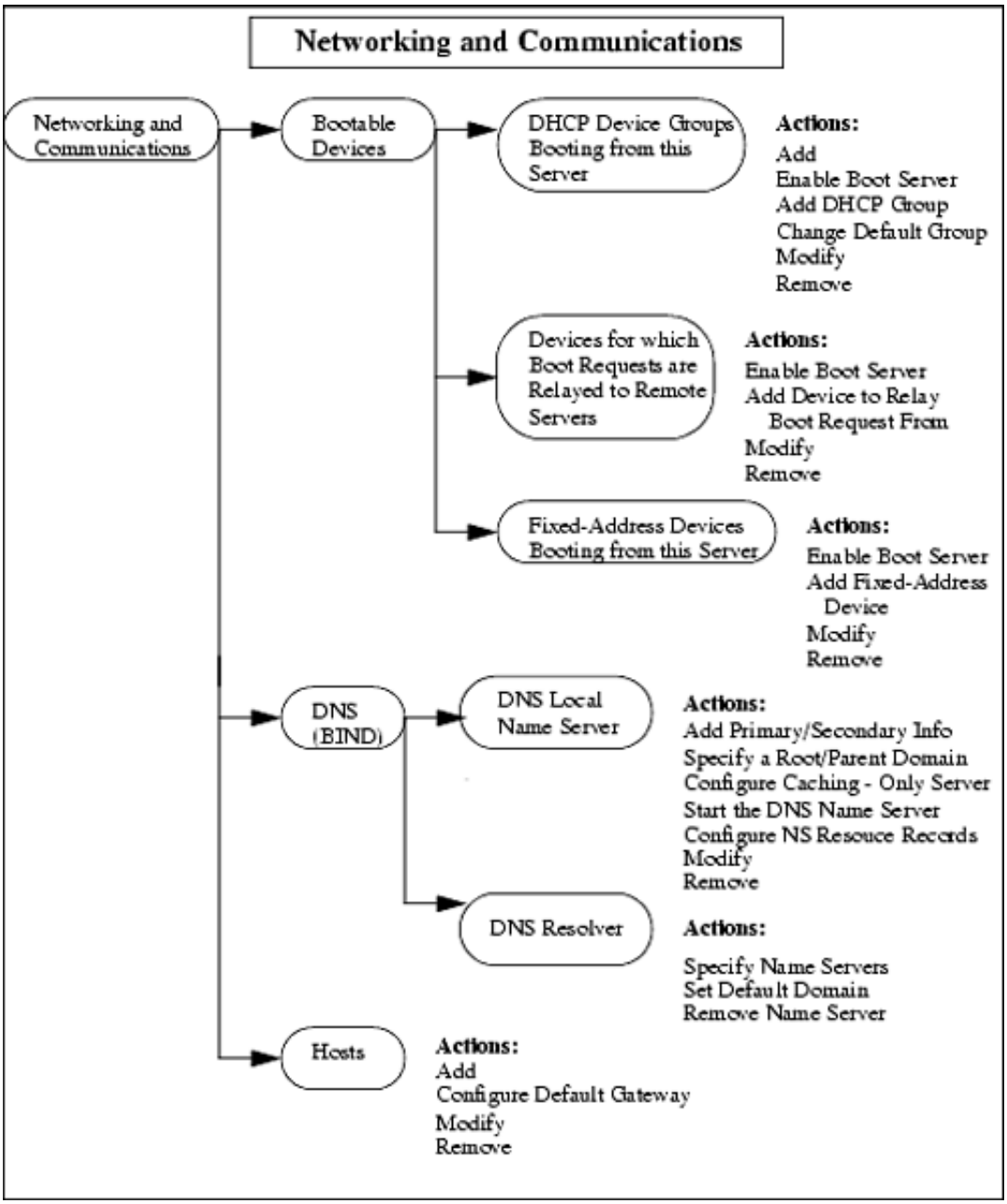


Figure 10-29. *Networking and Communications* Menu Structure (cont)

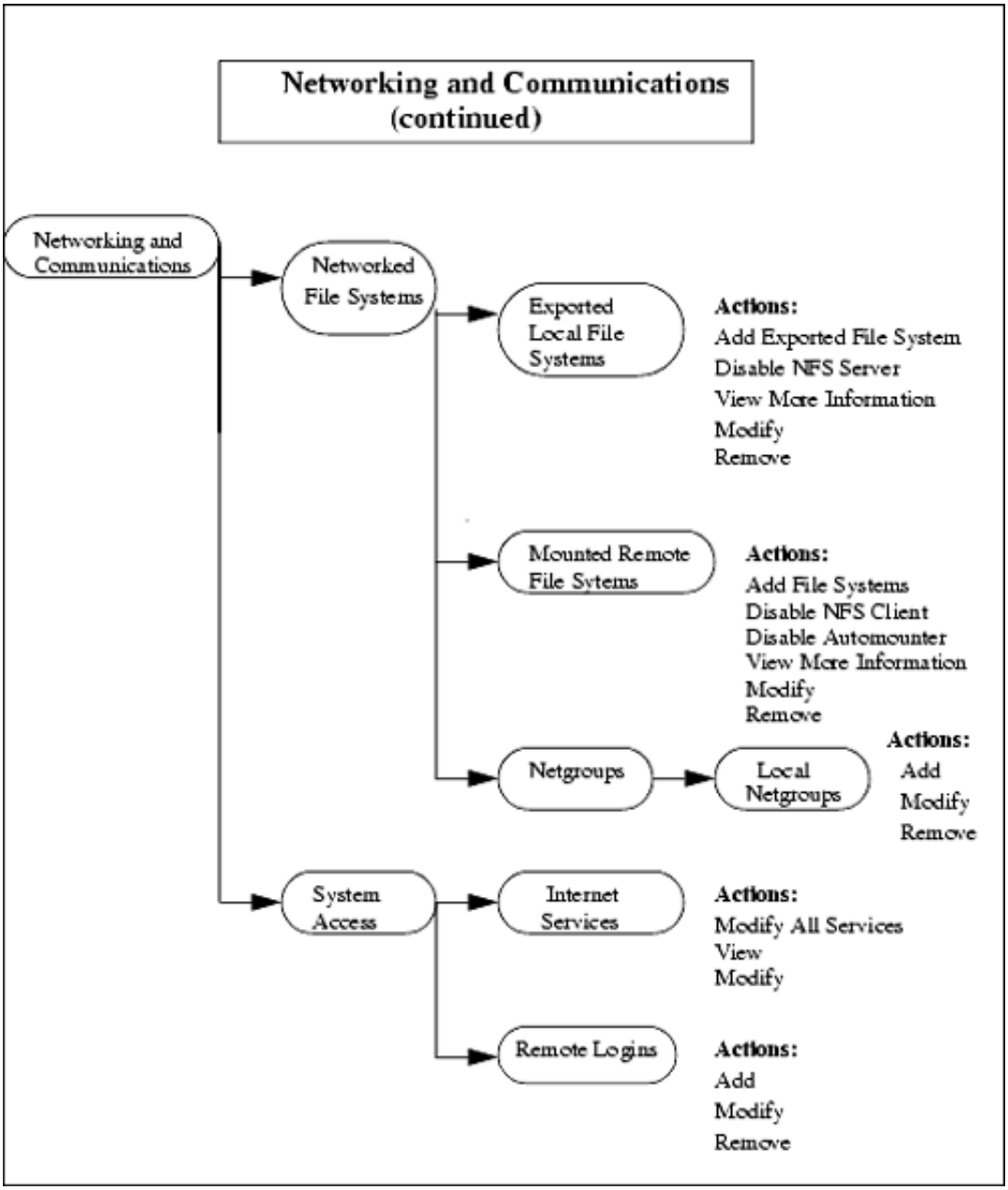
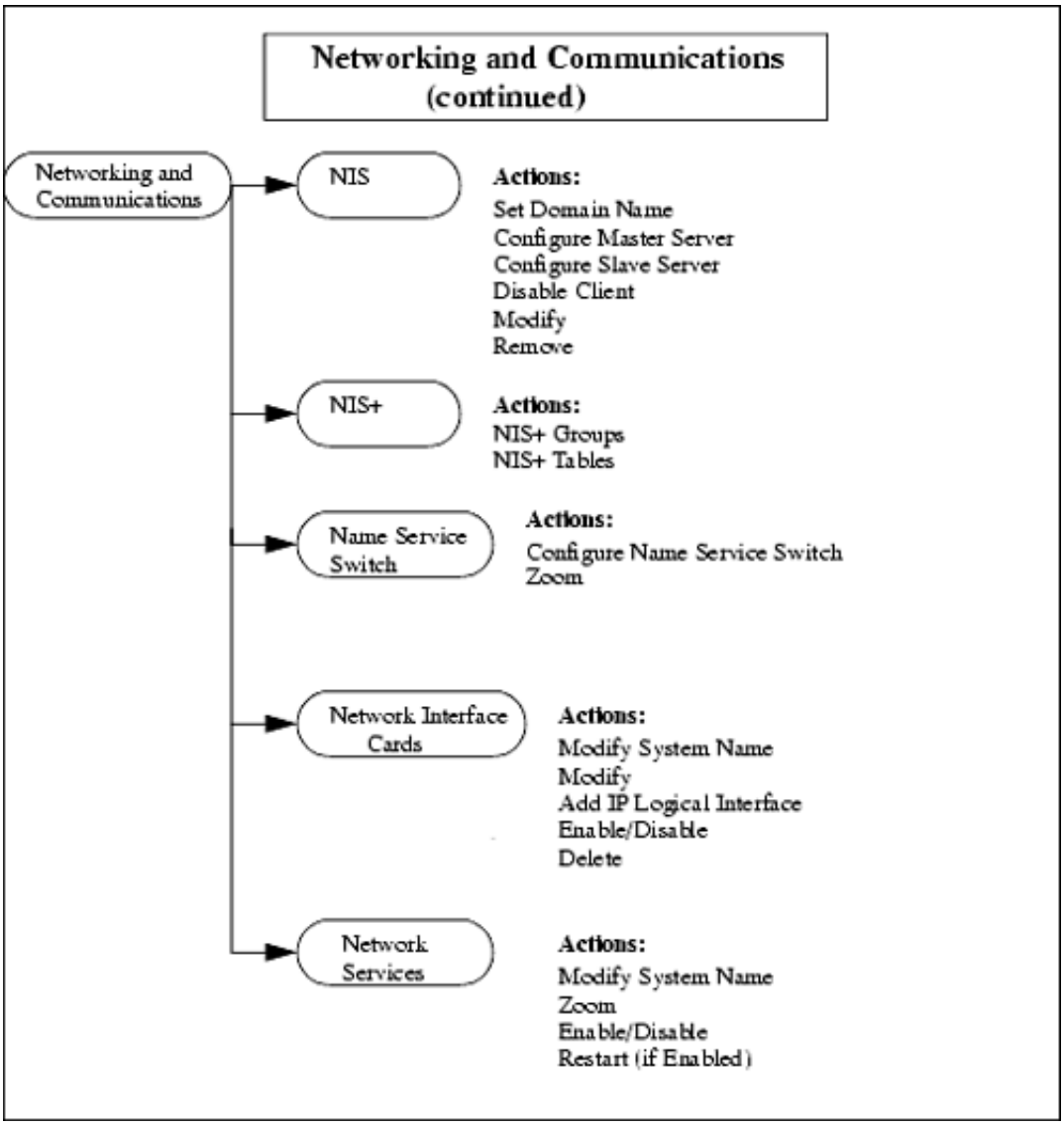


Figure 10-28. *Networking and Communications* Menu Structure (cont)



Bootable Devices

The *Bootable Devices* area is further subdivided into three subareas: *DHCP Device Groups Booting from this Server, Devices for which Boot Requests are Relayed to Remote Servers*, and *Fixed-Address Devices Booting from this Server*. I will briefly describe each subarea and its use. DHCP (Dynamic Host Configuration Protocol) is now available on HP-UX and is used by many services.

The *DHCP Device Groups Booting from this Server* subarea is where the device groups can be configured. Each group would contain a set of IP addresses for use by that device group. Devices could be such things as specific types of printers or specific types of terminals.

In the *Devices for which Boot Requests are Relayed to Remote Servers* subarea, you can view information about Bootstrap Protocol (Bootp) client devices that get their booting information from remote Bootp or DHCP servers. Information is displayed on the client or client groups, including the IP addresses of the remove servers and the maximum number of hops a boot request from a client or client group can be relayed.

In the *Fixed-Address Devices Booting from this Server* subarea, you can specify systems that will boot from your system using Bootstrap Protocol (Bootp) or DHCP. Bootp is a means by which a system can discover network information and boot automatically. The Bootp software must be loaded on your system in order for other devices to use it as a boot source (see the **swlist** command in Chapter 2 regarding how to list software installed on your system). In this subarea, you can add, modify, or remove a Bootp device. In

addition, you can enable or disable the Bootp Server. Similarly, DHCP allows the client to use one of a pool of IP addresses in order to boot automatically. Applications such as Ignite-UX can be configured to use this protocol.

When you enter the *Fixed-Address Devices Booting from this Server* subarea, you immediately receive a list of devices that can boot off your system. You can choose *Add* from the *Actions* menu and you'll be asked to enter the following information about the device you are adding:

- Host Name
- Internet Address
- Subnet Mask (this is optional)
- Station Address in hex or client ID (this is optional)
- Boot File Name
- Whether you'll be using Ethernet or IEEE 802.3 for booting
- Whether to send the hostname to the client or device

You can select *Enable Protocol Server* or *Disable Protocol Server* from the *Actions* menu, depending on whether your system is currently disabled or enabled to support this functionality. When you *Enable Protocol Server,* you also enable Trivial File Transfer Protocol (TFTP), which boot devices use to get boot files. When you enable or disable this, the **/etc/inetd.conf** is edited. This file contains configuration information about the networking services running on your system. If a line in **/etc/inetd.conf** is preceded by a "#", then it is viewed as a comment. The daemon that reads the entries in this file is **/usr/sbin/inetd**. Before enabling or disabling Bootp, you may want to view the **/etc/inetd.conf** file and see what services are enabled. After you make your change through SAM, you can again view **/etc/inetd.conf** to see what has been modified. See *System Access* for security related to **/etc/inetd.conf.** The following is the *beginning* of the **/etc/inetd.conf** file from a system showing Bootp and TFTP enabled. Also, a brief explanation of the fields in this file appears at the beginning of the file:

```
## Configured using SAM by root
##
#
# Inetd  reads its configuration information from this file upon ex-
# ecution and at some later time if it is reconfigured.
#
# A line in the configuration file has the following fields separated
# by tabs and/or spaces:
#
#   service name          as in /etc/services
#   socket type           either "stream" or "dgram"
#   protocol              as in /etc/protocols
#   wait/nowait            only applies to datagram sockets, stream
#                         sockets should specify nowait
#    user                  name of user as whom the server should run
#   server program        absolute pathname for the server inetd
#                         will execute
#  server program args.   arguments server program uses as they
#                          normally are starting with argv[0] which
#                         is the name of the server.
#
# See the inetd.conf(4) manual page for more information.
##

##
#
#              ARPA/Berkeley services
#
##
ftp         stream tcp nowait root /usr/lbin/ftpd      ftpd -l
telnet      stream tcp nowait root /usr/lbin/telnetd   telnetd

# Before uncommenting the "tftp" entry below, please make sure
# that you have a "tftp" user in /etc/passwd. If you don't
```

```
# have one, please consult the tftpd(1M) manual entry for
# information about setting up this service.

tftp       dgram  udp wait   root  /usr/lbin/tftpd    tftpd
bootps     dgram  udp wait   root  /usr/lbin/bootpd   bootpd
#finger    stream tcp nowait bin   /usr/lbing/fingerd fingerd
login      stream tcp nowait bin   /usr/lbin/rlogind  rlogind
shell      stream tcp nowait bin   /usr/lbin/remshd   remshd
exec       stream tcp nowait root  /usr/lbin/rexecd   rexecd
#uucp      stream tcp nowait bin   /usr/sbin/uucpd    uucpd

                      -
                      -
                      -
```

If you select *Fixed-Address Device Client Names*, you can then select *Modify* or *Remove* from the *Actions* menu and either change one of the parameters related to the client, such as its address or subnet mask, or completely remove the client.

DNS (BIND)

Domain Name Service (DNS) is a name server used to resolve hostname-to-IP addressing. HP-UX uses BIND, Berkeley InterNetworking Domain, one of the name services that can be used to implement DNS. A DNS server is responsible for the resolution of all host names on a network or subnet. Each DNS client would rely on the server to resolve all IP address-to-host name issues on the client's behalf. A boot file is used by the server to locate database files. The database files map host names to IP addresses and IP addresses to host names. Through SAM, a DNS server can be easily set up.

Information about DNS and its setup and administration is provided in the HP-UX manual *Installing and Administering Internet Services*.

Hosts

This subarea is for maintaining the default gateway and remote hosts on your system. When you enter this subarea, you receive a list of hosts specified on your system. This information is retrieved from the **/etc/hosts** file on your system.

You can then *Add* a new host, *Specify Default Gateway, Modify* one of the hosts, or *Remove* one of the hosts, all from the *Actions* menu. When adding a host, you'll be asked for information pertaining to the host, including its Internet Address, system name, aliases for the system, and comments.

NIS

Network Information Service (NIS) is a database system used to propagate common configuration files across a network of systems. Managed on a master server are such files as **/etc/passwd, /etc/hosts**, and **/etc/auto***, files used by automounter. Formerly called "yellow pages," NIS converts these files to its own database files, called *maps*, for use by clients in the NIS domain. When a client requests information, such as when a user logs in and enters their password, the information is retrieved from the server rather than from the client's system. Thus, this information only needs to be maintained only on the server.

Through SAM, the NIS master server, slave servers, and clients can be configured, enabled, disabled, and removed. Once the master, slaves, and clients are established, you can easily build, modify, and push the various maps to the slaves.

NIS is not available on trusted systems.

NIS+

HP-UX 11i supports NIS+. This is not an enhancement of NIS, but rather a new service that includes standard and trusted systems and non-HP-UX systems. If you already use NIS, a compatibility mode version of NIS+ allows servers to answer requests from both NIS and NIS+ clients. When NIS+ is configured on a trusted system, in the *Auditing and Security* area of SAM, a new subarea, *Audited NIS+*

Users, is displayed.

Name Service Switch

The Name Service Switch file, **/etc/nsswitch.conf**, can now be configured through SAM. This service allows you to prioritize which name service (FILES, NIS, NIS+, DNS, or COMPAT) to use to look up information. Unless you specifically use one of these services, the default of FILES should be used. The FILES designation supports the use of the local **/etc** directory for such administrative files as **/etc/passwd, /etc/hosts**, and **/etc/services**. (COMPAT is used with the compatibility mode of NIS+.)

More information about Name Service Switch file and its setup is described in the HP-UX manual *Installing and Administering NFS Services*.

Network Interface Cards

This subarea is used for configuring any networking cards in your system. You can *Enable, Disable*, and *Modify* networking cards as well as *Modify System Name,* all from the *Actions* menu. Under *Add IP Logical Interface,* you can add additional logical IP addresses to an existing network card.

The *Network Interface Cards* screen lists the network cards installed on your system, including the information listed below. You may have to expand the window or scroll over to see all this information.

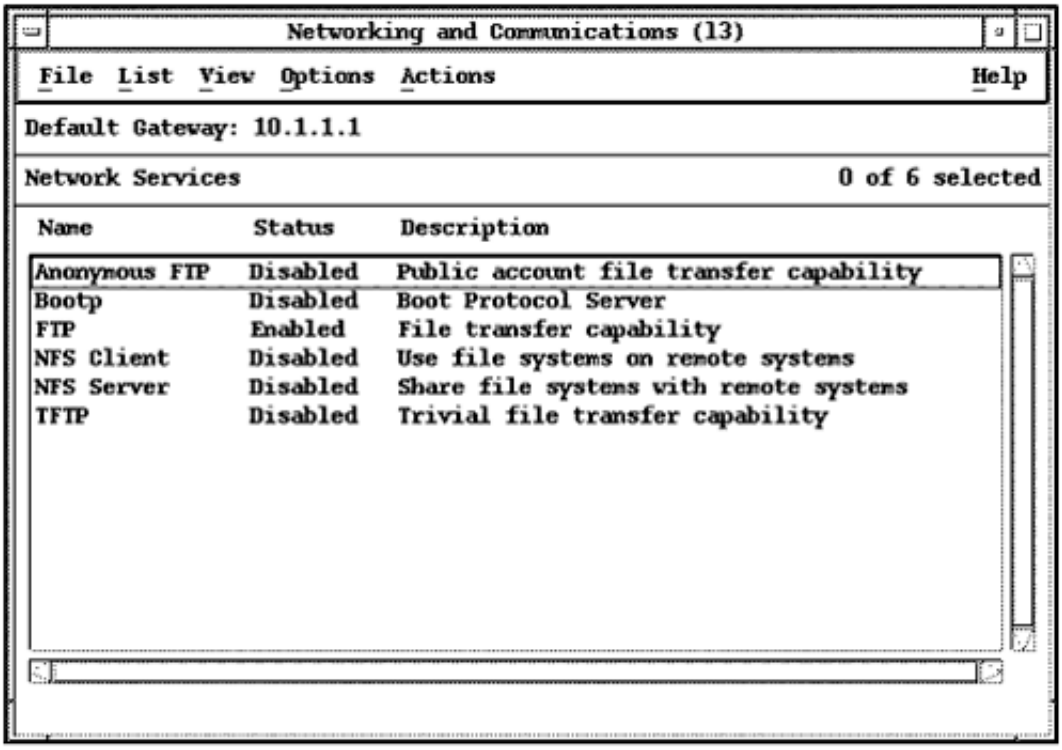
- Card Type such as Ethernet, IEEE 802.3, Token Ring, FDDI, etc.
- Card Name
- Hardware Path
- Status, such as whether or not the card is enabled
- Internet Address
- Subnet Mask
- Station Address in hex

Included under *Configure* for Ethernet cards is *Advanced Options,* which will modify the Maximum Transfer Unit (MTU) for this card. Other cards included in your system can also be configured here, such as ISDN, X.25, ATM, and so on.

Network Services

This subarea is used to enable or disable *some* of the network services on your system. You will recognize some of the network services in Figure 10-30 from the **/etc/inetd.conf** file shown earlier. This screen has three columns, which are the Name, Status, and Description of the network services. Figure 10-30 from the *Network Services* subarea shows some of the network services that can be managed:

Figure 10-30. *Network Services* Window



After selecting one of the network services shown, you can *Enable* or *Disable* the service, depending on its current status, *Restart* the service if it is currently enabled, get more information about the service with *Zoom*, or *Modify System Name,* all from the *Actions* menu.

Network File Systems

This subarea is broken down into *Exported Local File Systems, Mounted Remote File Systems,* and *Netgroups*. NFS is broken down into these first two areas because you can export a local file system without mounting a remote file system, and vice versa. This means that you can manage these independently of one another. You may have an NFS server in your environment that won't mount remote file systems, and you may have an NFS client that will mount only remote file systems and never export its local file system. *Entropies*, a part of NIS, allows you to group a set of systems or users to be used together. Among other things, netgroup designations can be used to export file systems to.

Under *Exported Local File Systems,* you can select the file systems you want to export. The first time you enter this screen you have no exported file systems listed. When you select *Add Exported File System* from the *Actions* menu, you enter such information as:

- Local directory name
- User ID
- Whether or not to allow asynchronous writes
- Permissions

After this exported file system has been added, you can select it and choose from a number of *Actions,* including *Modify* and *Remove*.

Under *Mounted Remote File Systems,* you have listed for you all of the directories and files that are mounted using NFS. These can be either mounted or unmounted on demand with the automounter. After selecting one of the mounted file systems, you can perform various *Actions*. For every remote file system mounted, you have the following columns:

- *Mount Directory*, which displays the name of the local directory name used to mount the remote directory.
- *Type*, which is either *NFS* for standard NFS or *Auto* for automounter (see the paragraph below).
- *Remote Server*, which displays the name of the remote system where the file or directory is mounted.
- *Remote Directory*, which is the name of the directory under which the directory is remotely mounted.

You should think about whether or not you want to use the NFS automounter. With automounter, you mount a remote file or directory on demand, that is, when you need it. Using a master map, you can specify which files and directories will be mounted when needed. The files and directories are not continuously mounted with automounter, resulting in more efficiency as far as how system resources are being used. There is, however, some overhead time associated with mounting a file or directory on-demand, as opposed to having it continuously mounted. From a user standpoint, this may be slightly more undesirable, but from an administration standpoint, using the automounter offers advantages. Since the automounter is managed through SAM, there is very little additional work you need to perform to enable it.

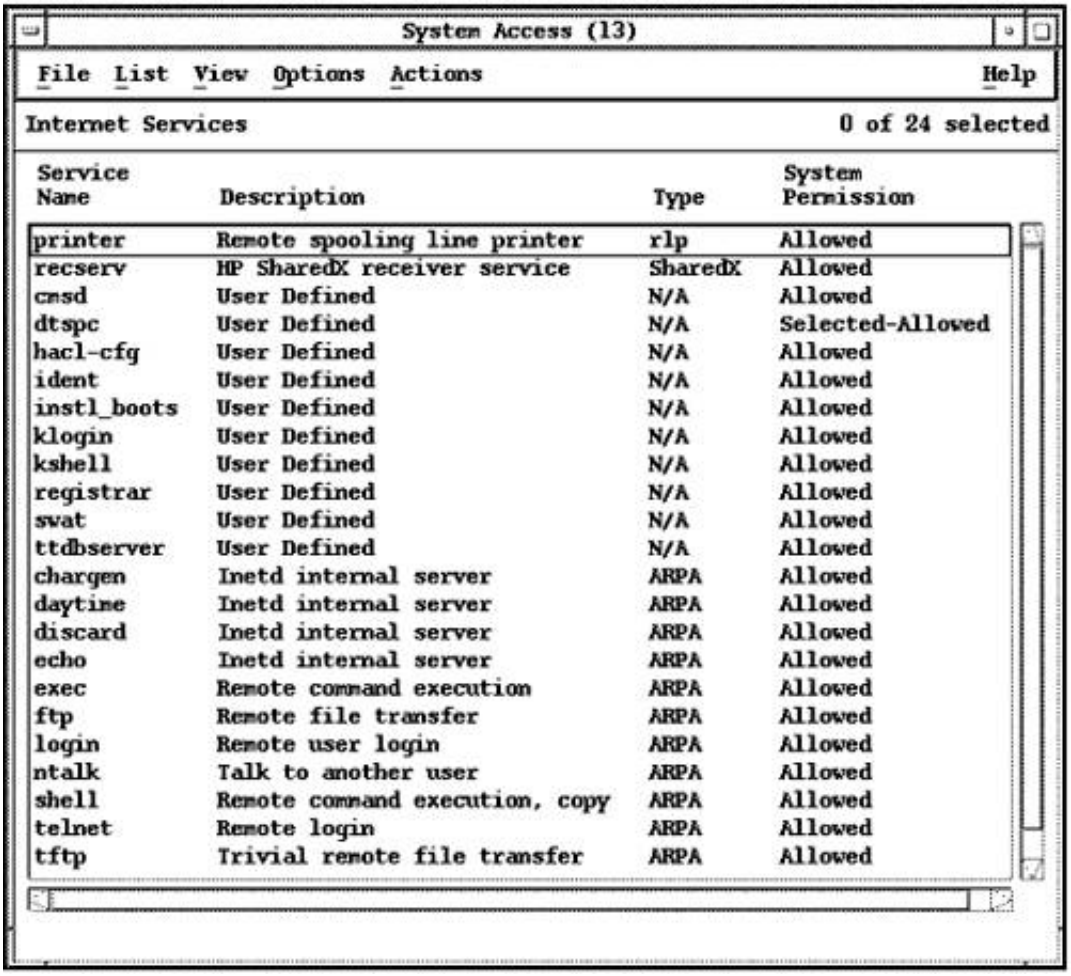
System Access

This subarea is broken down into *Internet Services* and *Remote Logins*.

When you select *Internet Services,* the screen lists the networking services that are started by the Internet daemon **/usr/sbin/inetd**. I earlier covered **/etc/inetd.conf**, which is a configuration file that lists all of the network services supported by a system that is read by **inetd**. There is also a security file, **/var/adm/inetd.sec**, which serves as a security check for **inetd**. Although many other components are involved, you can view **inetd, /etc/inetd.conf**, and **/var/adm/inetd.sec** as working together to determine what network services are supported and the security level of each.

Listed for you in the *System Access* subarea are *Service Name, Description, Type*, and *System Permission*. Figure 10-31 shows the defaults for my system:

Figure 10-31. *System Access* - Internet Services Window



You could change the permission for any of these entries by selecting them, using the *Modify* command from the *Actions* menu, and selecting the desired permissions.

Remote Logins is used to manage security restrictions for remote users who will access the local system. Two HP-UX files are used to manage users. The file **/etc/hosts.equiv** handles users, and **/.rhosts** handles superusers (root). When you enter this subarea, you get a list of users and the restrictions on each. You can then *Add, Remove*, or *Modify* login security.

[Team LiB]

◀ PREVIOUS NEXT ▶

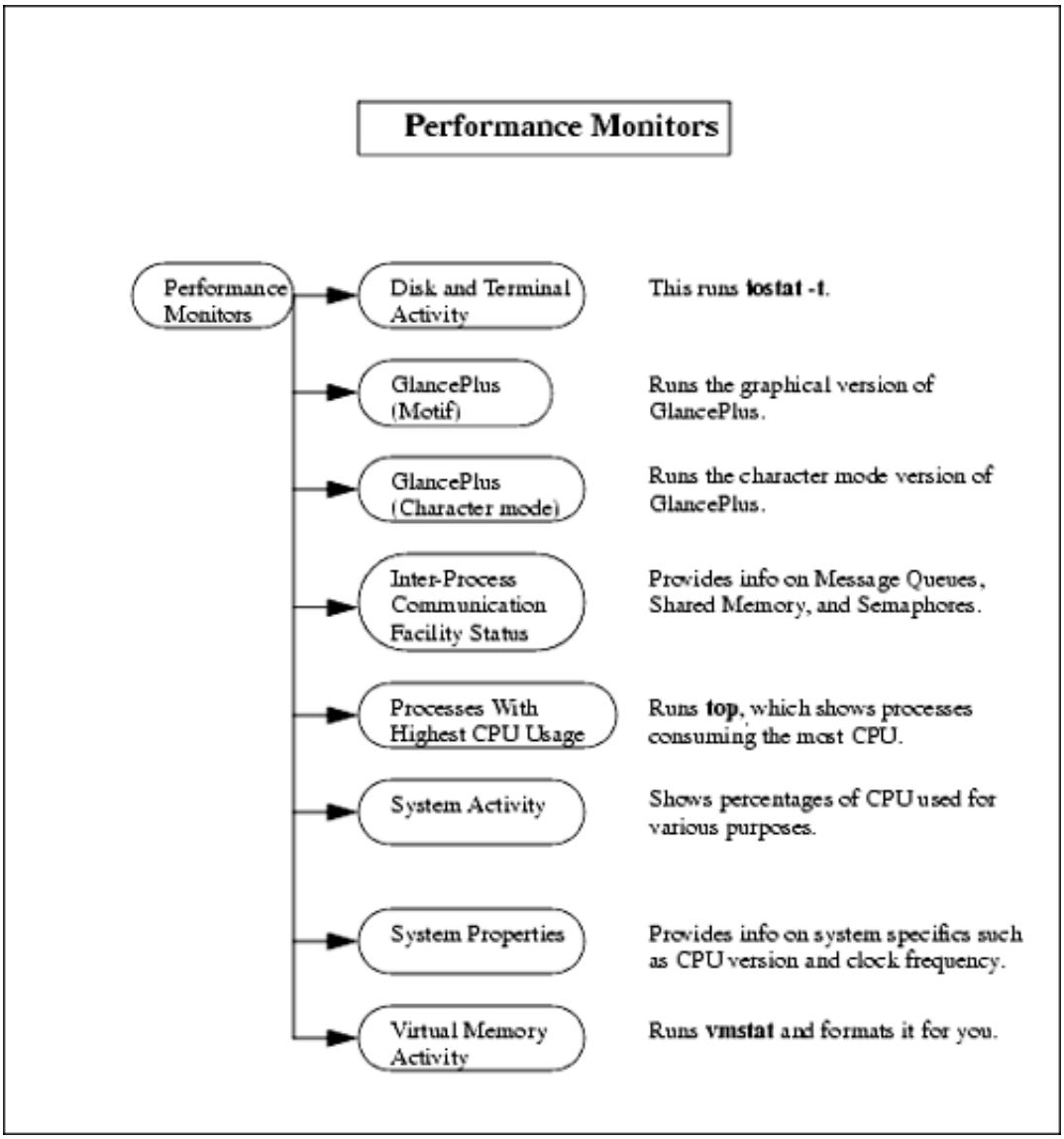
Partition Manager

parmgr is used to manage nPartitions (hard partitions) and can be invoked through SAM or at the command line. **parmgr** is covered in Chapter 17 along with many other techniques for managing nPartitions.

Performance Monitors

Under *Performance Monitors,* you can view the performance of your system in several different areas such as disk and virtual memory. Figure 10-32 shows the menu hierarchy of *Performance Monitors*:

Figure 10-32. *Performance Monitors* Menu Structure



Performance Monitors provides you with a window into several areas of your system. If you are serious about becoming familiar with the tools available on your system to help you understand how your system resources are being used, you should take a close look at Chapter 11. Chapter 11 is devoted to getting a handle on how your system resources are being used, including many built-in HP-UX commands. Some of the performance monitors you can select in this subarea are HP-UX commands, which you'll need some background in before you can use them. I'll cover these areas only briefly because this material is covered in more detail in Chapter 11.

Disk and Terminal Activity



Selecting *Disk and Terminal Activity* opens a window that shows the output of **iostat -t**. I have included the description of **iostat** from Chapter 11 to save you the trouble of flipping ahead. When the *Disks and Terminal Activity* window with the output of **iostat** is opened for you, it shows a single **iostat** output. When you press *Return,* the window is automatically closed for you.

The **iostat** command gives you an indication of the level of effort the CPU is putting into I/O and the amount of I/O taking place among your disks and terminals. The following example shows the **iostat -t** command, which will be executed every three seconds, and associated output from an HP-UX 11.x system:

```
    # iostat -t 3
                 tty             cpu
               tin   tout      us    ni    sy    id
                78     42       2     0    28    70

/dev/dsk/c0t1d0           /dev/dsk/c0t4d0          /dev/dsk/c0t6d0
bps  sps   msps           bps   sps  msps         bps  sps  msps
  0    0     0            33   8.3  25.2           7    1  19.5

                 tty             cpu
               tin   tout      us    ni    sy    id
                66     24       0     0    30    70

/dev/dsk/c0t1d0           /dev/dsk/c0t4d0          /dev/dsk/c0t6d0
bps  sps   msps           bps   sps  msps         bps  sps  msps
  5   12   15.9           36   9.7    21           7  1.2  13.8

                 tty             cpu
               tin   tout      us    ni    sy    id
                90     29       1     0    25    73
/dev/dsk/c0t1d0           /dev/dsk/c0t4d0          /dev/dsk/c0t6d0
bps  sps   msps           bps  sps   msps         bps  sps msps
 12  1.7   15.5           24    3   19.1          14  2.1 14.6

                 tty             cpu
               tin   tout      us    ni    sy    id
                48     16       1     0    16    83

/dev/dsk/c0t1d0           /dev/dsk/c0t4d0          /dev/dsk/c0t6d0
bps  sps   msps           bps  sps   msps         bps  sps  msps
  0    0     0            62  9.3    18           12    2  17.2

                 tty             cpu
               tin   tout      us    ni    sy    id
                32     48       7     0    14    79

/dev/dsk/c0t1d0           /dev/dsk/c0t4d0          /dev/dsk/c0t6d0
bps  sps msps             bps   sps  msps         bps sps  msps
  1  0.3   14.4           5    .9  16.2          171 29.4 18.2

                 tty             cpu
               tin   tout      us    ni    sy    id
                 2     40      20     1    42    27

/dev/dsk/c0t1d0           /dev/dsk/c0t4d0          /dev/dsk/c0t6d0
bps   sps msps            bps  sps  msps         bps  sps  msps
248 30.9  20.8            203 29.2  18.8          165 30.6  22.1
```

Descriptions of the reports you receive with **iostat** for terminals, the CPU, and mounted file systems follow.



For every terminal you have connected (tty), you see a "tin" and "tout," which represent the number of characters read from your terminal and the number of characters written to your terminal, respectively. The *-t* option produces this terminal report.

For your CPU, you see the percentage of time spent in user mode ("us"), the percentage of time spent running user processes at a low priority called nice ("ni"), the percentage of time spent in system mode ("sy"), and the percentage of time the CPU is idle ("id").

For every locally mounted file system, you receive information on the kilobytes transferred per second ("bps"), number of seeks per second ("sps"), and number of milliseconds per average seek ("msps"). For disks that are NFS-mounted or disks on client nodes of your server, you will not receive a report; **iostat** reports only on locally mounted file systems.

GlancePlus

GlancePlus is available here if it is installed on your system. You are given the choice of using either the Motif (graphical) version or the character mode version of GlancePlus. GlancePlus and other HP VantagePoint products are covered in Chapter 11.

Inter-Process Communication Facility Status

Inter-Process Communication Facility Status shows categories of information related to communication between processes. You receive status on Message Queues, Shared Memory, and Semaphores. This is a status window only, so again, when you press *Return*, and the window closes.

Processes with Highest CPU Usage

Processes with Highest CPU Usage is a useful window that lists the processes consuming the most CPU on your system. Such useful information as the *Process ID*, its *Resident Set Size*, and the *Percentage of CPU* it is consuming are listed.

System Activity

System Activity provides a report of CPU utilization. You receive the following list:

%usr	Percent of CPU spent in user mode.
%sys	Percent of CPU spent in system mode.
%wio	Percent of CPU idle with some processes waiting for I/O, such as virtual memory pages moving in or moving out.
%idle	Percent of CPU completely idle.

System Properties

System Properties gives you a great overview of system specifics. Included here are those hard-to-find items such as processor information, CPU version, clock frequency, kernel support (32-bit or 64-bit), memory information, operating system version, and network IP and MAC addresses.

Virtual Memory Activity



Virtual Memory Activity runs the **vmstat** command. This too is covered in Chapter 11, but I have included the **vmstat** description here so that you don't have to flip ahead. Some of the columns of **vmstat** are moved around a little when the *Virtual Memory Activity* window is opened for you.

vmstat provides virtual memory statistics. It provides information on the status of processes, virtual memory, paging activity, faults, and a breakdown of the percentage of CPU time. In the following example, the output was produced ten times at five-second intervals. The first argument to the **vmstat** command is the interval; the second is the number of times you would like output produced.

vmstat 5 10:

procs			memory		page						faults			cpu		
r	b	w	avm	free	re	at	pi	po	fr	de	sr	in	sy	cs	us	sy	id
4	0	0	1161	2282	6	22	48	0	0	0	0	429	289	65	44	18	38
9	0	0	1161	1422	4	30	59	0	0	0	0	654	264	181	18	20	62
6	0	0	1409	1247	2	19	37	0	0	0	0	505	316	130	47	10	43
1	0	0	1409	1119	1	10	19	0	0	0	0	508	254	180	69	15	16
2	0	0	1878	786	0	1	6	0	0	0	0	729	294	217	75	17	8
2	0	0	1878	725	0	0	3	0	0	0	0	561	688	435	67	32	1
2	0	0	2166	98	0	0	20	0	0	0	66	728	952	145	8	14	78
1	0	0	2310	90	0	0	20	0	0	0	171	809	571	159	16	21	63
1	0	0	2310	190	0	0	8	1	3	0	335	704	499	176	66	14	20
1	0	0	2316	311	0	0	3	1	5	0	376	607	945	222	4	11	85

You will get more out of the **vmstat** command than you want. Here is a brief description of the categories of information produced by **vmstat**:

Processes are classified into one of three categories: runnable ("r"), blocked on I/O or short-term resources ("b"), or swapped ("w").

Next you will see information about memory. "avm" is the number of virtual memory pages owned by processes that have run within the last 20 seconds. If this number is roughly the size of physical memory minus your kernel, then you are near paging. The "free" column indicates the number of pages on the system's free list. It doesn't mean that the process has finished running and these pages won't be accessed again; it just means that they have not been accessed recently.

Next is paging activity. The first field (*re*) shows the pages that were reclaimed. These pages made it to the free list but were later referenced and had to be salvaged. Check to see that "re" is a low number. If you are reclaiming pages that were thought to be free by the system, then you are wasting valuable time salvaging these. Reclaiming pages is also a symptom that you are short on memory.

Next you see the number of faults in three categories: interrupts per second, which usually come from hardware ("in"); system calls per second ("sy"); and context switches per second ("cs").

The final output is CPU usage percentage for user ("us"), system ("sy"), and idle ("id"). This is not as complete as the **iostat** output, which also shows **nice** entries.

Peripheral Devices

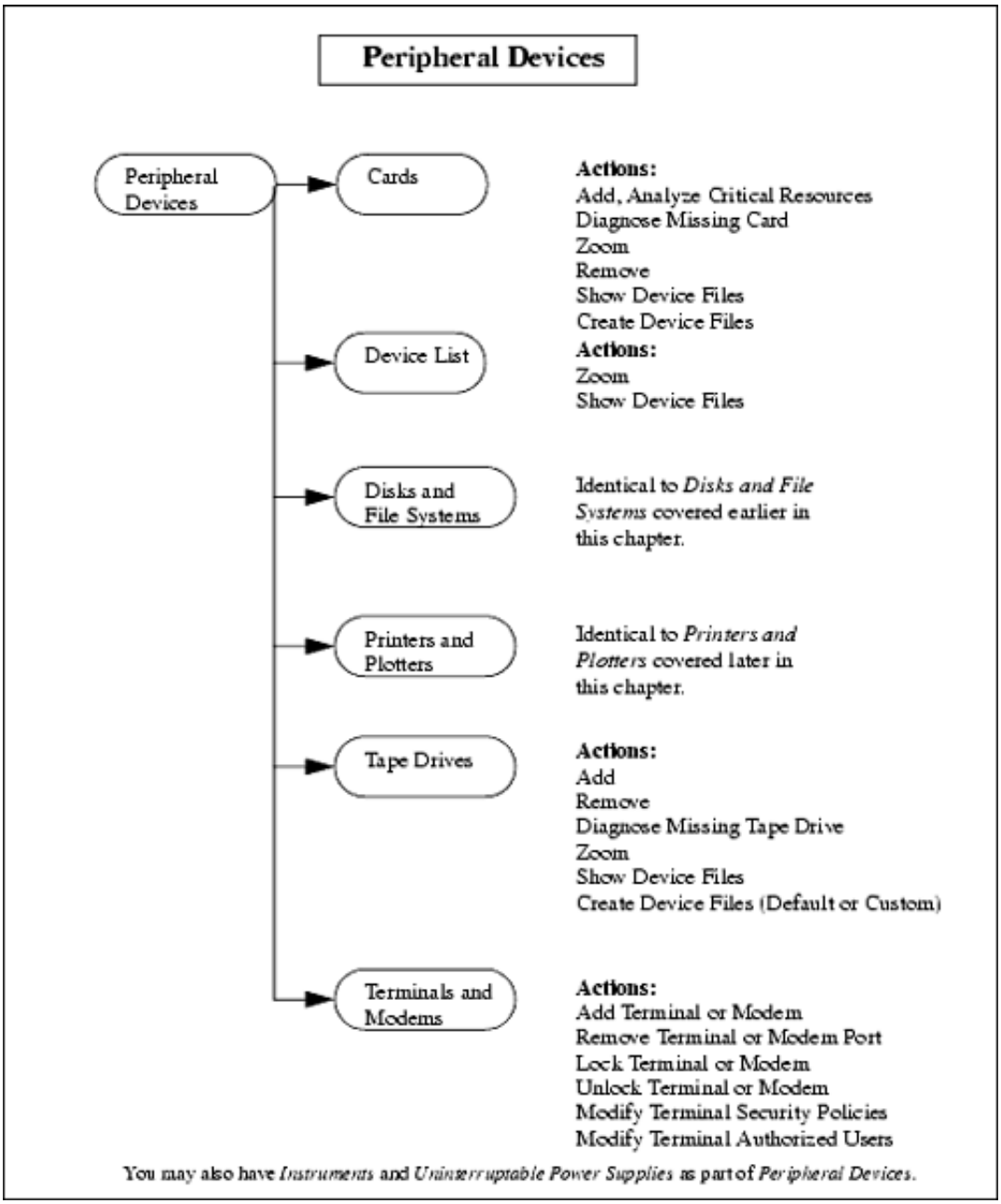
With *Peripheral Devices,* you can view any I/O cards installed in your system and peripherals connected to your system. These include both used and unused devices. You can also quickly configure any peripheral, including printers, plotters, tape drives, terminals, modems, and disks. This is a particularly useful area in SAM, because configuring peripherals in HP-UX is tricky. You perform one procedure to connect a printer, a different procedure to connect a disk, and so on, when you use the command line. In SAM, these procedures are menu-driven and therefore much easier.

Two of the six subareas, *Disks and File Systems* and *Printers and Plotters,* have their own dedicated hierarchy within SAM and are covered in this chapter. I don't cover these again in this section. The other four subareas, Cards, Device List, Tape Drives, and Terminals and Modems, are covered in this section.

It's impossible for me to cover every possible device that can be viewed and configured in SAM. What I'll do is give you examples of what you would see reported as devices on a server so that you can get a feel for what you can do under *Peripheral Devices* with SAM. From what I show here, you should be comfortable that SAM can help you configure peripherals.

Figure 10-33 shows the hierarchy of *Peripheral Devices*:

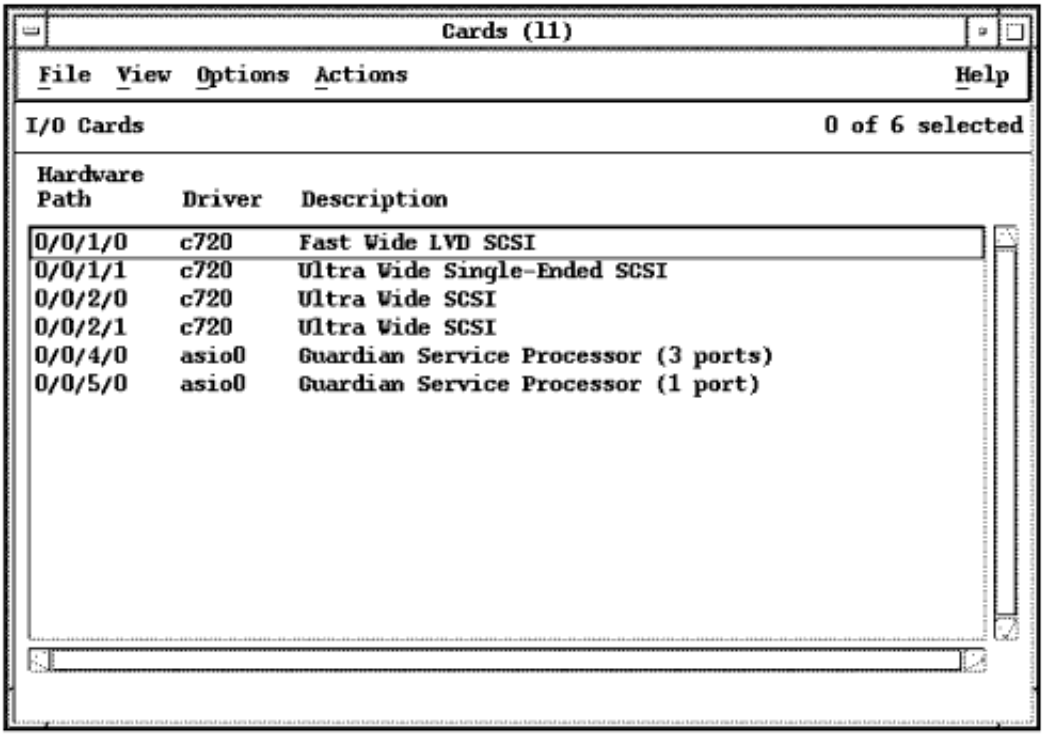
Figure 10-33. *Peripheral Devices* Menu Structure



Cards

When you select *Cards,* you are provided with a list of I/O cards in your system. You can also perform such tasks as adding and removing cards. Having this list of I/O cards is useful. Figure 10-34 shows a listing of I/O cards for an L-Class system:

Figure 10-34. *I/O Cards* Window for L-Class System



In *Cards,* you can perform the following *Actions*:

Add You can add a new I/O card in the window that is opened for you.

Diagnose Missing Card

If a card you have installed is not included in the list, you can select this action to determine the reason.

Zoom If you highlight a card and select *Zoom,* you will be provided such information as the hardware path, driver, and description of the card.

Remove If you highlight a card and select *Remove,* a window will appear that walks you through removing the card from the system.

Show Device Files

If you select this, a window will be opened in which the device files associated with the card will be listed.

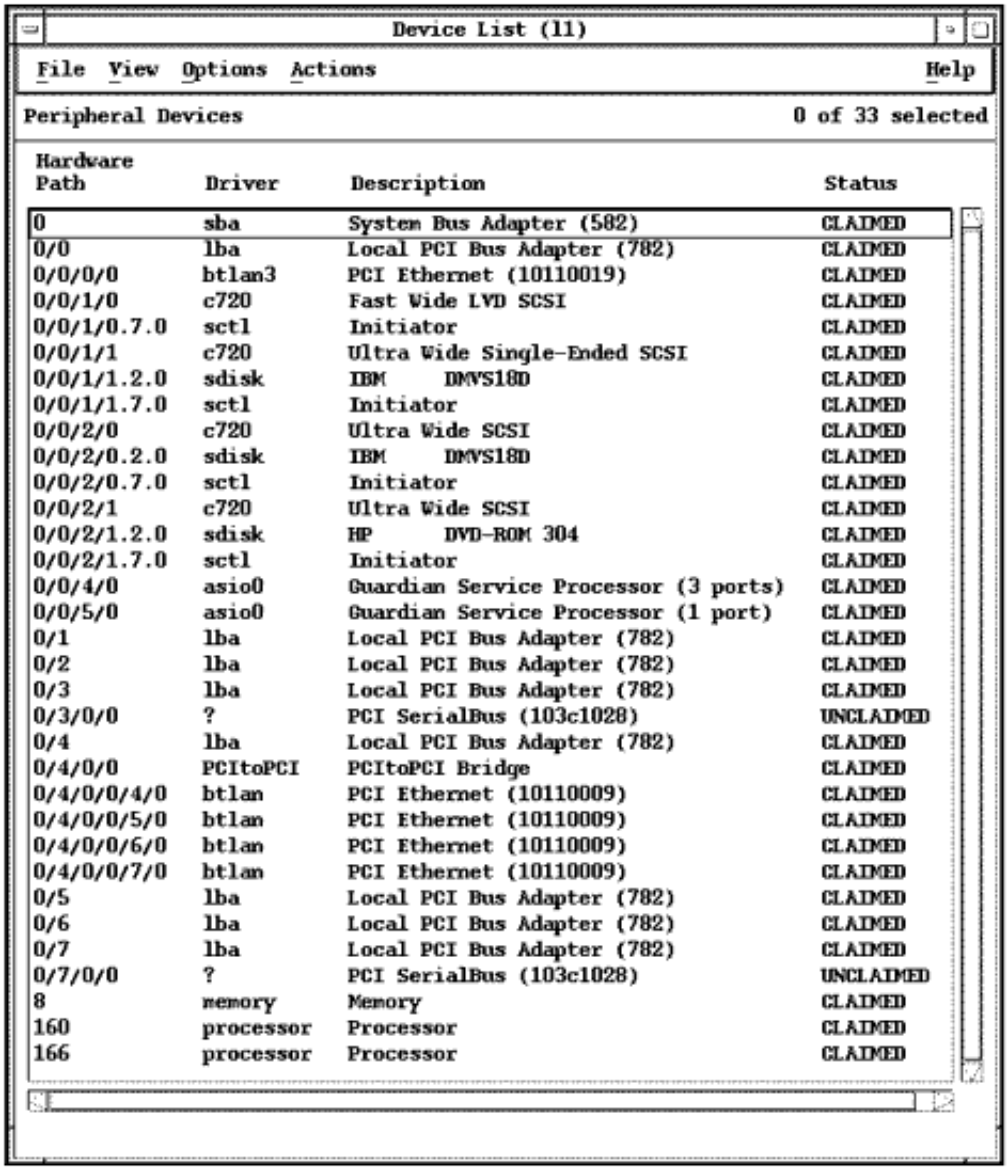
Create Device Files

Creates device files for the selected card. This takes place without any user interaction.

Device List

Device List shows all the peripherals configured into the system. Figure 10-35 shows a device list for an L-Class system:

Figure 10-35. *Peripheral Devices* Window



The two *Action* menu picks here are *Zoom* and *Show Device Files*. Selecting *Zoom* produces a window with such information as hardware path, driver, description, and status. The devices files associated with the item you have highlighted will be shown if you select *Show Device Files*.

Disks and File Systems was covered earlier in this chapter.

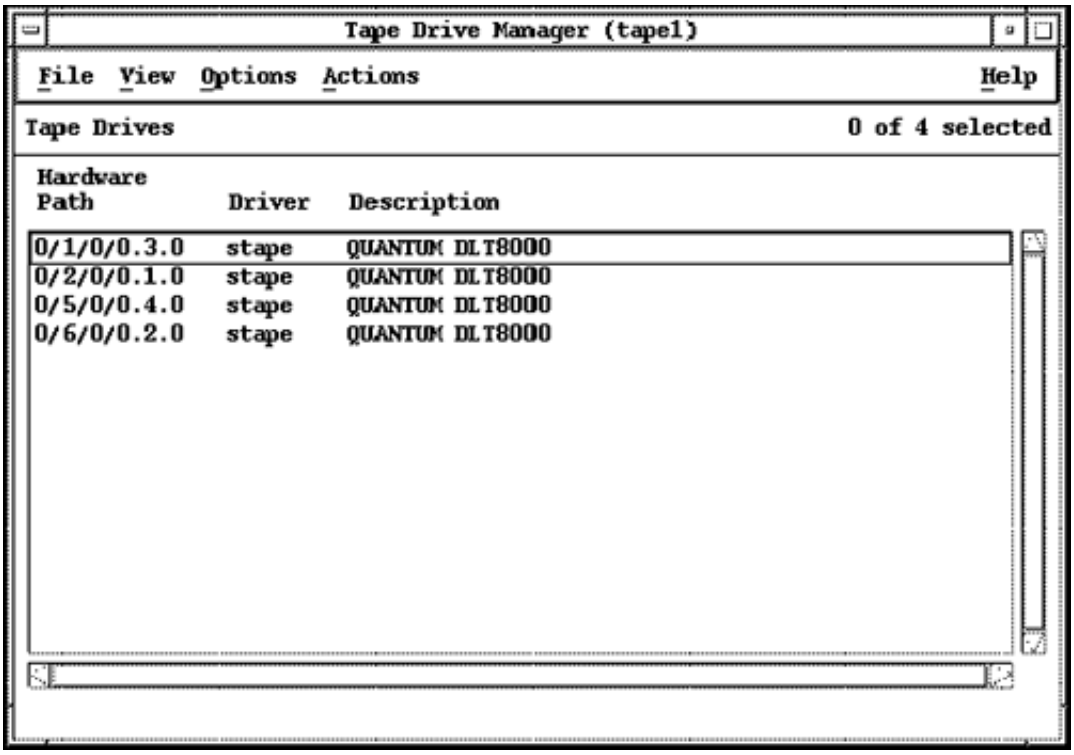
Instruments may appear if your system supports HP-IB cards.

Printers and Plotters is covered later in this chapter.

Tape Drives

Tape Drives lists the tape drives connected to your system. You are shown the *Hardware Path, Driver*, and *Description* for each tape drive. You can add, remove, diagnose tape drives, list tape drive device files, and add new tape drive device files. Figure 10-36 shows a four-drive DLT unit attached to the system:

Figure 10-36. *Tape Drives* Window



Terminals and Modems

Your system's terminals and modems are listed for you when you enter this subarea. You can perform a variety of tasks from the *Actions* menu, including the following:

- *Add Terminal or Modem*
- *Remove Terminal or Modem Port*
- *Lock Terminal or Modem Port*
- *Unlock Terminal or Modem Port*
- *Modify Terminal Security Policies*
- *Modify Terminal Security Policies*
- *Modify Terminal Authorized Users*
- *Additional Information*

Uninterruptable Power Supplies

Your system's uninterruptable power supplies are listed for you when you enter this area, including the UPS type, device file of the UPS, hardware path, port number, and whether or not shutdown is enabled. The *Actions* you can select are: *Modify Global Configuration, Add, Zoom, Remove,* and *Modify*.

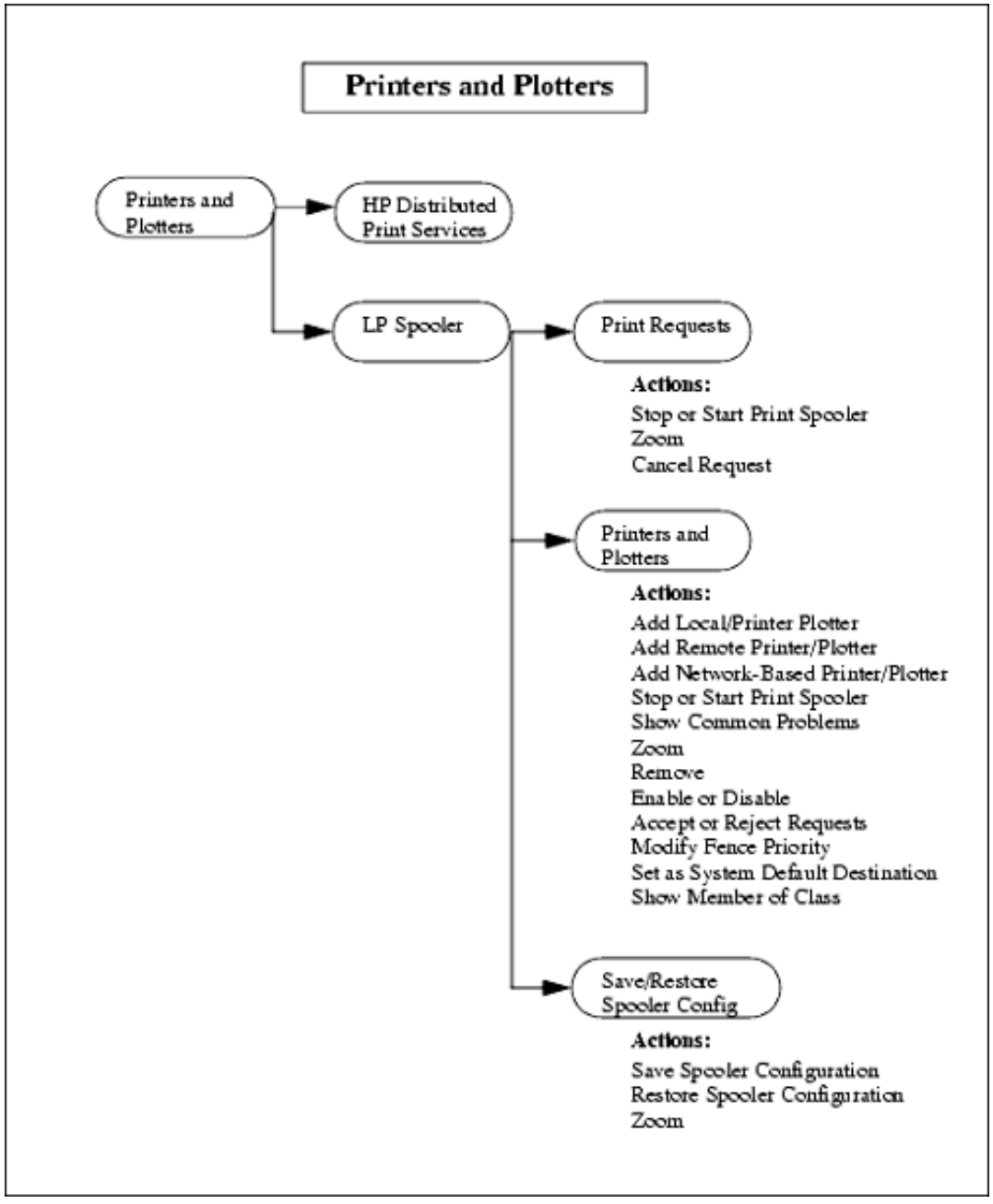
[Team LiB]

PREVIOUS NEXT

Printers and Plotters

Printers and Plotters is divided into two subareas: *HP Distributed Print Service* and *LP Spooler. HP Distributed Print Service* (HPDPS) is part of the Distributed Computing Environment (DCE). It is not covered in this book. *LP Spooler* is covered in this section; Figure 10-37 shows the hierarchy of *Printers and Plotters*:

Figure 10-37. *Printers and Plotters* Menu Structure



Print Requests

Under *Print Requests,* you can manage the print spooler and specific print jobs. You can start or stop the print spooler and cancel print jobs. The following information on print requests is listed for you:

Request ID An ID is associated with each print job. This is the Printer Name followed by a number.

Owner The name of the user who requested the print job.

Priority The priority of a print job is assigned when the job is submitted. The **-p** option of **lp** can be used to assign a priority to a job. Each print destination has a default priority that is assigned to jobs when **-p** is not used on the **lp** command.

File The name of the file sent to the print queue.

Size The size of the print job in bytes.

The *Actions* menu allows you to act on print jobs by cancelling them. In addition, the print spooler can be stopped and started.

Printers and Plotters

You can configure both local and remote printers in *Printers and Plotters*. When you select *Add Local Printer/Plotter* from the *Actions* menu and then the appropriate type of printer, a window is opened for you in which you can supply the specifics about the printer. Before this window is opened, however, you must specify whether the *type* of printer to be added is: parallel serial; HP-IB; non-standard device file; or a printer connected to a TSM terminal, as well as to which I/O card to add the printer. One huge advantage to adding the printer using SAM is that this process is entirely menu-driven, so you only have to select from among the information that is supplied.

The window appears to ask you for the following information:

Printer Name You can pick any name for the printer. I usually like to use a name that is somewhat descriptive, such as *ljet5* for a LaserJet 5. The name is limited to 14 alphanumeric characters and underscores.

Printer Model/Interface

SAM supplies a list of all interface models for you when this window is opened. These models are located in the /**usr/lib/lp/model** directory. Each printer has an interface program that is used by the spooler to send a print job to the printer. When an interface model is selected, the model is copied t0**/etc/lp/interface/**<*printername*>, where it becomes the printer's interface program. Models can be used without modification, or you can create customized interface programs.

Printer Class You can define a group of printers to be in a class, which means that print requests won't go to a specific printer but instead they will go to the first available printer within the class. This is optional.

Default Request Priority

This request defines the default priority level of all requests sent to this printer.

Default Destination

Users who do not specify a printer when requesting a print job will have the print request sent to the default printer.

You could use SAM to view printers or you could use the **lpstat** command showing printers configured on a system as shown in the following example:

```
$ /usr/bin/lpstat -t
scheduler is running
system default destination: ljet5
members of class laser:
        ljet5
device for ljet5: /dev/c1t0d0_lp
```

```
ljet5 accepting requests since Nov 21 22:45
printer ljet5 is idle. enabled since Nov 21 22:45
        fence priority : 0
no entries
```

As with all the other tasks that SAM helps you with, you can manage printers and plotters manually or you can use SAM. Not only does SAM make this easier for you, but I have also had nothing but good results having SAM do this for me. As you go through the SAM Log file, you will see a variety of **lp** commands that were issued. Some of the more common commands, including the **lpstat** command issued earlier, are listed in Table 10-1:

Table 10-1. lp Commands

COMMAND	DESCRIPTION
/usr/sbin/accept	Start accepting jobs to be queued
/usr/bin/cancel	Cancel a print job that is queued
/usr/bin/disable	Disable a device for printing
/usr/bin/enable	Enable a device for printing
/usr/sbin/lpfence	Set minimum priority for spooled file to be printed
/usr/bin/lp	Queue a job or jobs for printing
/usr/sbin/lpadmin	Configure the printing system with the options provided
/usr/sbin/lpmove	Move printing jobs from one device to another
/usr/sbin/lpsched	Start the **lp** scheduling daemon
/usr/sbin/lpshut	Stop the **lp** scheduling daemon
/usr/bin/lpstat	Show the status of printing based on the options provided
/usr/sbin/reject	Stop accepting jobs to be queued

Save/Restore Spooler Configuration

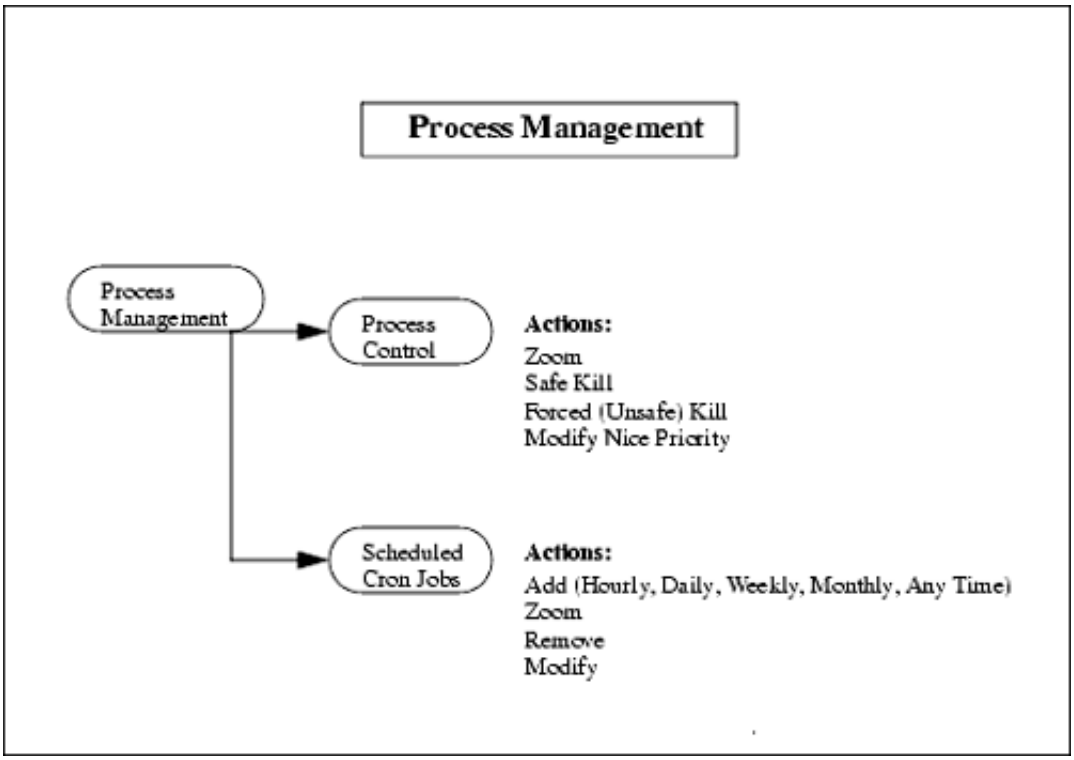
Occasionally, the spooler can get into an inconsistent state (usually something else has to go wrong with your system that ends up somehow changing or renaming some of the spooler configuration files). SAM keeps a saved version of the spooler's configuration each time it is used to make a change (only the most recent one is saved). This saved configuration can be restored by SAM to recover from the spooler having gotten into an inconsistent state. Your latest configuration is automatically saved by SAM, provided that you used SAM to create the configuration, as opposed to issuing **lp** commands at the command line, and it can be restored with *Restore Spooler Configuration* from *Save/Restore Spooler Config*. This screen allows you to save your current spooler configuration or restore a previously saved spooler configuration information.

Process Management



Process Management is broken down into two areas that allow you to control and schedule processes. *Process Control* allows you to control an individual process by performing such tasks as viewing it, changing its *nice* priority, killing it, stopping it, or continuing it. You can also view and schedule **cron** jobs under *Scheduled Cron Jobs*. Figure 10-38 shows the menu hierarchy of *Process Management*:

Figure 10-38. *Process Management* Menu Structure

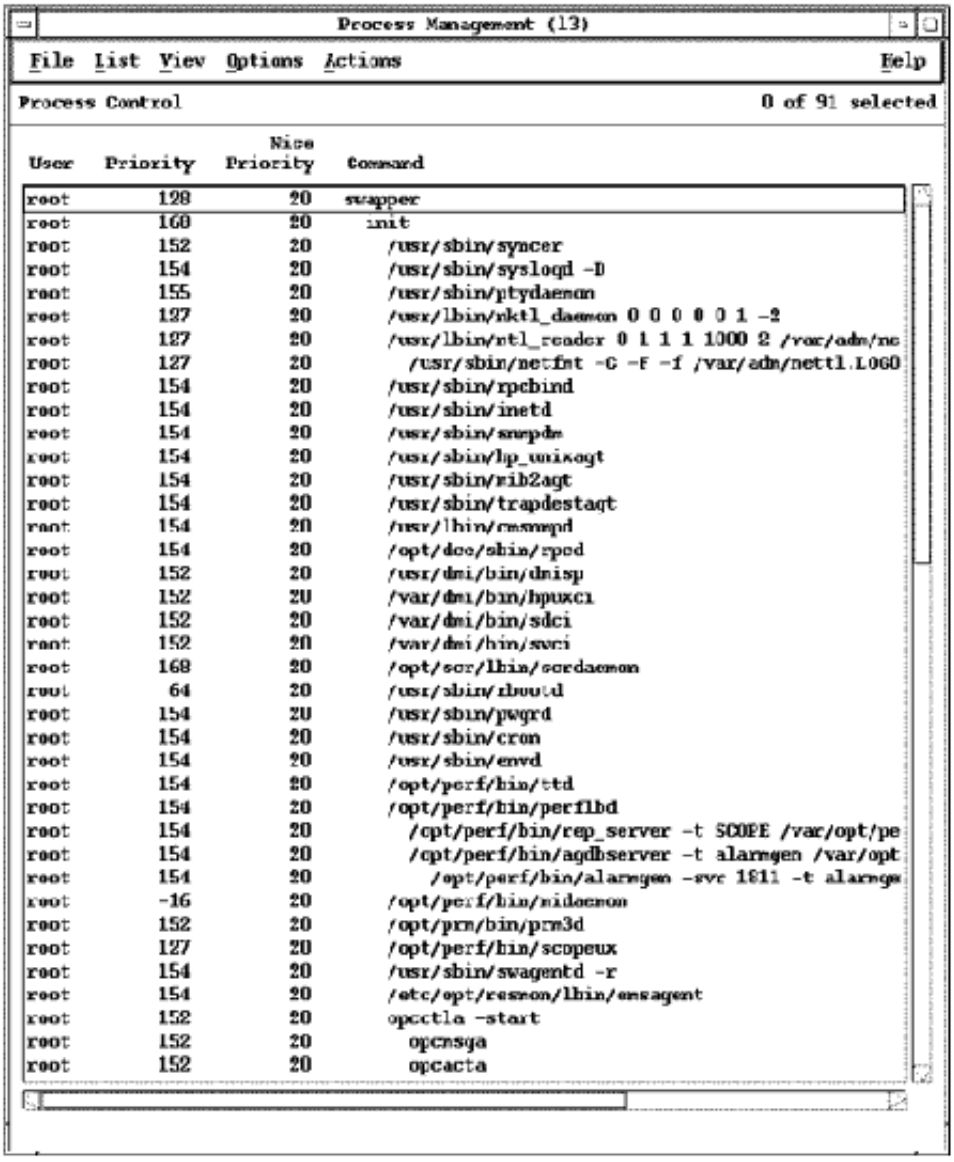


Process Control



When you pick *Process Control,* SAM lists the processes on your system and allows you to perform various actions. Using *Process Control* is a much easier way of controlling the processes on your system than is executing commands such as **ps, nice**, etc. Figure 10-39 shows a partial listing of processes:

Figure 10-39. Partial *Process Control* Listing.



There are the four columns of information listed for.

- *User* - The name of the user who owns the process.
- *Priority* - The priority of the process determines its scheduling by the CPU. The lower the number, the higher the priority. Unless you have modified these priorities, they will be default priorities. Changing the priority is done with the **nice** command.
- *Nice Priority* - If you have a process that you wish to run at a lower or higher priority, you could change this value. The lower the value, the higher the CPU scheduling priority.
- *Command* - Lists the names of all the commands currently being run or executed on the system.

In addition to these four columns, there are several others you can specify to be included in the list by selecting *Columns* from the *View* menu. You can include such information as the *Process ID, Parent Process ID, Processor Utilization, Core Image Size*, and so on. Adding *Processor Utilization* as a column, for instance, shows me how much of the processor all processes are consuming, including SAM.

You can now select one of the processes and an *Action* to perform.

When you select a process to kill and pick *Safe Kill* from the *Actions* menu, you will see a message that indicates the process number killed and that it may take a few minutes to kill it in order to terminate cleanly. If you select a process to kill and pick *Forced Kill* from the *Actions* menu, you don't get any feedback; SAM just kills the process and you move on.

Chapter 11 covered the **kill** command. To save you the trouble of flipping ahead, I have included some of the information related to **kill** here. The **kill** command can be either **/usr/bin/kill** or **kill**, which is part of the POSIX shell. The POSIX shell is the default shell for HP-UX 11i. The other shells provide their own **kill** commands as well. We use the phrase "kill a process" in the UNIX world all the time, I think, because it has a powerful connotation associated with it. What we are really saying is that we want to terminate a process. This termination is done with a signal. The most common signal to send is "SIGKILL," which terminates the process. There are other signals you can send to the process, but SIGKILL is the most common. As an alternative to sending the signal, you could send the corresponding signal number. A list of signal numbers and corresponding signals is shown as follows:

Signal Number	Signal
0	SIGNULL
1	SIGHUP
2	SIGINT
3	SIGQUIT
9	SIGKILL
15	SIGTERM
24	SIGSTOP
25	SIGTSTP
26	SIGCONT

I obtained this list of processes from the **kill** manual page.

To **kill** a process with a process ID of *234*, you would issue the following command:

```
$ kill -9 234
    |   |   |
    |   |   |> process id (PID)
    |   |> signal number
    |> kill command to terminate the process
```

The final selection from the *Actions* menu is to *Modify Nice Priority* of the process you have selected. If you were to read the manual page on **nice,** you would be very happy to see that you can modify this with SAM. Modifying the **nice** value in SAM simply requires you to select a process and specify its new **nice** value within the acceptable range.

Scheduling Cron Jobs

The *Scheduled Cron Jobs* menu selection lists all the **cron** jobs you have scheduled and allows you to *Add, Zoom, Remove*, and *Modify* **cron** jobs through the *Actions* menu. **cron** was described earlier in this chapter in the section "Backup and Recovery." I have included some of the **cron** background covered earlier to save you the trouble of flipping back.



The **crontab** file is used to schedule jobs that are automatically executed by **cron. crontab** files are in the **/var/spool/cron/crontabs** directory. **cron** is a program that runs other programs at the specified time. **cron** reads files that specify the operation to be performed and the date and time it is to be performed.

The format of entries in the **crontab** file is as follows:

minute hourmonth day month weekday user name command

minute - the minute of the hour, from 0-59
hour - the hour of the day, from 0-23
monthday - the day of the month, from 1-31
month - the month of the year, from 1-12
weekday - the day of the week, from 0 (Sunday) - 6 (Saturday)
user name - the user who will run the command if necessary
command - specifies the command line or script file to run

You have many options in the **crontab** file for specifying the *minute, hour, monthday, month,* and *weekday* to perform a task. You could list one entry in a field and then a space, several entries in any field separated by a comma, two entries separated by a dash indicating a range, or an asterisk, which corresponds to all possible entries for the field.

To list the contents of a **crontab** file, you would issue the **crontab -l** command.

[Team LiB]

PREVIOUS NEXT

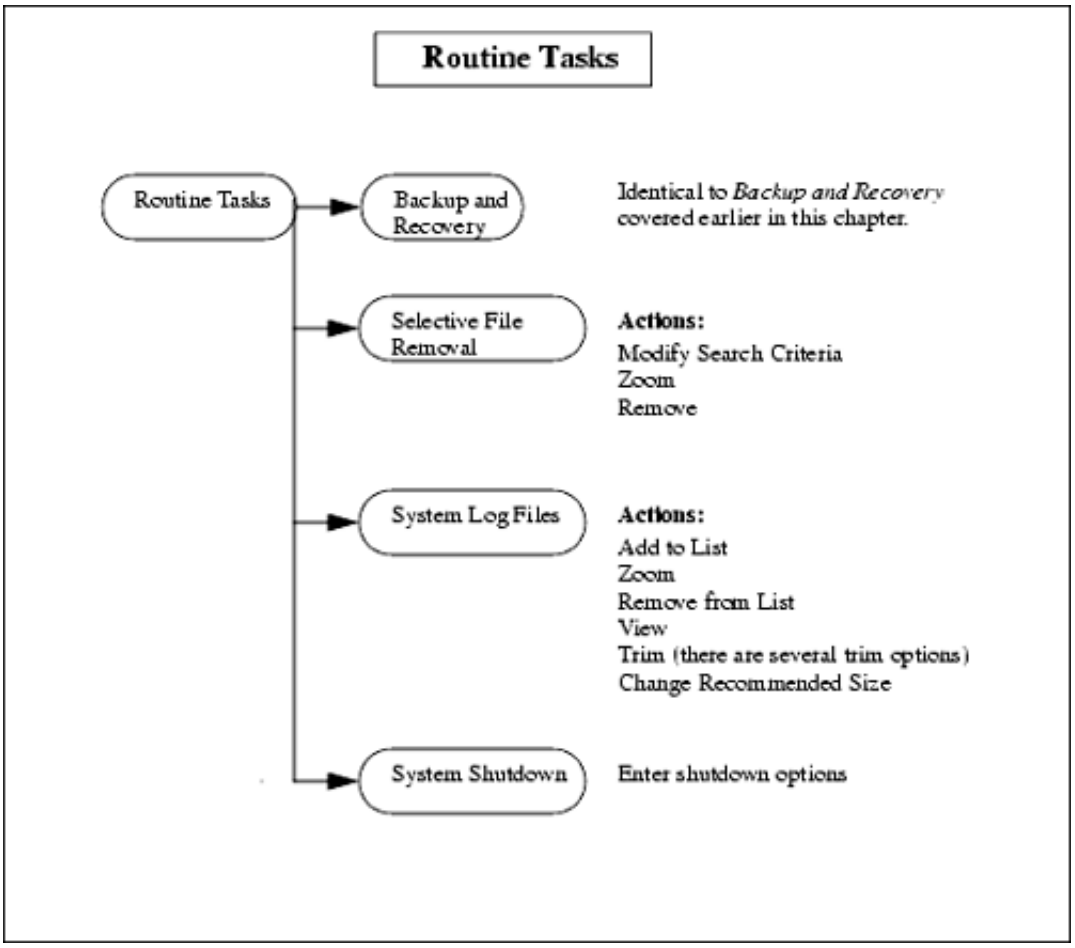
Routine Tasks

The following subareas exist under *Routine Tasks* in SAM:

- Backup and Recovery
- Find and Remove Unused Filesets
- Selective File Removal
- System Log Files
- System Shutdown

The hierarchy of *Routine Tasks* is shown in Figure 10-40. Please note that *Backup and Recovery* is identical to the SAM top-level Backup and Recovery area discussed earlier in this chapter.

Figure 10-40. *Routine Tasks* Menu Structure



Backup and Recovery

This is identical to the *Backup and Recovery* area covered earlier in this chapter.

Selective File Removal

Selective File Removal allows you to search for files to remove. You can specify a variety of criteria for selecting files to remove, including the following:

Type of file
: There are three different file types you can search for: *Large Files, Unowned Files*, and *Core Files*. A pop-up menu allows you to select which of these to search for. With *Large Files,* you are searching for files of a minimum size that haven't been modified in the specified time. *Unowned Files* are files owned by someone other than a valid system user. *Core Files* contain a core image of a terminated process when the process was terminated under certain conditions. Core files are usually related to a problem with a process and contain such information as data, stack, and so on.

Mount Points
: Specify whether or not you want to search across non-NFS mount points. If you select *Yes,* this means that the search will include mount points on your system, but not extend to NFS mount points. I chose not to include other mount points in the example.

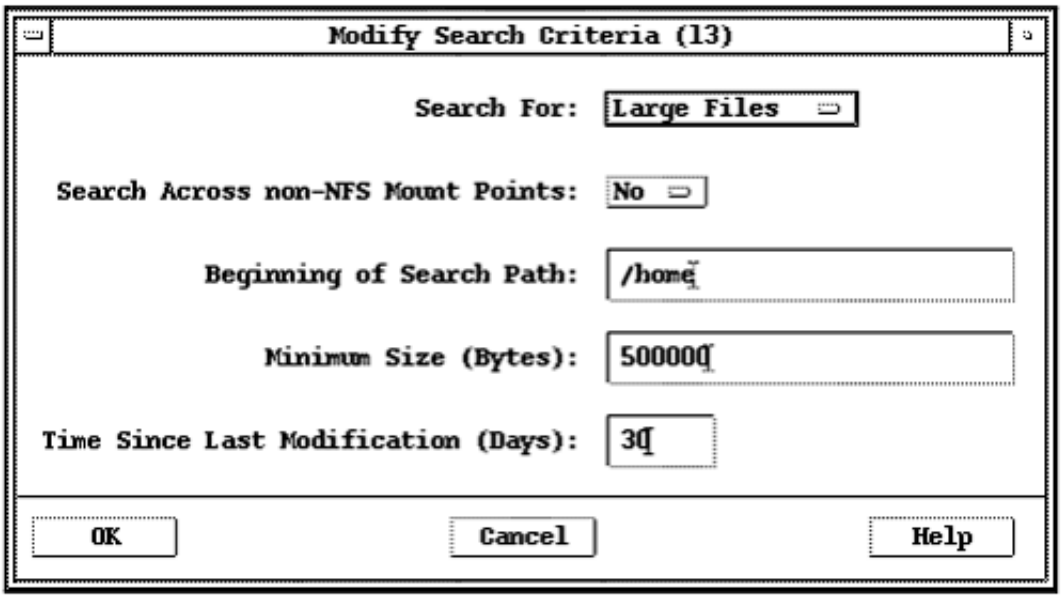
Beginning Path
: Your search can begin at any point in the system hierarchy. You can specify the start point of the search in this field. If you want to search only the **/home** directory for files, then change this entry to **/home** and you will search only that directory, as I did in the example.

Minimum Size
: Specify the smallest size file in bytes that you want to search for. Files smaller than this size will not be reported as part of the search. The minimum size in the example is 500,000 bytes.

Last Modification
: If you select *Large Files*, you can make an entry in this field. You enter the minimum number of days since the file was last modified, and files that have been modified within that time period will be excluded from the search. This is *30* days in the example.

Figure 10-41 shows an example of specifying which files to search for:

Figure 10-41. Searching for Files to Remove



The way to approach removing files is to start with an exceptionally large file size and work your way down in size. It may be that you have a few "unexpected" large files on your system that you can remove and ignore the smaller files.

System Log Files

System Log Files is used to manage the size of your system log files. Log files are generated by HP-UX for a variety of reasons, including backup, shutdown, **cron**, and so on. Your applications may very well be generating log files as well. Some of these log files can grow in size indefinitely, creating a potential catastrophe on your system by growing and crashing your system. You can be proactive and manage these log files in this subarea.

SAM is aware of many of the log files generated by HP-UX. When you enter the *System Log Files* subarea, information related to these log files is listed. You can add to the list of log files SAM knows about and have a complete list of log files presented to you each time you enter this subarea. SAM lists the following information related to log files each time you enter this subarea. (You may have to increase the size of the window to see all this information:)

File Name The full path name of the log file.

Percent Full SAM has what it thinks should be the maximum size of a log file. You can change this size by selecting *Change Recommended Size* from the *Actions* menu. The *Percent Full* is the percentage of the recommended size that the log file consumes.

Current Size The size of the file in bytes is listed for you. You may want to take a look at this. The current size of a log file may be much bigger than you would like. You could then change the recommended size and quickly see which files are greater than 100 percent. The converse may also be true. You may think the recommended size for a log file is far too small and change the recommended size to a larger value. In either case, you would like to quickly see which files are much bigger than recommended.

Recommended Size

This is what you define as the recommended size for the file. Check to make sure that you agree with this value.

Present on System

Yes if this file is indeed present on your system; *No* if it is not present on your system. If a file is not present on your system and it simply does not apply to you, then you can select *Remove from List* from the *Actions* menu. For example, you may not be running UUCP and therefore want to remove all the UUCP-related log files.

File Type The only file types listed are *ASCII* and *Non-ASCII*. I found it interesting that **/var/sam/log/samlog** was not one of the log files listed. This is not an ASCII file and must be viewed through *View SAM Log* from the *Actions* menu, but it is indeed a log file that I thought would appear in the list.

You can trim a log file using the *Trim* command from the *Actions* menu. You then have several options for trimming the file having to do with the size of the file when the trim is complete, and so on.

System Shutdown

SAM offers you the following three ways to shut down your system:

- Halt the System
- Reboot (Restart) the System
- Go to Single-User State

In addition, you can specify the number of minutes before shutdown occurs.



Run SAM on Remote Systems

I think SAM is great. If it works well on one system, then you, as the system administrator, might as well use it on other systems from a central point of control. *Run SAM on Remote Systems* allows you to set up the system on which you will run SAM remotely from a central point of control.

You can specify any number of remote systems to be controlled by a central system. With the *Actions* menu, you can:

Add System A window opens up in which you can specify the name of the remote system that you wish to administer locally.

Run SAM You can select the remote system on which you want to run SAM.

Remove System(s)

Remote systems can be removed from the list of systems on which you will run SAM remotely.





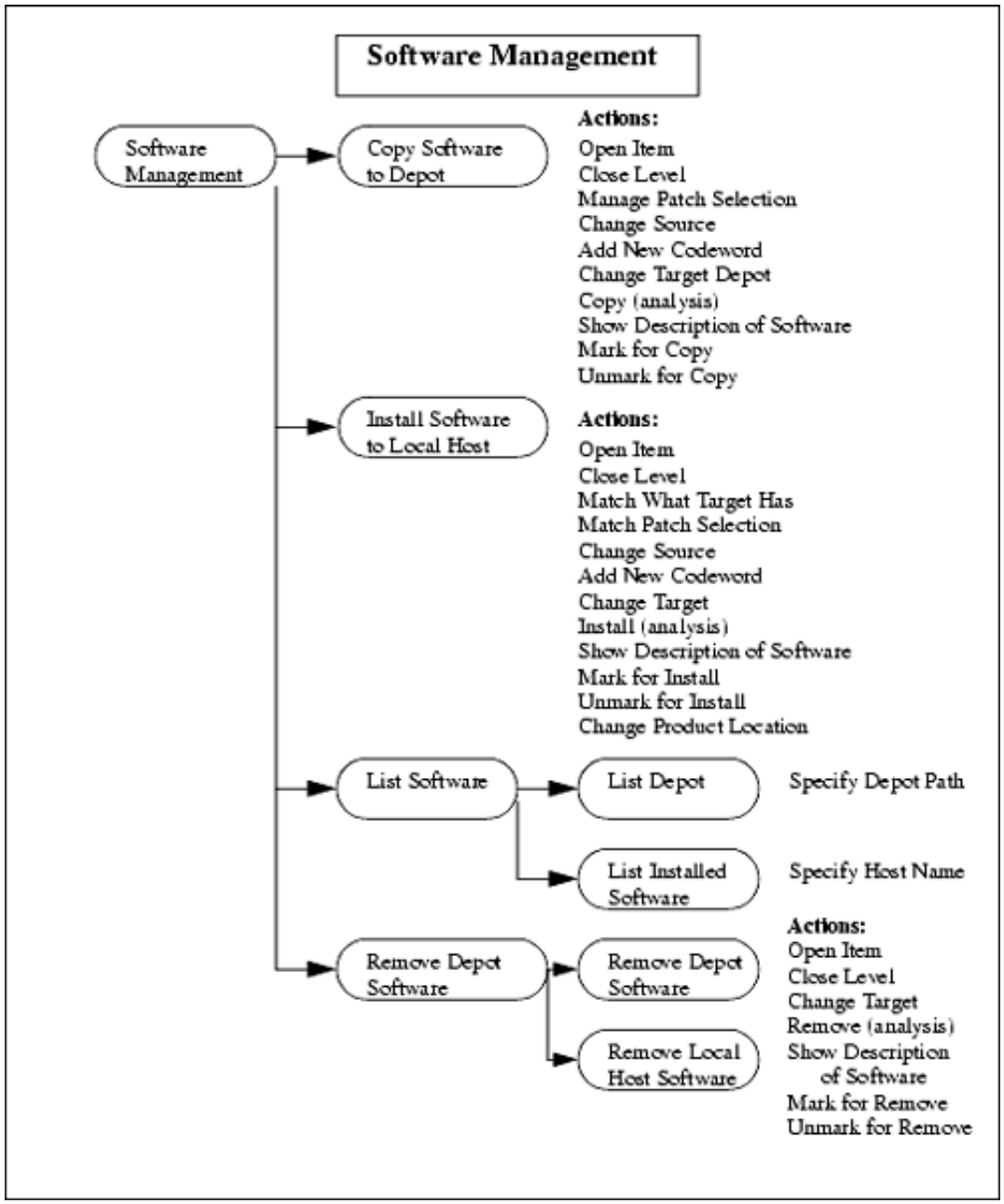
Software Management

Software Management under SAM uses Software Distributor-HP-UX (I'll call this Software Distributor), which was covered in detail in Chapter 2. SAM is giving you an interface to Software Distributor that allows you to perform software management by selecting the task you want to perform. In the end, all of the same Software Distributor commands are run, so I won't cover those again in this section. The following subareas exist under Software Management in SAM:

- Copy Software to Depot
- Install Software to Local Host
- List Software
- Remove Software

The hierarchy of *Software Management* is shown in Figure 10-42:

Figure 10-42. *Software Management* Menu Structure



As you can see from this diagram, when you select a task you want to perform SAM is selecting the Software Distributor command to run. Chapter 2 covers many of these commands.

[Team LiB]

PREVIOUS NEXT



Time

The *Time* area of SAM allows you to configure Network Time Protocol or NTP, and set the system clock. NTP is a service that allows you to synchronize the time on all your systems utilizing a Universal Coordinated Time server. I won't cover this area of SAM other than to say that NTP can be a little tricky to configure, so using SAM for this greatly simplifies NTP.





NFS Diskless Concepts

Rather than cover NFS Diskless as an area to manage, I'm going to deviate from the format found throughout this chapter and instead, provide a brief description of NFS Diskless.

This topic was introduced with HP-UX 10.x. Diskless nodes were implemented with Distributed HP-UX (DUX) in HP-UX 9.x and earlier releases. Distributed HP-UX was first introduced in HP-UX 6.0 in 1986 and was successfully used in many HP installations. The new implementation of diskless nodes as of HP-UX 10.x is NFS Diskless. It has many desirable features, including the following:

- NFS Diskless is the current de facto standard.
- It is not a proprietary solution.
- High-end diskless servers and clients can be symmetric multi-processing systems.
- Many file system types and features are available, such as UNIX File System, Journaled File System, Logical Volume Manager, disk mirroring, and so on.
- The System V Release 4 file system layout described throughout this book is implemented. This file system layout is conducive to extensive file sharing, which is used in NFS Diskless.
- Read-only NFS mounts, such as **/usr** and **/opt/**<*application*> are supported.
- Distributed HP-UX functionality, such as context-dependent files, has been removed.
- Servers can be both Series 700 and Series 800 units.
- The physical link doesn't matter, so servers can use many interfaces such as IEEE 802.3 and FDDI. A server can also assign some diskless systems to one network card, and other systems to other network cards.
- Diskless systems can boot across a gateway, thereby allowing subnets to be used.
- Booting is implemented with standard Boot Protocol (BOOTP) and Trivial File Transfer Protocol (TFTP) protocols.
- Clients can swap to a local disk or swap using NFS to a remote disk.

Many additional features of NFS Diskless exist; however, since our focus is on management, let's take a closer look at this. Using SAM, all tasks related to NFS Diskless administration can be performed. This means that you have a single point of administration for the cluster. You have cluster-wide resources, such as printers and file systems, that can be managed from any node in the cluster. You can defer some operations until a later point in time if a node is unreachable. And, of course, you can add and delete clients in SAM.

Using SAM, you get a single point of administration for several NFS Diskless systems. This means that performing an operation in SAM affects all systems in the cluster. The single point of administration areas in SAM include:

- Printers/Plotters
- File Systems
- Users/Groups

- Home Directories
- Electronic Mail
- Backups

Although a great deal could be covered on NFS Diskless and the many improvements in this area over Distributed HP-UX, the key point from an administrative perspective is that SAM provides a central point of administration for NFS Diskless administration. All tasks related to NFS Diskless administration can be performed through SAM.

[Team LiB]

PREVIOUS NEXT

ServiceControl Manager (SCM) Overview

SCM allows you to replicate tasks over multiple systems. Tasks that have traditionally been performed individually on many systems can now be performed across multiple systems in a cluster managed by SCM.



You can take a command or tool that runs on an individual system, such as the **find** or **df** command, dispatch it to run on multiple systems, and collect the results for review in SCM.

More advanced tools that were designed to run on multiple system, such as Ignite-UX, can also be run in the same manner using SCM.

Specific tasks can be assigned to specific individuals with SCM, thus creating a role-based environment. This obviates the need for root access to systems because specific management activities are assigned to specific users on specific nodes. These user roles provide great flexibility in defining the tasks that individual users can perform.

You can define the systems of which an SCM cluster is composed. The Central Management Server (CMS) in an SCM environment is the focal point in which management takes place. Agents on the managed nodes communicate with CMS to perform management functions.

SCM is available on the applications media of 11i and is loaded with Software Distributor like any other application.

After having loaded SCM from the Applications CD-ROM for HP-UX 11i, we'll run the program to set up SCM called **mxsetup**. Most of the programs related to SCM begin with "*mx*" and the application itself is loaded in **/opt/mx**, and the repository of SCM data is kept in **/var/opt/mx**.

```
# mxsetup

Please enter the ServiceControl repository password

Please re-enter the ServiceControl repository password

Please enter a name for your Managed Cluster
(default name: 'HP Managed Cluster'): servicecluster

Please enter the login name of the initial ServiceControl trusted user martyp

Would you like mxsetup to backup the ServiceControl repository after
it has initially configured it?(Y/N) [Y]:

Please enter the file path where mxsetup will store the backup data
(default path: /var/opt/mx/data/scm.backup) :

  ------------------------------------------------------------------
   You have entered,

   ServiceControl Managed Cluster Name  : servicecluster
   Initial ServiceControl trusted user  : martyp
   Backup the ServiceControl repository : y
   Backup file path                     : /var/opt/mx/data/scm.backup
  ------------------------------------------------------------------
   Would you like to continue? (Y/N) [Y] :

Please enter the file path where mxsetup will store the backup data
(default path: /var/opt/mx/data/scm.backup) :

  ------------------------------------------------------------------
   You have entered,

   ServiceControl Managed Cluster Name  : servicecluster
   Initial ServiceControl trusted user  : martyp
   Backup the ServiceControl repository : y
```

```
   Backup file path                    : /var/opt/mx/data/scm.backup
  ------------------------------------------------------------------
   Would you like to continue? (Y/N) [Y] :

   ==================================================================
   ServiceControl Manager configuration has '8' Tasks and
   it may take a while to do the configuration.

   Please wait while mxsetup configures the ServiceControl Manager...
   ==================================================================

1: Creating Agent Depots..

2: Naming the ServiceControl Managed Cluster
   as  'servicecluster'..................... OK

3: Configuring the ServiceControl daemons.......... OK

4: Starting the ServiceControl daemons.............. OK

5: Installing the AgentConfig...................... OK

6: Initializing ServiceControl..................... OK

7: Adding Tools ................................... OK

8: Backing up the Repository....................... OK

The ServiceControl Manager was successfully configured
Please see /var/opt/mx/logs/scmgr-setup.log for details.

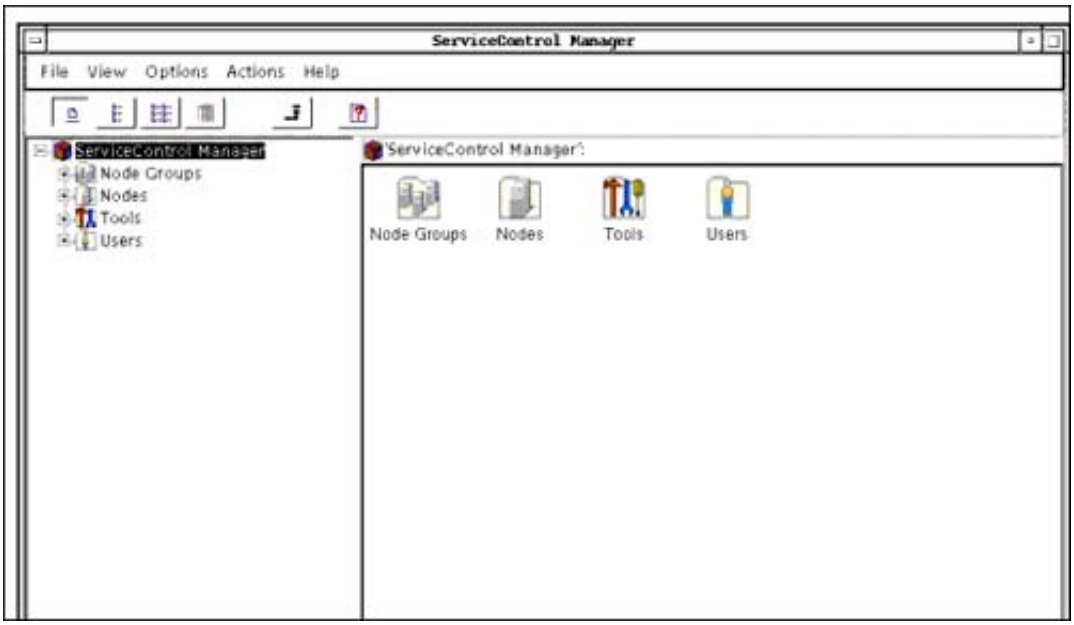
To start the ServiceControl Manager:
    - set DISPLAY environment variable
    - and enter '/opt/mx/bin/scmgr'

#
```

After the setup is complete you can issue a **ps** to view processes that begin with an *mx* to see the daemons started in order to run SCM.

With SCM having been set up we can now run the graphical interface to it with **/opt/mx/bin/scmgr** as shown in Figure 10-43:

Figure 10-43. ServiceControl Manager scmgr Screen



The four icons shown in the figure are launch points for the tasks you'll perform in SCM as described below:

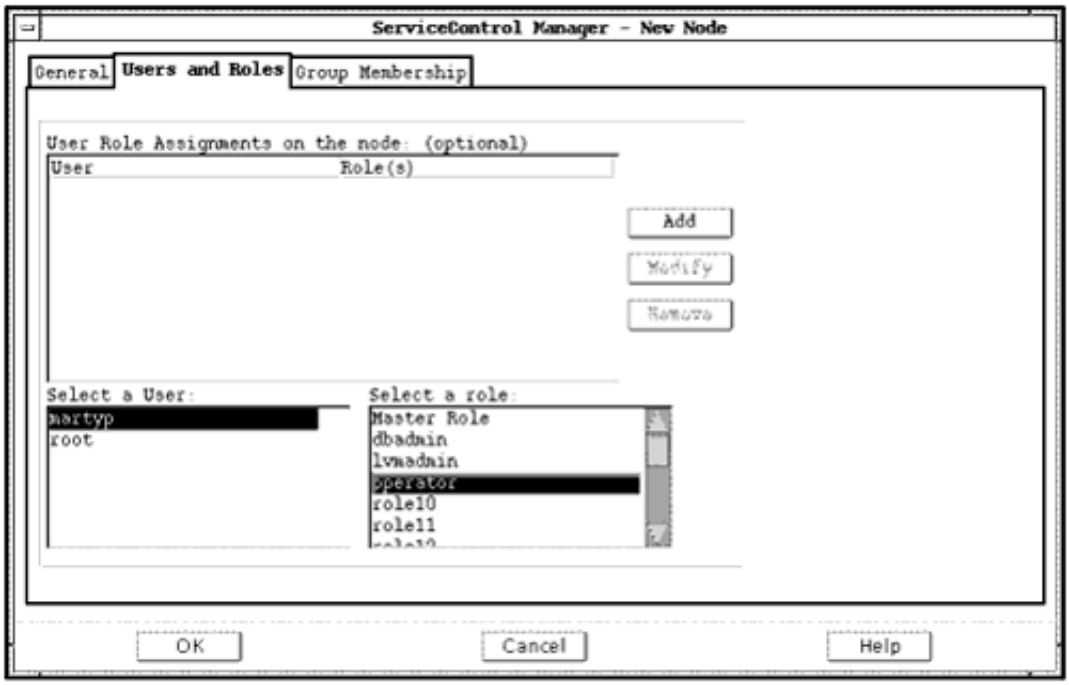
1. *Node Groups* provides a view of the managed nodes as groups. Once the node groups are displayed, a user can perform a task on a selected group of nodes.
2. *Nodes View* provides a view of all nodes that are managed. This is the point from which you can perform a variety of node-related tasks, such as adding and removing nodes.
3. *Tools View* provides a view of SCM tools.
4. *Users View* provides a view of HP-UX users who are part of SCM. A trusted user can perform a variety of user-related tasks from this view, such as adding or removing users.

If all of your systems are 11i, you won't have to load any software on clients in order to manage them from the central SCM system. If you have 11.0 or 10.20 systems, additional software is required on client systems. You can obtain the *Planning, Installing, Configuring and Updating ServiceControl Manager* document from docs.hp.com provides procedures for installing all SCM-related software.

In order to add managed nodes, you would go to the *Actions - New - Node...* menu pick and then select the nodes you wish to add.

Also under this menu pick is a tab area for *Users and Roles*. In Figure 10-44, we've added a user *martyp* with an *operator* role:

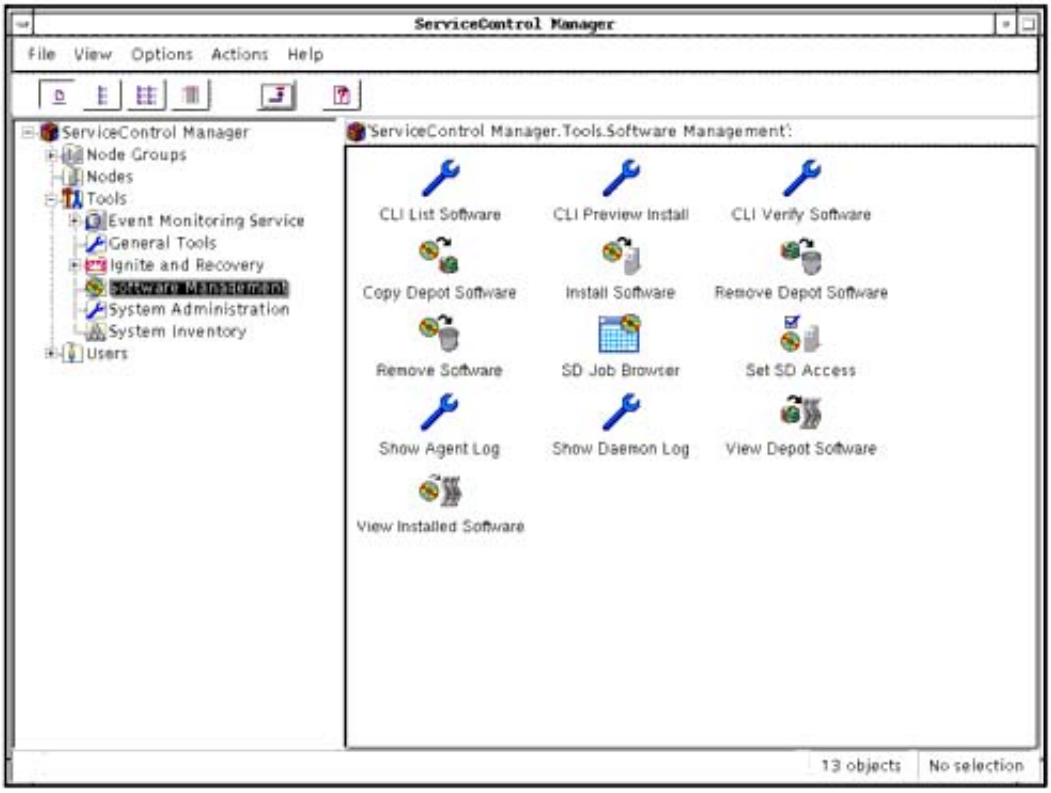
Figure 10-44. Selecting an *operator* Role for a User on a New Node



After adding nodes and grouping them, and adding users and defining their roles, you have set up the SCM management environment. You can use the pre-defined tools for users and add your own custom tools as well.

Figure 10-45 shows some of the software management tools that are predefined in SCM:

Figure 10-45. SCM *Software Management* Tools



The tools shown in Figure 10-45 are the default tools for software management. You can define and add your own tools as well. Note the categories of tools shown at the left-hand side of the figure.

From the graphical environment shown in the previous figure you can launch into SCM-related work. You can also work with SCM from the command line. Table 10-2 lists some of the more commonly used SCM commands you can issue at the command line:

Table 10-2. Commonly Used SCM Commands

COMMAND	DESCRIPTION
mxauth	Add, remove, or list authorizations in SCM.
mxexec	Run an SCM tool.
mxngroup	Add, modify, remove, or list node groups in SCM.
mxnode	Add, remove, or list nodes in SCM.
mxrepositoryrestore	Restore the SCM repository.
mxrepositorysave	Back up the SCM repository.
mxrole	List or modify SCM roles.
mxsetup	Set up or uninstall the Central Management Server of an SCM managed cluster.
mxtool	Add, modify, remove, or list tools in SCM.
mxuser	Add, modify, remove, or list users in SCM.

To view SCM roles, you would issue the following command:

```
# mxrole -lt
```

```
NAME             ENABLED?  DESCRIPTION
role16           true      For use by ServiceControl administrators
role15           true      For use by ServiceControl administrators
role14           true      For use by ServiceControl administrators
role13           true      For use by ServiceControl administrators
role12           true      For use by ServiceControl administrators
role11           true      For use by ServiceControl administrators
role10           true      For use by ServiceControl administrators
role9            true      For use by ServiceControl administrators
role8            true      For use by ServiceControl administrators
role7            true      For use by ServiceControl administrators
role6            true      For use by ServiceControl administrators
lvmadmin         true      A role for LVM Administrators
operator         true      A read-only role for operators
webadmin         true      A role for WEB Server Administrators
dbadmin          true      A role for Database Administrators
Master Role      true      The ServiceControl Master Role
#
```

This output shows the default roles that have been set up for SCM and a brief description of each. Since you can perform almost any function you want in the graphical version of SCM, I don't expect you'll be running many functions at the command line; however, it's good to know that these commands exist.

[Team LiB]

PREVIOUS NEXT

Manual Pages for Commands Used in Chapter 10

There are man pages in Appendix A for the following commands used in this Chapter:

crontab
find
iostat
ps
shutdown
vmstat



Chapter 11. Introduction to UNIX Performance Tools - Security

Introduction

Standard UNIX Commands

Killing a Process

Signals

Show System Swap

sar: The System Activity Reporter

timex to Analyze a Command

More Advanced and Graphical Performance Tools

HP GlancePlus/UX

HP OpenView Performance Agent and HP OpenView Performance Analyzer/UX

Manual Pages for Commands Used in Chapter 11



Introduction

This is a chapter on HP-UX 11i performance; however, there are examples of performance work on UNIX variants other than HP-UX 11i. I left in the examples from non-HP-UX systems because some readers of my books are coming from UNIX variants other than HP-UX. Most of the HP-UX examples in this chapter have been updated to reflect 11i systems, however, There are not any differences between these performance examples on 11i vs. 11.0.

You can take a variety of approaches to performance analysis on your system. These choices range from quick snapshots that take but a few seconds to create, to long-range capacity planning programs that you may want to run for weeks or months before you even begin to analyze the data they produce.

Standard UNIX Commands

To begin, let's look at some commands you can issue from the UNIX prompt to give you some information about your system. The commands I'll cover are:

- **iostat**
- **vmstat**
- **netstat**
- **ps**
- **kill**
- **showmount**
- **swapinfo and swap**
- **sar**
- **timex**

We'll first look at each of these commands so that you get an understanding of the output produced by them and how this output may be used. There are manual pages for some of the commands in Appendix A.

Please keep in mind that, like all topics we have covered, the output of these commands may differ somewhat among UNIX variants. The basic information produced on most UNIX variants is the same; however, the format of the outputs may differ somewhat. This usually is not significant if you're viewing the outputs; however, if you're writing programs that accept these outputs and manipulate them in some way, then the format of the outputs is important.

I/O and CPU Statistics with iostat



The **iostat** command gives you an indication of the level of effort the CPU is putting into I/O and the amount of I/O taking place among your disks and terminals. **iostat** provides a lot of useful information; however, it acts somewhat differently among UNIX variants. The following examples show issuing **iostat** on a Solaris system, an HP-UX system, and an AIX system. **iostat** was not supported on the Linux system I was using for this chapter. Note that on some systems, using the *-t* option for terminal information produces just terminal information, and on some systems it produces a full output. You will, of course, have to determine the best options for your needs on your UNIX variant. The following examples show the **iostat** command:

Here is a Solaris example executed ten times at five-second intervals:

```
# iostat 5 10
      tty          fd0           sd1           sd3           sd6           cpu
 tin tout kps tps serv  kps tps serv  kps tps serv  kps tps serv  us sy wt id
   0    0   0   0    0    0   0    0    3   0   57    0  79    0   0  7 49 43
   0   47   0   0    0    0   0    0   14   2   75    0   0    0   0  2  0 98
   0   16   0   0    0    0   0    0    0   0    0    0   0    0   0  1  0 98
   0   16   0   0    0    0   0    0    0   0    0    0   0    0   0  2  0 98
   0   16   0   0    0    0   0    0    0   0    0    0   0    0   0  0  0 100
   0   16   0   0    0    0   0    0    0   0    0    0   0    0   0  0  0 100
   0   16   0   0    0    0   0    0    0   0    0    0   0    0   0  1  0 99
   0   16   0   0    0    0   0    0    0   0    0    0   0    0   0  0  0 100
```

```
   0   16   0   0    0    0   0    0    6   1   35    0   0    0   0  4  0 96
   0   16   0   0    0    0   0    0    0   0    0    0   0    0   0  0  0 100
```

An HP-UX example includes the *-t* option executed five times at five-second intervals:

```
# iostat -t 5 5
                 tty             cpu
               tin tout        us  ni  sy  id
                 1   58         5   1  10  84

  device    bps     sps    msps

  c1t2d0      0     0.0     1.0


                 tty             cpu
               tin tout        us  ni  sy  id
                 0   30         0   2  26  72

  device    bps     sps    msps

  c1t2d0    484    249.6     1.0


                 tty             cpu
               tin tout        us  ni  sy  id
                 0   31         1   3  23  73

  device    bps     sps    msps

  c1t2d0    517    256.1     1.0


                 tty             cpu
               tin tout        us  ni  sy  id
                 0   35         0   2  23  75

  device    bps     sps    msps

  c1t2d0    456    254.4     1.0


                 tty             cpu
               tin tout        us  ni  sy  id
                 0  744         1   6  38  55

  device    bps     sps    msps

  c1t2d0    155    83.1     1.0


#
```



Here is an AIX example executed ten times at five-second intervals:

```
# iostat 5 10

tty:      tin         tout   avg-cpu:  % user    % sys     % idle    % iowait
          0.0          0.0               0.3      1.0       98.4       0.3

Disks:        % tm_act     Kbps      tps    Kb_read   Kb_wrtn
hdisk0           0.4       2.7       0.4    2366635    959304
hdisk1           0.0       0.0       0.0      18843     37928
hdisk2           0.1       0.6       0.1     269803    423284
hdisk3           0.0       0.0       0.0      20875       172
cd0              0.0       0.0       0.0         14         0

tty:      tin         tout   avg-cpu:  % user    % sys     % idle    % iowait
          0.0        108.2               0.0      0.2       99.8       0.0
```

```
Disks:        % tm_act     Kbps      tps    Kb_read   Kb_wrtn
hdisk0           0.0       0.0       0.0          0         0
hdisk1           0.0       0.0       0.0          0         0
hdisk2           0.0       0.0       0.0          0         0
hdisk3           0.0       0.0       0.0          0         0
cd0              0.0       0.0       0.0          0         0

tty:      tin         tout   avg-cpu:  % user    % sys     % idle    % iowait
          0.0        108.4               0.2      0.8       99.0       0.0

Disks:        % tm_act     Kbps      tps    Kb_read   Kb_wrtn
hdisk0           0.0       0.0       0.0          0         0
hdisk1           0.0       0.0       0.0          0         0
hdisk2           0.0       0.0       0.0          0         0
hdisk3           0.0       0.0       0.0          0         0
cd0              0.0       0.0       0.0          0         0

tty:      tin         tout   avg-cpu:  % user    % sys     % idle    % iowait
          0.0        108.4               0.4      0.2       99.4       0.0

Disks:        % tm_act     Kbps      tps    Kb_read   Kb_wrtn
hdisk0           0.0       0.0       0.0          0         0
hdisk1           0.0       0.0       0.0          0         0
hdisk2           0.0       0.0       0.0          0         0
hdisk3           0.0       0.0       0.0          0         0
cd0              0.0       0.0       0.0          0         0

tty:      tin         tout   avg-cpu:  % user    % sys     % idle    % iowait
          0.0        108.2               0.4      0.6       99.0       0.0

Disks:        % tm_act     Kbps      tps    Kb_read   Kb_wrtn
hdisk0           0.0       0.0       0.0          0         0
hdisk1           0.0       0.0       0.0          0         0
hdisk2           0.0       0.0       0.0          0         0
hdisk3           0.0       0.0       0.0          0         0
cd0              0.0       0.0       0.0          0         0

tty:      tin         tout   avg-cpu:  % user    % sys     % idle    % iowait
          0.0        108.4               0.0      0.4       99.6       0.0

Disks:        % tm_act     Kbps      tps    Kb_read   Kb_wrtn
hdisk0           0.0       0.0       0.0          0         0
hdisk1           0.0       0.0       0.0          0         0
hdisk2           0.0       0.0       0.0          0         0
hdisk3           0.0       0.0       0.0          0         0
cd0              0.0       0.0       0.0          0         0

tty:      tin         tout   avg-cpu:  % user    % sys     % idle    % iowait
          0.0        108.4               0.6      0.0       99.4       0.0

Disks:        % tm_act     Kbps      tps    Kb_read   Kb_wrtn
hdisk0           0.0       0.0       0.0          0         0
hdisk1           0.0       0.0       0.0          0         0
hdisk2           0.0       0.0       0.0          0         0
hdisk3           0.0       0.0       0.0          0         0
cd0              0.0       0.0       0.0          0         0

tty:      tin         tout   avg-cpu:  % user    % sys     % idle    % iowait
          0.0        108.2               0.2      0.8       99.0       0.0

Disks:        % tm_act     Kbps      tps    Kb_read   Kb_wrtn
hdisk0           0.0       0.0       0.0          0         0
hdisk1           0.0       0.0       0.0          0         0
hdisk2           0.0       0.0       0.0          0         0
hdisk3           0.0       0.0       0.0          0         0
cd0              0.0       0.0       0.0          0         0

tty:      tin         tout   avg-cpu:  % user    % sys     % idle    % iowait
          0.0        108.4               0.4      0.0       99.6       0.0

Disks:        % tm_act     Kbps      tps    Kb_read   Kb_wrtn
hdisk0           0.0       0.0       0.0          0         0
hdisk1           0.0       0.0       0.0          0         0
hdisk2           0.0       0.0       0.0          0         0
```

```
hdisk3            0.0       0.0       0.0          0         0
cd0               0.0       0.0       0.0          0         0

tty:      tin         tout   avg-cpu:  % user    % sys     % idle    % iowait
          0.0        108.4                0.4       0.4       99.2       0.0

Disks:        % tm_act     Kbps      tps    Kb_read   Kb_wrtn
hdisk0            0.0       0.0       0.0          0         0
hdisk1            0.0       0.0       0.0          0         0
hdisk2            0.0       0.0       0.0          0         0
hdisk3            0.0       0.0       0.0          0         0
cd0               0.0       0.0       0.0          0         0
```



The following are descriptions of the reports you receive with **iostat** for terminals, the CPU, and mounted file systems. Because the reports are somewhat different, I have included detailed information from the HP-UX output. A more detailed description of these fields is included in the **iostat** manual page in Appendix A. Most of the fields appear in the outputs; however, the outputs of the commands differ somewhat among UNIX variants.

For every terminal you have connected (*tty*), you see a "tin" and "tout," which represent the number of characters read from your terminal and the number of characters written to your terminal, respectively.

For your CPU, you see the percentage of time spent in user mode ("us"), the percentage of time spent running user processes at a low priority called nice ("ni"), the percentage of time spent in system mode ("sy"), and the percentage of time the CPU is idle ("id").

For every locally mounted file system, you receive information on the kilobytes transferred per second ("bps"), number of seeks per second ("sps"), and number of milliseconds per average seek ("msps"). For disks that are NFS-mounted or disks on client nodes of your server, you will not receive a report; **iostat** reports only on locally mounted file systems.

When viewing the output of **iostat**, there are some parameters to take note of.

First, note that the time that your CPU is spending in the four categories shown. The CPU report is produced with the *-t* option. I have worked on systems with poor performance that the administrator assumed to be a result of a slow CPU because the "id" number was very high, indicating that the CPU was actually idle most of the time. If the CPU is mostly idle, the chances are that the bottleneck is not the CPU, but may be I/O, memory, or networking. If the CPU is indeed busy most of the time ("id" is very low), see whether any processes are running "nice" (check the "ni" number). It may be that there are some background processes consuming a lot of CPU time that can be changed to run "nice."

Second, compare the number of transfers taking place. These are usually indicated by something like blocks per second (bps), transfers per second (*tps*), or seeks per second (*sps)*. These numbers give an indication of the amount of activity taking place on a disk. If one volume is consistently much higher than other volumes, then it may be performing an inordinate amout of the workload. Notice on HP-UX that the milliseconds per average seek (*msps*) for all disks is always equal to one.

Virtual Memory Statistics with vmstat



vmstat provides virtual memory statistics. It provides information on the status of processes, virtual memory, paging activity, faults, and a breakdown of the percentage of CPU time. **vmstat** acts somewhat differently among UNIX variants. The following examples show issuing **vmstat** on a Solaris system, an HP-UX system, an AIX system, and a Linux system. You will, of course, have to determine the best options for your needs on your UNIX variant. In the following examples, the output was produced nine times at five-second intervals. The first argument to the **vmstat** command is the interval; the second is the number of times you would like the output produced.

Solaris example:

```
# vmstat 5 9
 procs     memory            page            disk          faults      cpu
 r b w   swap  free  re  mf pi po fr de sr f0 s1 s3 s6   in   sy   cs us sy id
 0 0 0   4480  4696   0   0  1  0  0  0  0  0  0  0 79  864  130  297  0  7 92
 0 0 0 133020  5916   0   3  0  0  0  0  0  0  0  3  0  102   42   24  0  2 98
 0 0 0 133020  5916   0   0  0  0  0  0  0  0  0  0  0   70   48   24  0  0 100
 0 0 0 133020  5916   0   0  0  0  0  0  0  0  0  0  0   74   42   24  0  0 100
```

```
 0 0 0 133020  5916   0   0  0  0  0  0  0  0  0  0   35   45   23  0  0 99
 0 0 0 133020  5916   0   0  0  0  0  0  0  0  0  0   65   66   26  0  0 100
 0 0 0 133020  5916   0   0  0  0  0  0  0  0  0  0   52   44   23  0  1 99
 0 0 0 133020  5916   0   0  0  0  0  0  0  0  0  0   53   54   24  0  1 99
 0 0 0 133020  5916   0   0  0  0  0  0  0  0  1  0   60   53   25  0  2 98
```

HP-UX example:

```
# vmstat 5 9

procs       memory                  page                    faults         cpu
r   b  w    avm    free   re  at   pi   po   fr   de   sr    in     sy    cs  us sy id
5 240 0 17646    3979    2   0    0    0    0    0    0     0    778   193  17  3 80
4 242 0 16722    4106    0   0    0    0    0    0    0   814  20649   258  89 10  2
4 240 0 16649    4106    0   0    0    0    0    0    0    83  18384   218  91  9  0
4 240 0 16468    4106    0   0    0    0    0    0    0   792  19552   273  89 11  1
5 239 0 15630    4012    9   0    0    0    0    0    0   804  18295   270  93  8 -1
5 241 0 16087    3934    6   0    0    0    0    0    0   920  21044   392  89 10  0
5 241 0 15313    3952   11   0    0    0    0    0    0   968  20239   431  90 10  0
4 242 0 16577    4043    3   0    0    0    0    0    0   926  19230   409  89 10  0
6 238 0 17453    4122    0   0    0    0    0    0    0   837  19269   299  89  9  2
```

AIX example:

```
martyp $ vmstat 5 9
kthr     memory             page              faults        cpu
----- ----------- ------------------------ ------------ -----------
 r  b   avm   fre  re  pi  po  fr   sr  cy  in   sy  cs us sy id wa
 0  0 16604   246   0   0   0   0    2   0 149   79  36  0  1 98  0
 0  0 16604   246   0   0   0   0    0   0 153  125  41  0  0 99  0
 0  0 16604   246   0   0   0   0    0   0 143   83  33  0  0 99  0
 0  0 16604   246   0   0   0   0    0   0 140   94  35  0  1 99  0
 0  0 16604   246   0   0   0   0    0   0 166   62  32  0  0 99  0
 0  0 16604   246   0   0   0   0    0   0 150  102  38  1  0 99  0
 0  0 16604   246   0   0   0   0    0   0 183   78  34  0  0 99  0
 0  0 16604   246   0   0   0   0    0   0 132   87  33  0  1 99  0
 0  0 16604   246   0   0   0   0    0   0 147   84  38  0  0 99  0
```

Linux example:

```
# vmstat 5 5
   procs                  memory    swap          io     system         cpu
 r  b  w   swpd   free   buff  cache  si  so    bi    bo   in    cs  us  sy  id
 1  0  0   9432   1160    656  12024   1   2    14     1  138   274   3   1  96
 1  0  0   9684    828    652  12148   0  50     0    14  205  8499  82  18   0
 1  0  0   9684    784    652  11508   0   0     0     1  103  8682  81  19   0
 1  0  0   9684    800    652  10996   0   0     0     0  101  8683  80  20   0
 0  0  0   9772    796    652   9824  12  18     3     4  160  6577  66  17  18
```



You certainly get a lot for your money out of the **vmstat** command. Here is a brief description of the categories of information produced by **vmstat**. I have included a description of the fields in the HP-UX example because of the manual page that appears in Appendix A for HP-UX. You can see, however, that the outputs are very similar.

Processes are classified into one of three categories: runnable ("r"), blocked on I/O or short-term resources ("b"), or swapped ("w").

Next you will see information about memory. "avm" is the number of virtual memory pages owned by processes that have run within the last 20 seconds. If this number is roughly the size of physical memory minus your kernel, then you are near forced paging. The "free" column indicates the number of pages on the system's free list. It doesn't mean that the process is finished running and these pages won't be accessed again; it just means that they have not been accessed recently. I suggest that you ignore this column.

Next is paging activity. The first field ("re") shows the pages that were reclaimed. These pages made it to the free list but were later referenced and had to be salvaged.

Next you see the number of faults in three categories: interrupts per second, which usually come from hardware ("in"), system calls per second ("sy"), and context switches per second ("cs").

The final output is CPU usage percentage for user ("us"), system ("sy"), and idle



("id"). This is not as complete as the **iostat** output, which also shows **nice** entries.

If you are running an I/O intensive workload you may indeed see a lot of activity in runnable processes ("r"), blocked processes("b"), and the runnable but swapped ("w") processes. If you have many runnable but swapped processes, then you probably have an I/O bottleneck.

Network Statistics with netstat



netstat provides information related to network statistics. Because network bandwidth has as much to do with performance as the CPU and memory in some networks, you want to get an idea of the level of network traffic you have.

I use two forms of **netstat** to obtain network statistics. The first is **netstat -i**, which shows the state of interfaces that are autoconfigured. Although **netstat -i** gives a good rundown of the primary LAN interface, such as the network it is on, its name, and so on, it does not show useful statistical information.

The following shows the output of **netstat -i**:

netstat -i

Name	Mtu	Network	Address	Ipkts	Ierrs	Opkts	Oerrs	Col
lan0	1497	151.150	a4410.e.h.c	242194	120	107665	23	19884



netstat provides a concise output. Put another way, most of what you get from **netstat** is useful. Here is a description of the nine fields in the **netstat** example:

Name	The name of your network interface (Name), in this case, "lan0."
Mtu	The "maximum transmission unit," which is the maximum packet size sent by the interface card.
Network	The network address of the LAN to which the interface card is connected (151.150).
Address	The host name of your system. This is the symbolic name of your system as it appears in the **/etc/hosts** file if your networking is configured to use **/etc/hosts**.

Below is the statistical information. Depending on the system you are using, or revision of OS, you may not see some of these commands:

Ipkts	The number of packets received by the interface card, in this case, "lan0."
Ierrs	The number of errors detected on incoming packets by the interface card.
Opkts	The number of packets transmitted by the interface card.
Oerrs	The number of errors detected during the transmission of packets by the interface card.
Col	The number of collisions that resulted from packet traffic.



netstat provides cumulative data since the node was last powered up; therefore you might have a long elapsed time over which data was accumulated. If you are interested in seeing useful statistical information, you can use **netstat** with different options. You can also specify an interval to report statistics. I usually ignore the first entry, because it shows all data since the system was last powered up. This means that the data includes non-prime hours when the system was idle. I prefer to view data at the time the system is working its hardest. The following examples show running **netstat -I** and specifying the *lan* interface for Solaris, HP-UX, and AIX. These outputs are nearly identical, although the name of the network interface does vary among UNIX variants. The **netstat** command is run at an interval of five seconds. The Linux version of this command, which is not shown, does not allow me to specify an interval.

Solaris example:

```
# netstat -I le0 5
    input   le0       output           input  (Total)    output
packets errs  packets errs  colls  packets errs  packets errs  colls
116817990 0     3299582 11899 1653100 116993185 0     3474777 11899 1653100
185     0     3       0     0       185     0     3       0     0
273     0     8       0     0       273     0     8       0     0
153     0     3       0     0       153     0     3       0     0
154     0     3       0     0       154     0     3       0     0
126     0     3       0     0       126     0     3       0     0
378     0     2       0     0       378     0     2       0     0
399     0     4       0     0       399     0     4       0     0
286     0     2       0     0       286     0     2       0     0
```

HP-UX example:

```
# netstat -I lan0 5

   (lan0)-> input           output          (Total)-> input          output
       packets  errs  packets   errs  colls       packets  errs  packets  errs  colls
      269841735 27   256627585   1    5092223    281472199  27   268258048 1    5092223
          1602     0     1238      0      49          1673     0     1309     0     49
          1223     0     1048      0      25          1235     0     1060     0     25
          1516     0     1151      0      42          1560     0     1195     0     42
          1553     0     1188      0      17          1565     0     1200     0     17
          2539     0     2180      0      44          2628     0     2269     0     44
          3000     0     2193      0     228          3000     0     2193     0    228
          2959     0     2213      0     118          3003     0     2257     0    118
          2423     0     1981      0      75          2435     0     1993     0     75
```

AIX example:

```
# netstat -I en0 5
    input    (en0)     output           input   (Total)    output
 packets  errs  packets  errs colls  packets  errs  packets  errs colls
46333531     0  1785025     0     0 47426087     0  2913405     0     0
     203     0        1     0     0      204     0        2     0     0
     298     0        1     0     0      298     0        1     0     0
     293     0        1     0     0      304     0       12     0     0
     191     0        1     0     0      191     0        1     0     0
     150     0        2     0     0      151     0        3     0     0
     207     0        3     0     0      218     0       15     0     0
     162     0        3     0     0      162     0        4     0     0
     120     0        2     0     0      120     0        2     0     0
```

With this example, you get multiple outputs of what is taking place on the LAN interface, including the totals on the right side of the output. As I mentioned earlier, you may want to ignore the first output, because it includes information over a long time period. This may include a time when your network was idle, and therefore the data may not be important to you.

You can specify the network interface on which you want statistics reported by using **-I interface**; in the case of the example, it was **-I** and either *le0, lan0*, or *en0*. An interval of five seconds was also used in this example.

Analyzing **netstat** statistical information is intuitive. You want to verify that the collisions (Colls) are much lower than the packets transmitted (Opkts). Collisions occur on output from your LAN interface. Every collision your LAN interface encounters slows down the network. You will get varying opinions about what is too many collisions. If your collisions are less than 5 percent of "Opkts," you're probably in good shape and better off spending your time analyzing some other system resource. If this number is high, you may want

to consider segmenting your network in some way such as installing networking equipment between portions of the network that don't share a lot of data.

As a rule of thumb, if you reduce the number of packets you are receiving and transmitting ("Ipkts" and "Opkts"), then you will have less overall network traffic and fewer collisions. Keep this in mind as you plan your network or upgrades to your systems. You may want to have two LAN cards in systems that are in constant communication. That way, these systems have a "private" LAN over which to communicate and do not adversely affect the performance of other systems on the network. One LAN interface on each system is devoted to intrasystem communication. This provides a "tight" communication path among systems that usually act as servers. The second LAN interface is used to communicate with any systems that are usually clients on a larger network.



You can also obtain information related to routing with **netstat** (see Chapter 12). The *-r* option to **netstat** shows the routing tables, which you usually want to know about, and the *-n* option can be used to print network addresses as numbers rather than as names. In the following examples, **netstat** is issued with the *-r* option (this will be used when describing the **netstat** output) and the *-rn* options, so that you can compare the two outputs:

$ netstat -r

Routing tables

Destination	Gateway	Flags	Refs	Use	Interface	Pmtu
hp700	localhost	UH	0	28	lo0	4608
default	router1	UG	0	0	lan0	4608
128.185.61	system1	U	347	28668	lan0	1500

$ netstat -rn

Routing tables

Destination	Gateway	Flags	Refs	Use	Interface	Pmtu
127.0.0.1	127.0.0.1	UH	0	28	lo0	4608
default	128.185.61.1	UG	0	0	lan0	4608
128.185.61	128.185.61.2	U	347	28668	lan0	1500

With **netstat**, some information is provided about the router, which is the middle entry. The *-r* option shows information about routing, but there are many other useful options to this command are available. Of particular interest in this output is "Flags," which defines the type of routing that takes place. Here are descriptions of the most common flags, which may be different among UNIX variants, from the manual page in Appendix A.

1=*U*	Route to a *network* via a gateway that is the local host itself.
3=*UG*	Route to a *network* via a gateway that is the remote host.
5=*UH*	Route to a *host* via a gateway that is the local host itself.
7=*UGH*	Route to a *host* via a remote gateway that is a host.



The first line is for the local host, or loopback interface called, *lo0* at address 127.0.0.1 (you can see this address in the **netstat -rn** example). The *UH* flags indicate that the destination address is the local host itself. This Class A address allows a client and server on the same host to communicate with one another via TCP/IP. A datagram sent to the loopback interface won't go out onto the network; it will simply go through the loopback.

The second line is for the default route. This entry says to send packets to Router 1 if a more specific route

can't be found. In this case, the router has a *UG* under *Flags*. Some routers are configured with a *U*; others, such as the one in this example, with a *UG.* I've found that I usually end up determining through trial and error whether a *U* or *UG* is required. If there is a *U* in *Flags* and I am unable to ping a system on the other side of a router, a *UG* entry usually fixes the problem.

The third line is for the system's network interface, *lan0*. This means to use this network interface for packets to be sent to 128.185.61.

Check Processes with ps

Knowing about the processes running on your system, and knowing how to stop them, are important to both system administration and performance.



To find the answer to "What is my system doing?," use **ps -ef**. This command provides information about every running process on your system. If, for instance, you want to know whether NFS is running, you simply type **ps -ef** and look for NFS daemons. Although **ps** tells you every process that is running on your system, it doesn't provide a good summary of the level of system resources being consumed. I would guess that **ps** is the most often issued system administration command. There are a number of options you can use with **ps**. I normally use *e* and *f*, which provide information about every ("*e*") running process and lists this information in full ("*f*"). **ps** outputs are almost identical from system to system. The following three examples are from a Solaris, AIX, and HP-UX system, respectively:

Solaris example:

```
martyp $ ps -ef
     UID   PID  PPID  C    STIME TTY      TIME CMD
    root     0     0  0   Feb 18 ?        0:01 sched
    root     1     0  0   Feb 18 ?        1:30 /etc/init -
    root     2     0  0   Feb 18 ?        0:02 pageout
    root     3     0  1   Feb 18 ?      613:44 fsflush
    root  3065  3059  0   Feb 22 ?        5:10 /usr/dt/bin/sdtperfmeter -f -H -r
    root    88     1  0   Feb 18 ?        0:01 /usr/sbin/in.routed -q
    root   478     1  0   Feb 18 ?        0:00 /usr/lib/saf/sac -t 300
    root    94     1  0   Feb 18 ?        2:50 /usr/sbin/rpcbind
    root   150     1  0   Feb 18 ?        6:03 /usr/sbin/syslogd
    root    96     1  0   Feb 18 ?        0:00 /usr/sbin/keyserv
    root   144     1  0   Feb 18 ?       50:37 /usr/lib/autofs/automountd
    root  1010     1  0   Apr 12 ?        0:00 /opt/perf/bin/midaemon
    root   106     1  0   Feb 18 ?        0:02 /usr/lib/netsvc/yp/ypbind -broadt
    root   156     1  0   Feb 18 ?        0:03 /usr/sbin/cron
    root   176     1  0   Feb 18 ?        0:00 /usr/lib/lpsched
    root   129     1  0   Feb 18 ?        0:00 /usr/lib/nfs/lockd
  daemon   130     1  0   Feb 18 ?        0:01 /usr/lib/nfs/statd
    root 14798     1  0   Mar 09 ?       31:10 /usr/sbin/nscd
    root   133     1  0   Feb 18 ?        0:10 /usr/sbin/inetd -s
    root   197     1  0   Feb 18 ?        0:00 /usr/lib/power/powerd
    root   196     1  0   Feb 18 ?        0:35 /etc/opt/licenses/lmgrd.ste -c /d
    root   213     1  0   Feb 18 ?     4903:09 /usr/sbin/vold
    root   199   196  0   Feb 18 ?        0:03 suntechd -T  4 -c /etc/optd
    root   219     1  0   Feb 18 ?        0:08 /usr/lib/sendmail -bd -q15m
    root   209     1  0   Feb 18 ?        0:05 /usr/lib/utmpd
    root  2935   266  0   Feb 22 ?       48:08 /usr/openwin/bin/Xsun :0 -nobanna
    root 16795 16763  1 07:51:34 pts/4    0:00 ps -ef
    root  2963  2954  0   Feb 22 ?        0:17 /usr/openwin/bin/fbconsole
    root   479     1  0   Feb 18 console  0:00 /usr/lib/saf/ttymon -g -h -p sunc
    root 10976     1  0   Jun 01 ?        0:00 /opt/perf/bin/ttd
    root  7468     1  0   Feb 24 ?        0:13 /opt/perf/bin/pvalarmd
    root   266     1  0   Feb 18 ?        0:01 /usr/dt/bin/dtlogin -daemon
  martyp 16763 16761  0 07:46:46 pts/4    0:01 -ksh
    root 10995     1  0   Jun 01 ?        0:01 /opt/perf/bin/perflbd
    root   484   478  0   Feb 18 ?        0:00 /usr/lib/saf/ttymon
    root   458     1  0   Feb 18 ?       20:06 /usr/lib/snmp/snmpdx -y -c /etc/f
    root 16792  3059  0 07:50:37 ?        0:00 /usr/dt/bin/dtscreen -mode blank
    root   471     1  0   Feb 18 ?        0:07 /usr/lib/dmi/dmispd
    root   474     1  0   Feb 18 ?        0:00 /usr/lib/dmi/snmpXdmid -s
    root   485   458  0   Feb 18 ?      739:44 mibiisa -r -p 32874
    root  2954  2936  0   Feb 22 ?        0:01 /bin/ksh /usr/dt/bin/Xsession
    root  2936   266  0   Feb 22 ?        0:00 /usr/dt/bin/dtlogin -daemon
    root  3061  3059  0   Feb 22 ?        1:32 dtwm
```

```
    root  3058     1  0   Feb 22 pts/2    0:01 /usr/dt/bin/ttsession
    root   712   133  0   Feb 18 ?        0:01 rpc.ttdbserverd
    root 11001 11000  0                   0:01 <defunct>
    root  2938     1  0   Feb 22 ?        0:00 /usr/openwin/bin/fbconsole -d :0
    root  2999  2954  0   Feb 22 pts/2    0:16 /usr/dt/bin/sdt_shell -c      unt
    root  3059  3002  0   Feb 22 pts/2   283:35 /usr/dt/bin/dtsession
    root  3063  3059  0   Feb 22 ?        0:03 /usr/dt/bin/dthelpview -helpVolur
    root  3099  3062  0   Feb 22 ?        0:13 /usr/dt/bin/dtfile -geometry +700
    root 11000 10995  0   Jun 01 ?        0:02 /opt/perf/bin/agdbserver -t alar/
    root  3002  2999  0   Feb 22 pts/2    0:01 -ksh -c      unset DT;     DISPLg
    root   730   133  0   Feb 18 ?        1:37 rpc.rstatd
    root  3062  3059  0   Feb 22 ?        2:17 /usr/dt/bin/dtfile -geometry +700
    root  3067     1  0   Feb 22 ?        0:00 /bin/ksh /usr/dt/bin/sdtvolcheckm
    root  3000     1  0   Feb 22 ?        0:00 /usr/dt/bin/dsdm
    root  3078  3067  0   Feb 22 ?        0:00 /bin/cat /tmp/.removable/notify0
    root 10984     1  0   Jun 01 ?       12:42 /opt/perf/dce/bin/dced -b
    root 16761   133  0 07:46:45 ?        0:00 in.telnetd
martyp $
```



AIX example:

```
martyp $ ps -ef
     UID   PID  PPID   C    STIME    TTY  TIME CMD
    root     1     0   0   Feb 24      -  5:07 /etc/init
    root  2208 15520   0   Feb 24      -  8:21 dtwm
    root  2664     1   0   Feb 24      -  0:00 /usr/dt/bin/dtlogin -daemon
    root  2882     1   0   Feb 24      - 158:41 /usr/sbin/syncd 60
    root  3376  2664   5   Feb 24      - 3598:41 /usr/lpp/X11/bin/X -D /usr/lib/
    root  3624  2664   0   Feb 24      -  0:00 dtlogin <:0>        -daemon
    root  3950     1   6   Feb 24      - 5550:30 /usr/lpp/perf/bin/llbd
    root  4144     1   0   Feb 24      -  0:00 /usr/lpp/perf/bin/midaemon
    root  4490     1   0   Feb 24      -  0:48 /usr/lpp/perf/bin/perflbd
    root  4906     1   0   Feb 24      -  0:00 /usr/lib/errdemon
    root  5172     1   0   Feb 24      -  0:00 /usr/sbin/srcmstr
    root  5724  5172   0   Feb 24      -  9:54 /usr/sbin/syslogd
    root  6242  5172   0   Feb 24      -  0:00 /usr/sbin/biod 6
    root  6450  5172   0   Feb 24      -  0:02 sendmail: accepting connections
    root  6710  5172   0   Feb 24      -  7:34 /usr/sbin/portmap
    root  6966  5172   0   Feb 24      -  0:23 /usr/sbin/inetd
    root  7224  5172   0   Feb 24      -  1:09 /usr/sbin/timed -S
    root  7482  5172   0   Feb 24      - 11:55 /usr/sbin/snmpd
    root  8000     1   0   Feb 24      -  9:17 ovspmd
    root  8516  8782   0   Feb 24      -  0:00 netfmt -CF
    root  8782     1   0   Feb 24      -  0:00 /usr/OV/bin/ntl_reader 0 1 1 1
    root  9036  8000   0   Feb 24      - 10:09 ovwdb -O -n5000
    root  9288  8000   0   Feb 24      -  0:44 pmd -Au -At -Mu -Mt -m
    root  9546  8000   0   Feb 24      - 20:05 trapgend -f
    root  9804  8000   0   Feb 24      -  0:28 trapd
    root 10062  8000   0   Feb 24      -  0:47 orsd
    root 10320  8000   0   Feb 24      -  0:33 ovesmd
    root 10578  8000   0   Feb 24      -  0:30 ovelmd
    root 10836  8000   0   Feb 24      - 13:12 ovtopmd -O
    root 11094  8000   0   Feb 24      - 17:50 netmon -P
    root 11352  8000   0   Feb 24      -  0:02 snmpCollect
    root 11954     1   0   Feb 24      -  1:22 /usr/sbin/cron
    root 12140  5172   0   Feb 24      -  0:01 /usr/lib/netsvc/yp/ypbind
    root 12394  5172   0   Feb 24      -  1:39 /usr/sbin/rpc.mountd
    root 12652  5172   0   Feb 24      -  0:29 /usr/sbin/nfsd 8
    root 12908  5172   0   Feb 24      -  0:00 /usr/sbin/rpc.statd
    root 13166  5172   0   Feb 24      -  0:29 /usr/sbin/rpc.lockd
    root 13428     1   0   Feb 24      -  0:00 /usr/sbin/uprintfd
    root 14190  5172   0   Feb 24      - 72:59 /usr/sbin/automountd
    root 14452  5172   0   Feb 24      -  0:17 /usr/sbin/qdaemon
    root 14714  5172   0   Feb 24      -  0:00 /usr/sbin/writesrv
    root 14992     1   0   Feb 24      - 252:26 /usr/lpp/perf/bin/scopeux
    root 15520  3624   1   Feb 24      - 15:29 /usr/dt/bin/dtsession
    root 15742     1   0   Feb 24      -  0:00 /usr/lpp/diagnostics/bin/diagd
    root 15998     1   0   Feb 24   lft0  0:00 /usr/sbin/getty /dev/console
    root 16304 18892   0   Feb 24  pts/0  0:00 /bin/ksh
```

```
   root 16774     1   0   Feb 24      -  0:00 /usr/lpp/perf/bin/ttd
   root 17092  4490   0   Feb 24      - 68:54 /usr/lpp/perf/bin/rep_server -t
   root 17370 19186   3                  0:00 <defunct>
   root 17630 15520   0   Mar 25      -  0:00 /usr/dt/bin/dtexec -open 0 -ttp
   root 17898 15520   0   Mar 20      -  0:00 /usr/dt/bin/dtexec -open 0 -ttp
   root 18118 19888   0   Feb 24  pts/1  0:00 /bin/ksh
   root 18366  6966   0   Feb 24      -  0:00 rpc.ttdbserver 100083 1
   root 18446 15520   0   Mar 15      -  0:00 /usr/dt/bin/dtexec -open 0 -ttp
   root 18892 15520   0   Feb 24      -  3:46 /usr/dt/bin/dtterm
   root 19186 16304   0   Feb 24  pts/0  0:01 /usr/lpp/X11/bin/msmit
   root 19450     1   0   Feb 24      - 26:53 /usr/dt/bin/ttsession -s
   root 19684  2208   0   Feb 24      -  0:00 /usr/dt/bin/dtexec -open 0 -ttp
   root 19888 19684   0   Feb 24      -  0:00 /usr/dt/bin/dtterm
   root 20104 15520   0   Feb 27      -  0:00 /usr/dt/bin/dtexec -open 0 -ttp
   root 20248 20104   0   Feb 27      -  0:03 /usr/dt/bin/dtscreen
   root 20542 29708   0   May 14      -  0:03 /usr/dt/bin/dtscreen
   root 20912 26306   0   Apr 05      -  0:03 /usr/dt/bin/dtscreen
   root 33558     1   0   May 18      -  3:28 /usr/atria/etc/lockmgr -a /var/
   root 33834  6966   3 07:55:49      -  0:00 telnetd
   root 34072     1   0   May 18      -  0:00 /usr/atria/etc/albd_server
 martyp 36296 36608  13 07:56:07  pts/2  0:00 ps -ef
 martyp 36608 33834   1 07:55:50  pts/2  0:00 -ksh
   root 37220 15520   0   May 28      -  0:00 /usr/dt/bin/dtexec -open 0 -ttp
```



HP-UX example (partial listing):

```
martyp $ ps -ef

    UID   PID  PPID  C    STIME    TTY  TIME COMMAND
    root     0     0  0   Mar  9  ?   107:28 swapper
    root     1     0  0   Mar  9  ?    2:27 init
    root     2     0  0   Mar  9  ?   14:13 vhand
    root     3     0  0   Mar  9  ?   114:55 statdaemon
    root     4     0  0   Mar  9  ?    5:57 unhashdaemon
    root     7     0  0   Mar  9  ?   154:33 ttisr
    root    70     0  0   Mar  9  ?    0:01 lvmkd
    root    71     0  0   Mar  9  ?    0:01 lvmkd
    root    72     0  0   Mar  9  ?    0:01 lvmkd
    root    13     0  0   Mar  9  ?    9:54 vx_sched_thread
    root    14     0  0   Mar  9  ?    1:54 vx_iflush_thread
    root    15     0  0   Mar  9  ?    2:06 vx_ifree_thread
    root    16     0  0   Mar  9  ?    2:27 vx_inactive_cache_thread
    root    17     0  0   Mar  9  ?    0:40 vx_delxwri_thread
    root    18     0  0   Mar  9  ?    0:33 vx_logflush_thread
    root    19     0  0   Mar  9  ?    0:07 vx_attrsync_thread
                       .
                       .
                       .

    root    69     0  0   Mar  9  ?    0:09 vx_inactive_thread
    root    73     0  0   Mar  9  ?    0:01 lvmkd
    root    74     0 19   Mar  9  ?   3605:29 netisr
    root    75     0  0   Mar  9  ?    0:18 netisr
    root    76     0  0   Mar  9  ?    0:17 netisr
    root    77     0  0   Mar  9  ?    0:14 netisr
    root    78     0  0   Mar  9  ?    0:48 nvsisr
    root    79     0  0   Mar  9  ?    0:00 supsched
    root    80     0  0   Mar  9  ?    0:00 smpsched
    root    81     0  0   Mar  9  ?    0:00 smpsched
    root    82     0  0   Mar  9  ?    0:00 sblksched
    root    83     0  0   Mar  9  ?    0:00 sblksched
    root    84     0  0   Mar  9  ?    0:00 strmem
    root    85     0  0   Mar  9  ?    0:00 strweld
    root  3730     1  0 16:39:22 console   0:00 /usr/sbin/getty console console
    root   404     1  0   Mar  9  ?    3:57 /usr/sbin/swagentd
  oracle   919     1  0 15:23:23 ?    0:00 oraclegprd (LOCAL=NO)
    root   289     1  2   Mar  9  ?    78:34 /usr/sbin/syncer
    root   426     1  0   Mar  9  ?    0:10 /usr/sbin/syslogd -D
    root   576     1  0   Mar  9  ?    0:00 /usr/sbin/portmap
```

```
    root   429     1  0  Mar  9  ?    0:00 /usr/sbin/ptydaemon
    root   590     1  0  Mar  9  ?    0:00 /usr/sbin/biod 4
    root   442     1  0  Mar  9  ?    0:00 /usr/lbin/nktl_daemon 0 0 0 0 0 1 -2
  oracle  8145     1  0 12:02:48 ?    0:00 oraclegprd (LOCAL=NO)
    root   591     1  0  Mar  9  ?    0:00 /usr/sbin/biod 4
    root   589     1  0  Mar  9  ?    0:00 /usr/sbin/biod 4
    root   592     1  0  Mar  9  ?    0:00 /usr/sbin/biod 4
    root   604     1  0  Mar  9  ?    0:00 /usr/sbin/rpc.lockd
    root   598     1  0  Mar  9  ?    0:00 /usr/sbin/rpc.statd
    root   610     1  0  Mar  9  ?    0:16 /usr/sbin/automount -f /etc/auto_master
    root   638     1  0  Mar  9  ?    0:06 sendmail: accepting connections
    root   618     1  0  Mar  9  ?    0:02 /usr/sbin/inetd
    root   645     1  0  Mar  9  ?    5:01 /usr/sbin/snmpdm
    root   661     1  0  Mar  9  ?   11:28 /usr/sbin/fddisubagtd
    root   711     1  0  Mar  9  ?   30:59 /opt/dce/sbin/rpcd
    root   720     1  0  Mar  9  ?    0:00 /usr/sbin/vtdaemon
    root   867   777  1  Mar  9  ?    0:00 <defunct>
      lp   733     1  0  Mar  9  ?    0:00 /usr/sbin/lpsched
    root   777     1  0  Mar  9  ?    8:55 DIAGMON
    root   742     1  0  Mar  9  ?    0:15 /usr/sbin/cron
  oracle  7880     1  0 11:43:47 ?    0:00 oraclegprd (LOCAL=NO)
    root   842     1  0  Mar  9  ?    0:00 /usr/vue/bin/vuelogin
  oracle  5625     1  0 07:00:14 ?    0:01 ora_smon_gprd
    root   781     1  0  Mar  9  ?    0:00 /usr/sbin/envd
    root   833   777  0  Mar  9  ?    0:00 DEMLOG  DEMLOG;DEMLOG;0;0;
    root   813     1  0  Mar  9  ?    0:00 /usr/sbin/nfsd 4
    root   807     1  0  Mar  9  ?    0:00 /usr/sbin/rpc.mountd
    root   815   813  0  Mar  9  ?    0:00 /usr/sbin/nfsd 4
    root   817   813  0  Mar  9  ?    0:00 /usr/sbin/nfsd 4
    root   835   777  0  Mar  9  ?    0:13 PSMON  PSMON;PSMON;0;0;
```

Here is a brief description of the headings:

UID	The user ID of the process owner.
PID	The process ID (you can use this number to kill the process).
PPID	The process ID of the parent process.
C	Processor utilization. On a multi-processor system, you may see this number go beyond 100%. It could potentially go to 100% per processor, so a two-processor system may show 200% utilization. This varies among UNIX variants.
STIME	The start time of the process.
TTY	The controlling terminal for the process.
TIME	The cumulative execution time for the process.
COMMAND	The command name and arguments.



ps gives a quick profile of the processes running on your system. To get more detailed information, you can include the "/" option, which includes a lot of useful additional information, as shown in the following example:

```
martyp $ ps -efl
 F S      UID   PID  PPID  C PRI NI     ADDR     SZ    WCHAN    STIME TTY      D
19 T     root     0     0  0   0 SY f026f7f0      0             Feb 18 ?        d
 8 S     root     1     0  0  41 20 f5b90808    175 f5b90a30   Feb 18 ?        -
19 S     root     2     0  0   0 SY f5b90108      0 f0283fd0   Feb 18 ?        t
19 S     root     3     0  0   0 SY f5b8fa08      0 f0287a44   Feb 18 ?       6h
 8 S     root  3065  3059  0  40 20 f626d040   1639 f62aab96   Feb 22 ?        c
 8 S     root    88     1  0  40 20 f5b8d708    377 f5b59df6   Feb 18 ?        q
 8 S     root   478     1  0  41 20 f5b8ec08    388 f5b51bb8   Feb 18 ?        0
 8 S     root    94     1  0  41 20 f5b8d008    527 f5b59e46   Feb 18 ?        d
 8 S     root   150     1  0  41 20 f5da1a10    808 f5b59806   Feb 18 ?        d
 8 S     root    96     1  0  67 20 f5da2810    535 f5b59ad6   Feb 18 ?        v
 8 S     root   144     1  0  41 20 f5da0c10   2694 ef69f61c   Feb 18 ?       5d
 8 S     root  1010     1  0   0 RT f61da330    496 f5dbec1c   Apr 12 ?        n
```

```
 8 S     root   106     1  0  41 20 f5da1310    485 f5b59e96   Feb 18 ?        s
 8 S     root   156     1  0  51 20 f5b8de08    446 f5b51eb8   Feb 18 ?        n
 8 S     root   176     1  0  53 20 f5da2110    740 f5b59036   Feb 18 ?        d
 8 S     root   129     1  0  56 20 f5d9fe10    447 f5b59cb6   Feb 18 ?        d
 8 S   daemon   130     1  0  41 20 f5d9f710    564 f5b59b76   Feb 18 ?        d
 8 S     root 14798     1  0  45 20 f5b8e508    616 f5b8e730   Mar 09 ?       3d
 8 S     root   133     1  0  51 20 f5e18818    507 f5b59c66   Feb 18 ?        s
 8 S     root   197     1  0  63 20 f5e15e18    284 f5e16040   Feb 18 ?        d
 8 S     root   196     1  0  41 20 f5da0510    429 f5c68f8e   Feb 18 ?        c
 8 S     root   213     1  0  41 20 f5e16518    586 f5c68b2e   Feb 18 ?       4d
 8 S     root   199   196  0  41 20 f5e16c18    451 f5b59f86   Feb 18 ?        i
 8 S     root   219     1  0  41 20 f5e17318    658 f5b59d06   Feb 18 ?        m
 8 S     root   209     1  0  41 20 f5e18118    234 f5c68e4e   Feb 18 ?        d
 8 S     root  2935   266  0  40 20 f61db130   2473 f62aaa56   Feb 22 ?       4
 8 S     root 16800  3059  1  81 30 f626f340   1466 f61b345e 07:59:40 ?        k
 8 S     root  2963  2954  0  40 20 f5f52028    513 f61b313e   Feb 22 ?        e
 8 S     root   479     1  0  55 20 f5ee7120    407 f5fde2c6   Feb 18 console  g
 8 S     root 10976     1  0  65 20 f5f55828    478 f5c6853e   Jun 01 ?        d
 8 S     root  7468     1  0  46 20 f621da38   2851    8306c   Feb 24 ?        d
 8 S     root   266     1  0  41 20 f5ee5520   1601 f5c6858e   Feb 18 ?        n
 8 S   martyp 16763 16761  0  51 20 f6270140    429 f62701ac 07:46:46 pts/4    h
 8 S     root 10995     1  0  41 20 f5b8f308   2350 f5fde5e6   Jun 01 ?        d
 8 S     root   484   478  0  41 20 f5ee4e20    408 f5ee5048   Feb 18 ?        n
 8 S     root   458     1  0  41 20 f5f54a28    504 f5fde906   Feb 18 ?       2m
 8 O     root 16802 16763  1  61 20 f5ee7820    220          08:00:05 pts/4    l
 8 S     root   471     1  0  41 20 f5f53c28    658 f5fde726   Feb 18 ?        d
 8 S     root   474     1  0  51 20 f5f53528    804 f61a58b6   Feb 18 ?        g
 8 S     root   485   458  0  40 20 f5f52e28    734 f607ecde   Feb 18 ?       74
 8 S     root  2954  2936  0  40 20 f626e540    433 f626e5ac   Feb 22 ?        n
 8 S     root  2936   266  0  66 20 f5ee4720   1637 f5ee478c   Feb 22 ?        n
 8 S     root  3061  3059  0  40 20 f5e17a18   2041 f61b359e   Feb 22 ?        m
 8 S     root  3058     1  0  40 20 f61daa30   1067 f62aadc6   Feb 22 pts/2    n
 8 S     root   712   133  0  41 20 f61d8e30    798 f61b390e   Feb 18 ?        d
 8 Z     root 11001 11000  0   0                                              >
 8 S     root  2938     1  0  60 20 f5ee6320    513 f601bfb6   Feb 22 ?        0
 8 S     root  2999  2954  0  40 20 f621e138   1450 f61b33be   Feb 22 pts/2    t
 8 S     root  3059  3002  1  51 20 f626de40   4010 f62aafa6   Feb 22 pts/2   2n
 8 S     root  3063  3059  0  50 20 f621e838   1952 f62aa556   Feb 22 ?
 8 S     root  3099  3062  0  40 20 f5f52728   2275 f60a1d18   Feb 22 ?        0
 8 S     root 11000 10995  0  48 20 f626d740   2312    55694   Jun 01 ?        e
 8 S     root  3002  2999  0  43 20 f61d8730    427 f61d879c   Feb 22 pts/2    =
 8 S     root   730   133  0  40 20 f61d9530    422 f62aa9b6   Feb 18 ?        d
 8 S     root  3062  3059  0  61 20 f621b738   2275 f62aa506   Feb 22 ?        0
 8 S     root  3067     1  0  40 20 f5ee5c20    424 f5ee5c8c   Feb 22 ?        d
 8 S     root  3000     1  0  40 20 f61d8030    518 f62aa8c6   Feb 22 ?        m
 8 S     root  3078  3067  0  40 20 f61d9c30    211 f5b512b8   Feb 22 ?        0
 8 S     root 10984     1  0  41 20 f5f54328   2484 eee46e84   Jun 01 ?       1b
 8 S     root 16761   133  0  44 20 f5ee4020    411 f5c6894e 07:46:45 ?        d
martyp $
```

In this example, the first column is *F* for flags. *F* provides octal information about whether the process is swapped, in core, a system process, and so on. The octal value sometimes varies from system to system, so check the manual pages for your system to see the octal value of the flags.

S is for state. The state can be sleeping, as indicated by *S* for most of the processes shown in the example, waiting, running, intermediate, terminated, and so on. Again, some of these values may vary from system to system, so check your manual pages.

Some additional useful information in this output are: *NI* for the nice value, *ADDR* for the memory address of the process, *SZ* for the size of the process in physical pages, and *WCHAN*, which is the event for which the process is waiting.

[Team LiB]

◀ PREVIOUS NEXT ▶

Killing a Process



If you issue the **ps** command and find that one of your processes is hung, or if you started a large job that you wish to stop, you can do so with the **kill** command. **kill** is a utility that sends a signal to the process you identify. You can **kill** any process that you own. In addition, the superuser can kill almost any process on the system.

To kill a process that you own, simply issue the **kill** command and the Process ID (PID). The following example shows issuing the **ps** command to find all processes owned by *martyp*, killing a process, and checking to see that it has disappeared:




```
martyp $ ps -ef | grep martyp
  martyp 19336 19334 0 05:24:32 pts/4  0:01 -ksh
  martyp 19426 19336 0 06:01:01 pts/4  0:00 grep martyp
  martyp 19424 19336 5 06:00:48 pts/4  0:01 find / -name .login
martyp $ kill 19424
martyp $ ps -ef | grep martyp
  martyp 19336 19334 0 05:24:32 pts/4  0:01 -ksh
  martyp 19428 19336 1 06:01:17 pts/4  0:00 grep martyp
[1] + Terminated              find / -name .login &
martyp $
```

The example shows killing process *19424,* which is owned by *martyp*. We confirm that the process has indeed been killed by reissuing the **ps** command. You can also use the *-u* option to **ps** to list processes with the login name you specify.

You can kill several processes on the command line by issuing **kill** followed by a space-separated list of all the process numbers you wish to kill.

Take special care when killing processes if you are logged in as superuser. You may adversely affect the way the system runs and have to manually restart processes or reboot the system.

Signals



When you issue the **kill** command and process number, you are also sending a *signal* associated with the **kill**. We did not specify a *signal* in our **kill** example; however, the default *signal* of 15, or *SIGTERM*, was used. These *signals* are used by the system to communicate with processes. The *signal* of 15 we used to terminate our process is a software termination *signal* that is usually enough to terminate a user process such as the **find** we had started. A process that is difficult to kill may require the *SIGKILL*, or 9 *signal*. This *signal* causes an immediate termination of the process. I use this only as a last resort because processes killed with *SIGKILL* do not always terminate smoothly. To kill such processes as the shell, you sometimes have to use *SIGKILL*.

You can use either the *signal* name or number. These signal numbers sometimes vary from system to system, so view the manual page for *signal*, usually in section 5, to see the list of *signals* on your system. A list of some of the most frequently used *signal* numbers and corresponding *signals* follows:

Signal Number	Signal
1	SIGHUP
2	SIGINT
3	SIGQUIT
9	SIGKILL
15	SIGTERM
24	SIGSTOP

To kill a process with id *234* with *SIGKILL*, you would issue the following command:

```
$ kill -9 234
    |    |  |
    |    |  |> process id (PID)
    |    |> signal number
    |> kill command to terminate the process
```

Show Remote Mounts with showmount

showmount is used to show all remote systems (clients) that have mounted a local file system. **showmount** is useful for determining the file systems that are most often mounted by clients with NFS. The output of **showmount** is particularly easy to read because it lists the host name and directory that was mounted by the client.

NFS servers often end up serving many NFS clients that were not originally intended to be served. This situation ends up consuming additional UNIX system resources on the NFS server, as well as additional network bandwidth. Keep in mind that any data transferred from an NFS server to an NFS client consumes network bandwidth, and in some cases, may be a substantial amount of bandwith if large files or applications are being transferred from the NFS server to the client. The following example is a partial output of **showmount** taken from a system. **showmount** runs on the HP-UX, AIX, and Linux systems I have been using throughout this chapter, but not on the Solaris system:

```
# showmount -a

sys100.ct.mp.com:/applic

sys101.ct.mp.com:/applic
```

sys102.cal.mp.com:/applic

sys103.cal.mp.com:/applic

sys104.cal.mp.com:/applic

sys105.cal.mp.com:/applic

sys106.cal.mp.com:/applic

sys107.cal.mp.com:/applic

sys108.cal.mp.com:/applic

sys109.cal.mp.com:/applic

sys200.cal.mp.com:/usr/users

sys201.cal.mp.com:/usr/users

sys202.cal.mp.com:/usr/users

sys203.cal.mp.com:/usr/users

sys204.cal.mp.com:/usr/users

sys205.cal.mp.com:/usr/users

sys206.cal.mp.com:/usr/users

sys207.cal.mp.com:/usr/users

sys208.cal.mp.com:/usr/users

sys209.cal.mp.com:/usr/users

The three following options are available for the **showmount** command:

-a prints output in the format "name:directory," as shown above.

-d lists all the local directories that have been remotely mounted by clients.

-e prints a list of exported file systems.

The following are examples of **showmount -d** and **showmount -e**:

showmount -d

/applic

/usr/users

/usr/oracle

/usr/users/emp.data

/network/database

/network/users

/tmp/working

showmount -e

export list for server101.cal.mp.com

/applic

/usr/users

/cdrom

[Team LiB]

Show System Swap

If your system has insufficient main memory for all the information it needs to work with, it will move pages of information to your swap area or swap entire processes to your swap area. Pages that were most recently used are kept in main memory, and those not recently used will be the first moved out of main memory.

System administrators spend a lot of time determining the right amount of swap space for their systems. Insufficient swap may prevent a system from starting additional processes, hang applications, or not permit additional users to get access to the system. Having sufficient swap prevents these problems from occurring. System administrators usually go about determining the right amount of swap by considering many important factors, including the following:

1. How much swap is recommended by the application(s) you run? Use the swap size recommended by your applications. Application vendors tend to be realistic when recommending swap space. There is sometimes competition among application vendors to claim the lowest memory and CPU requirements in order to keep the overall cost of solutions as low as possible, but swap space recommendations are usually realistic.

2. How many applications will you run simultaneously? If you are running several applications, sum the swap space recommended for each application you plan to run simultaneously. If you have a database application that recommends 200 MBytes of swap and a development tool that recommends 100 MBytes of swap, then configure your system with 300 MBytes of swap, minimum.

3. Will you be using substantial system resources on peripheral functionality such as NFS? The nature of NFS is to provide access to file systems, some of which may be very large, so this use may have an impact on your swap space requirements.

Swap is listed and manipulated on different UNIX variants with different commands. The following example shows listing the swap area on a Solaris system with **swap -l**:

```
# swap -l
swapfile             dev  swaplo blocks   free
/dev/dsk/c0t3d0s1   32,25      8 263080 209504
```

These values are all in 512 KByte blocks. In this case, the free blocks are *209504*, which is a significant amount of the overall swap allocated on the system.

You can view the amount of swap being consumed on your HP-UX system with **swapinfo**. The following is an example output of **swapinfo**:

```
# swapinfo

             Kb      Kb      Kb   PCT  START/      Kb
TYPE      AVAIL    USED    FREE  USED   LIMIT RESERVE  PRI  NAME
dev       49152   10532   38620   21%       0       -    1  /dev/vg00/lvol2
dev      868352   10888  759160    1%       0       -    1  /dev/vg00/lvol8
reserve       -  532360 -532360
memory   816360  469784  346576   58%
```

Following is a brief overview of what **swapinfo** gives you.

In the previous example, the "TYPE" field indicated whether the swap was "dev" for device, "reserve" for paging space on reserve, or "memory." Memory is a way to allow programs to reserve more virtual memory than you have hard disk paging space setup for on your system.

"Kb AVAIL" is the total swap space available in 1024-byte blocks. This includes both used and unused swap.

"Kb USED" is the current number of 1024-byte blocks in use.

"Kb FREE" is the difference between "Kb AVAIL" and "Kb USED."

"PCT USED" is "Kb USED" divided by "Kb AVAIL."

"START/LIMIT" is the block address of the start of the swap area.

"Kb RESERVE" is "-" for device swap or the number of 1024-byte blocks for file system swap.

"PRI" is the priority given to this swap area.

"NAME" is the device name of the swap device.

You can also issue the **swapinfo** command with a series of options. Here are some of the options you can include:

-m	to display output of **swapinfo** in MBytes rather than in 1024-byte blocks.
-d	prints information related to device swap areas only.
-f	prints information about file system swap areas only.

[Team LiB]

PREVIOUS NEXT

sar: The System Activity Reporter

sar is another UNIX command for gathering information about activities on your system. You can gather data over an extended time period with **sar** and later produce reports based on the data. **sar** is similar among UNIX variants in that the options and outputs are similar. The Linux system I was using for the examples did not support **sar**, but the Solaris, HP-UX, and AIX systems had the same options and nearly identical outputs. The following are some useful options to **sar,** along with examples of reports produced with these options where applicable:

sar -o Saves data in a file specified by "o." After the file name, you would usually also enter the time interval for samples and the number of samples. The following example shows saving the binary data in file /tmp/sar.data at an interval of 60 seconds 300 times:

```
# sar -o /tmp/sar.data 60 300
```

The data in **/tmp/sar.data** can later be extracted from the file.

sar -f Specify a file from which you will extract data.





Report CPU utilization with the headings %usr, %sys, %wio, %idle with some processes waiting for block I/O, %idle. This report is similar to the **iostat** and **vmstat** CPU reports. You extract the binary data saved in a file to get CPU information, as shown in the following example. The following is a **sar -u** example:

sar -u

sar -u -f /tmp/sar.data

```
Header Information for your system

12:52:04    %usr    %sys    %wio   %idle
12:53:04      62       4       5      29
12:54:04      88       5       3       4
12:55:04      94       5       1       0
12:56:04      67       4       4      25
12:57:04      59       4       4      32
12:58:04      61       4       3      32
12:59:04      65       4       3      28
13:00:04      62       5      16      17
13:01:04      59       5       9      27
13:02:04      71       4       3      22
13:03:04      60       4       4      32
13:04:04      71       5       4      20
13:05:04      80       6       8       7
13:06:04      56       3       3      37
13:07:04      57       4       4      36
13:08:04      66       4       4      26
13:09:04      80      10       2       8
13:10:04      73      10       2      15
13:11:04      64       6       3      28
13:12:04      56       4       3      38
13:13:04      55       3       3      38
13:14:04      57       4       3      36
13:15:04      70       4       5      21
13:16:04      65       5       9      21
13:17:04      62       6       2      30
13:18:04      60       5       3      33
```

```
13:19:04       77         3         4        16
13:20:04       76         5         3        15
                    .
                    .
                    .
14:30:04       50         6         6        38
14:31:04       57        12        19        12
14:32:04       51         8        20        21
14:33:04       41         4         9        46
14:34:04       43         4         9        45
14:35:04       38         4         6        53
14:36:04       38         9         7        46
14:37:04       46         3        11        40
14:38:04       43         4         7        46
14:39:04       37         4         5        54
14:40:04       33         4         5        58
14:41:04       40         3         3        53
14:42:04       44         3         3        50
14:43:04       27         3         7        64

Average        57         5         8        30
```

sar -b Report buffer cache activity. A database application such as Oracle would recommend that you use this option to see the effectiveness of buffer cache use. You extract the binary data saved in a file to get CPU information, as shown in the following example:

```
# sar -b -f /tmp/sar.data

Header information for your system

12:52:04 bread/s lread/s %rcache bwrit/s lwrit/s %wcache pread/s pwrit/s
12:53:04       5     608      99        1       11       95        0        0
12:54:04       7     759      99        0       14       99        0        0
12:55:04       2    1733     100        4       24       83        0        0
12:56:04       1     836     100        1       18       96        0        0
12:57:04       0     623     100        2       21       92        0        0
12:58:04       0     779     100        1       16       96        0        0
12:59:04       0    1125     100        0       14       98        0        0
13:00:04       2    1144     100        9       89       89        0        0
13:01:04      10     898      99       11       76       86        0        0
13:02:04       0    1156     100        0       14       99        0        0
13:03:04       1     578     100        2       22       88        0        0
13:04:04       5    1251     100        0       12       99        0        0
13:05:04       3    1250     100        0       12       97        0        0
13:06:04       1     588     100        0       12       98        0        0
13:07:04       1     649     100        2       15       86        0        0
13:08:04       1     704     100        2       15       86        0        0
13:09:04       1    1068     100        0       18      100        0        0
13:10:04       0     737     100        1       44       99        0        0
13:11:04       0     735     100        1       13       95        0        0
13:12:04       0     589     100        1       15       93        0        0
13:13:04       0     573     100        0       16       99        0        0
13:14:04       1     756     100        1       16       91        0        0
13:15:04       1    1092     100        9       49       81        0        0
13:16:04       2     808     100        6       82       93        0        0
13:17:04       0     712     100        1        9       93        0        0
13:18:04       1     609     100        0       13       97        0        0
13:19:04       1     603     100        0       10       99        0        0
13:20:04       0    1127     100        0       14       98        0        0
                         .
                         .
                         .
14:30:04       2     542     100        1       22       94        0        0
14:31:04      10     852      99       12      137       92        0        0
14:32:04       2     730     100       10      190       95        0        0
14:33:04       4     568      99        2       26       91        0        0
14:34:04       4     603      99        1       13       91        0        0
14:35:04       1     458     100        1       13       89        0        0
14:36:04      13     640      98        1       24       98        0        0
14:37:04      21     882      98        1       18       95        0        0
14:38:04       7     954      99        0       19       98        0        0
14:39:04       3     620     100        1       11       94        0        0
14:40:04       3     480      99        2       15       85        0        0
```

```
14:41:04       1     507     100       0       9      98       0       0
14:42:04       1    1010     100       1      10      91       0       0
14:43:04       5     547      99       1       9      93       0       0

Average        3     782     100       3      37      91       0       0
```

sar -d Report disk activity. You get the device name, percent that the device was busy, average number of requests outstanding for the device, number of data transfers per second for the device, and other information. You extract the binary data saved in a file to get CPU information, as shown in the following example:

```
# sar -d -f /tmp/sar.data

Header information for your system

12:52:04   device  %busy   avque   r+w/s  blks/s  avwait  avserv
12:53:04   c0t6d0    0.95    1.41       1      10   16.76   17.28
           c5t4d0  100.00    1.03      20     320    8.36   18.90
           c4t5d1   10.77    0.50      13     214    5.02   18.44
           c5t4d2    0.38    0.50       0       3    4.61   18.81
12:54:04   c0t6d0    0.97    1.08       1      11   10.75   14.82
           c5t4d0  100.00    1.28      54     862    9.31   20.06
           c4t5d1   12.43    0.50      15     241    5.21   16.97
           c5t4d2    0.37    0.50       0       3    3.91   18.20
12:55:04   c0t6d0    1.77    1.42       1      22   13.32   14.16
           c5t4d0  100.00    0.79      26     421    8.33   16.00
           c4t5d1   14.47    0.51      17     270    5.30   13.48
           c5t4d2    0.72    0.50       0       7    4.82   15.69
12:56:04   c0t6d0    1.07   21.57       1      22   72.94   19.58
           c5t4d0  100.00    0.60      16     251    6.80   13.45
           c4t5d1    8.75    0.50      11     177    5.05   10.61
           c5t4d2    0.62    0.50       0       6    4.79   15.43
12:57:04   c0t6d0    0.78    1.16       1       9   13.53   14.91
           c5t4d0  100.00    0.66      15     237    7.60   13.69
           c4t5d1    9.48    0.54      13     210    5.39   13.33
           c5t4d2    0.87    0.50       1      10    4.86   14.09
12:58:04   c0t6d0    1.12    8.29       1      17   54.96   14.35
           c5t4d0  100.00    0.60      11     176    7.91   14.65
           c4t5d1    5.35    0.50       7     111    5.23   10.35
           c5t4d2    0.92    0.50       1      10    4.63   16.08
12:59:04   c0t6d0    0.67    1.53       1       8   18.03   16.05
           c5t4d0   99.98    0.54      11     174    7.69   14.09
           c4t5d1    3.97    0.50       5      83    4.82    9.54
           c5t4d2    1.05    0.50       1      11    4.69   16.29
13:00:04   c0t6d0    3.22    0.67       3      39    8.49   16.53
           c5t4d0  100.00    0.60      65    1032    8.46   14.83
           c4t5d1   21.62    0.50      31     504    5.30    8.94
           c5t4d2    6.77    0.50       5      78    4.86   14.09
13:01:04   c0t6d0    4.45    3.08       5      59   25.83   11.49
           c5t4d0  100.00    0.65      42     676    7.85   14.52
           c4t5d1   21.34    0.55      30     476    5.87   18.49
           c5t4d2    4.37    0.50       3      51    5.32   13.50
                     .
                     .
                     .

14:42:04   c0t6d0    0.53    0.83       0       7   12.21   16.33
           c5t4d0  100.00    0.56       7     107    6.99   14.65
           c4t5d1    6.38    0.50       7     113    4.97   15.18
           c5t4d2    0.15    0.50       0       2    4.53   16.50
14:43:04   c0t6d0    0.52    0.92       0       7   11.50   15.86
           c5t4d0   99.98    0.92      17     270    8.28   18.64
           c4t5d1   10.26    0.50       9     150    5.35   16.41
           c5t4d2    0.12    0.50       0       1    5.25   14.45

Average    c0t6d0    1.43  108.80       2      26    0.00   14.71
Average    c5t4d0  100.00    0.74      25     398    7.83  -10.31
Average    c4t5d1   19.11    0.51      25     399    5.26  -13.75
Average    c5t4d2    1.71    0.53       1      21    5.29   13.46
```

sar Report average queue length. You may have a problem any time the run queue length is greater

-q than the number of processors on the system:

```
# sar -q -f /tmp/sar.data

Header information for your system

12:52:04 runq-sz %runocc swpq-sz %swpocc
12:53:04     1.1      20     0.0        0
12:54:04     1.4      51     0.0        0
12:55:04     1.3      71     0.0        0
12:56:04     1.1      22     0.0        0
12:57:04     1.3      16     0.0        0
12:58:04     1.1      14     0.0        0
12:59:04     1.2      12     0.0        0
13:00:04     1.2      21     0.0        0
13:01:04     1.1      18     0.0        0
13:02:04     1.3      20     0.0        0
13:03:04     1.2      15     0.0        0
13:04:04     1.2      20     0.0        0
13:05:04     1.2      43     0.0        0
13:06:04     1.1      14     0.0        0
13:07:04     1.2      15     0.0        0
13:08:04     1.2      26     0.0        0
13:09:04     1.5      38     0.0        0
13:10:04     1.5      30     0.0        0
13:11:04     1.2      23     0.0        0
13:12:04     1.3      11     0.0        0
13:13:04     1.3      12     0.0        0
13:14:04     1.4      16     0.0        0
13:15:04     1.4      27     0.0        0
13:16:04     1.5      20     0.0        0
13:17:04     1.3      21     0.0        0
13:18:04     1.1      15     0.0        0
13:19:04     1.2      19     0.0        0
13:20:04     1.4      22     0.0        0
                  .
                  .
                  .

14:30:04     1.5       5     0.0        0
14:31:04     1.6      12     0.0        0
14:32:04     1.4       9     0.0        0
14:33:04     1.1       6     0.0        0
14:34:04     1.3       3     0.0        0
14:35:04     1.1       4     0.0        0
14:36:04     1.2       6     0.0        0
14:37:04     1.4       5     0.0        0
14:38:04     1.2      10     0.0        0
14:39:04     1.3       4     0.0        0
14:40:04     1.1       3     0.0        0
14:41:04     1.6       3     0.0        0
14:42:04     1.1       4     0.0        0
14:43:04     1.3       1     0.0        0

Average      1.3      17     1.2        0
```

sar -w Report system swapping activity.

```
# sar -w -f /tmp/sar.data

Header information for your system

12:52:04 swpin/s bswin/s swpot/s bswot/s pswch/s
12:53:04    1.00     0.0    1.00     0.0     231
12:54:04    1.00     0.0    1.00     0.0     354
12:55:04    1.00     0.0    1.00     0.0     348
12:56:04    1.00     0.0    1.00     0.0     200
12:57:04    1.00     0.0    1.00     0.0     277
12:58:04    1.00     0.0    1.00     0.0     235
12:59:04    1.02     0.0    1.02     0.0     199
13:00:04    0.78     0.0    0.78     0.0     456
13:01:04    1.00     0.0    1.00     0.0     435
```

```
13:02:04    1.02     0.0    1.02     0.0     216
13:03:04    0.98     0.0    0.98     0.0     204
13:04:04    1.02     0.0    1.02     0.0     239
13:05:04    1.00     0.0    1.00     0.0     248
13:06:04    0.97     0.0    0.97     0.0     170
13:07:04    1.00     0.0    1.00     0.0     166
13:08:04    1.02     0.0    1.02     0.0     209
13:09:04    0.98     0.0    0.98     0.0     377
13:10:04    1.00     0.0    1.00     0.0     200
13:11:04    1.00     0.0    1.00     0.0     192
13:12:04    0.87     0.0    0.87     0.0     187
13:13:04    0.93     0.0    0.93     0.0     172
13:14:04    1.00     0.0    1.00     0.0     170
13:15:04    1.00     0.0    1.00     0.0     382
13:16:04    1.00     0.0    1.00     0.0     513
13:17:04    1.00     0.0    1.00     0.0     332
13:18:04    1.00     0.0    1.00     0.0     265
13:19:04    1.02     0.0    1.02     0.0     184
13:20:04    0.98     0.0    0.98     0.0     212
                  .
                  .
                  .

14:30:04    0.00     0.0    0.00     0.0     301
14:31:04    0.00     0.0    0.00     0.0     566
14:32:04    0.00     0.0    0.00     0.0     539
14:33:04    0.00     0.0    0.00     0.0     400
14:34:04    0.00     0.0    0.00     0.0     242
14:35:04    0.00     0.0    0.00     0.0     286
14:36:04    0.00     0.0    0.00     0.0     295
14:37:04    0.00     0.0    0.00     0.0     249
14:38:04    0.00     0.0    0.00     0.0     300
14:39:04    0.00     0.0    0.00     0.0     296
14:40:04    0.00     0.0    0.00     0.0     419
14:41:04    0.00     0.0    0.00     0.0     234
14:42:04    0.00     0.0    0.00     0.0     237
14:43:04    0.00     0.0    0.00     0.0     208

Average     0.70     0.0    0.70     0.0     346
```

[Team LiB]

PREVIOUS NEXT

timex to Analyze a Command

If you have a specific command you want to find out more about, you can use **timex**, which reports the elapsed time, user time, and system time spent in the execution of any command you specify.

timex is a good command for users because it gives you an idea of the system resources you are consuming when issuing a command. The following two examples show issuing **timex** with no options to get a short output of the amount of *cpu* consumed; the second example shows issuing **timex -s** to report "total" system activity on a Solaris system:

```
martyp $ timex listing

real        0.02
user        0.00
sys         0.02


martyp $ timex -s listing

real        0.02
user        0.00
sys         0.01


SunOS 5.7 Generic sun4m    08/21

07:48:30    %usr    %sys    %wio   %idle
07:48:31      32      68       0       0

07:48:30 bread/s lread/s %rcache bwrit/s lwrit/s %wcache pread/s pwrit/s
07:48:31       0       0     100       0       0     100       0       0

Average        0       0     100       0       0     100       0       0

07:48:30   device        %busy   avque   r+w/s  blks/s  avwait  avserv

07:48:31   fd0               0     0.0       0       0     0.0     0.0
           nfs1              0     0.0       0       0     0.0     0.0
           nfs219            0     0.0       0       0     0.0     0.0
           sd1               0     0.0       0       0     0.0     0.0
           sd1,a             0     0.0       0       0     0.0     0.0
           sd1,b             0     0.0       0       0     0.0     0.0
           sd1,c             0     0.0       0       0     0.0     0.0
           sd1,g             0     0.0       0       0     0.0     0.0
           sd3               0     0.0       0       0     0.0     0.0
           sd3,a             0     0.0       0       0     0.0     0.0
           sd3,b             0     0.0       0       0     0.0     0.0
           sd3,c             0     0.0       0       0     0.0     0.0
           sd6               0     0.0       0       0     0.0     0.0


Average    fd0               0     0.0       0       0     0.0     0.0
           nfs1              0     0.0       0       0     0.0     0.0
           nfs219            0     0.0       0       0     0.0     0.0
           sd1               0     0.0       0       0     0.0     0.0
           sd1,a             0     0.0       0       0     0.0     0.0
           sd1,b             0     0.0       0       0     0.0     0.0
           sd1,c             0     0.0       0       0     0.0     0.0
           sd1,g             0     0.0       0       0     0.0     0.0
           sd3               0     0.0       0       0     0.0     0.0
           sd3,a             0     0.0       0       0     0.0     0.0
           sd3,b             0     0.0       0       0     0.0     0.0
           sd3,c             0     0.0       0       0     0.0     0.0
           sd6               0     0.0       0       0     0.0     0.0

07:48:30 rawch/s canch/s outch/s rcvin/s xmtin/s mdmin/s
07:48:31       0       0     147       0       0       0

Average        0       0     147       0       0       0
```

```
07:48:30 scall/s sread/s swrit/s  fork/s  exec/s rchar/s wchar/s
07:48:31    2637       0      95   15.79   15.79       0   19216

Average     2637       0      95   15.79   15.79       0   19216

07:48:30 swpin/s bswin/s swpot/s bswot/s pswch/s
07:48:31    0.00     0.0    0.00     0.0     116

Average     0.00     0.0    0.00     0.0     116

07:48:30  iget/s namei/s dirbk/s
07:48:31       0     195     121

Average        0     195     121

07:48:30 runq-sz %runocc swpq-sz %swpocc
07:48:31     2.0     526

Average      2.0     526

07:48:30  proc-sz    ov  inod-sz    ov  file-sz    ov   lock-sz
07:48:31   45/986     0  973/4508    0  357/357     0    0/0


07:48:30   msg/s  sema/s
07:48:31    0.00    0.00

Average     0.00    0.00

07:48:30  atch/s  pgin/s ppgin/s  pflt/s  vflt/s slock/s
07:48:31    0.00    0.00    0.00  505.26 1036.84    0.00

Average     0.00    0.00    0.00  505.26 1036.84    0.00

07:48:30  pgout/s ppgout/s pgfree/s pgscan/s %ufs_ipf
07:48:31     0.00     0.00     0.00     0.00     0.00

Average      0.00     0.00     0.00     0.00     0.00

07:48:30 freemem freeswap
07:48:31   15084  1224421

Average    15084  1224421

07:48:30 sml_mem   alloc  fail  lg_mem   alloc  fail  ovsz_alloc  fail
07:48:31 2617344 1874368     0 17190912 10945416     0     3067904     0

Average   186953  133883     0 1227922  781815     0      219136     0
```

[Team LiB]

PREVIOUS NEXT



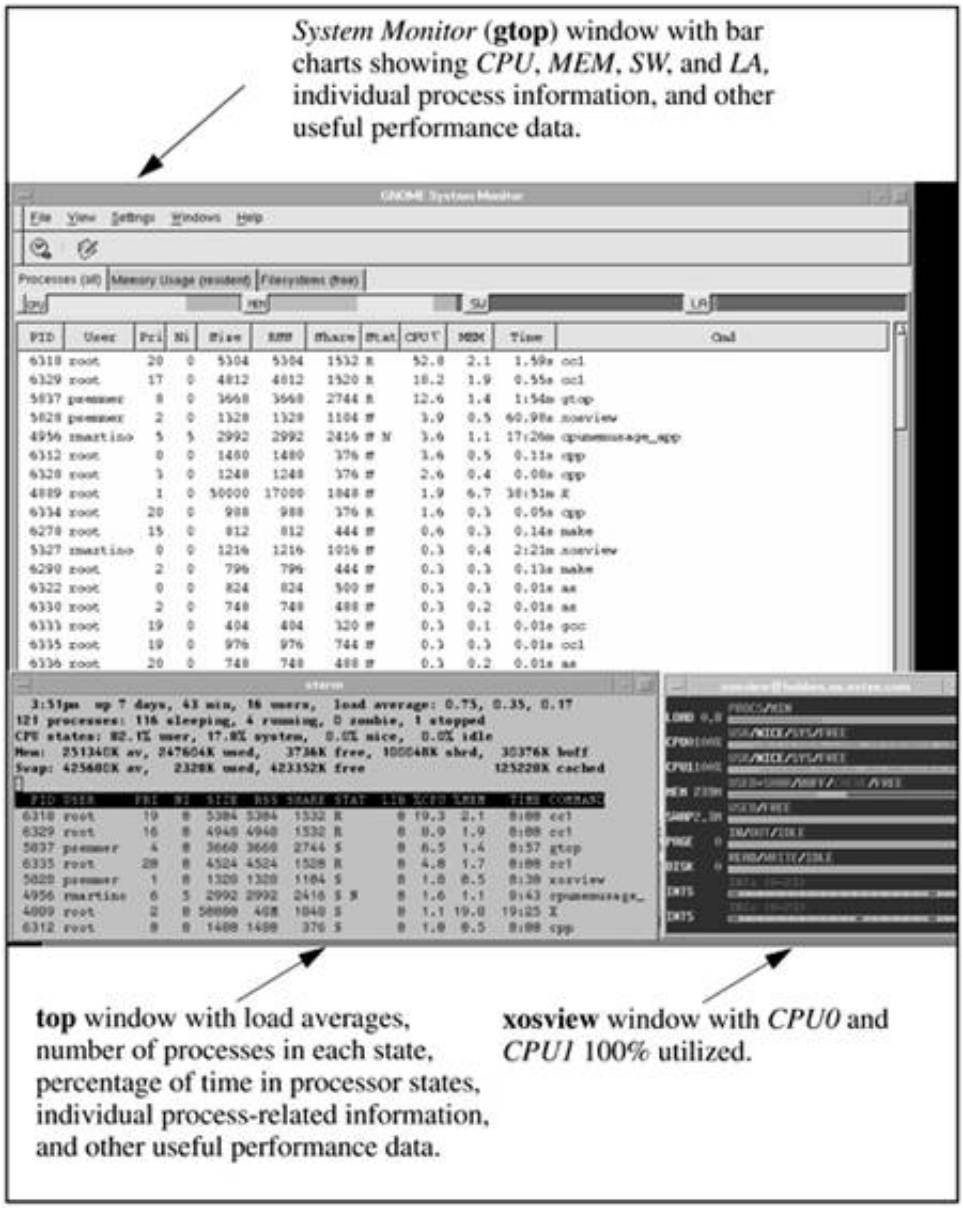
More Advanced and Graphical Performance Tools

The command line is a way of life when working with UNIX. UNIX grew out of the command line and is still primarily command line-based. Although you need to know a lot when issuing commands, especially when it comes to system performance, you can dig deeply very quickly with many of the commands I just covered.

You have the option with most UNIX variants to buy graphical performance tools. Some systems come with basic graphical performance tools, but will usually end up buying an advaced performance analysis tool if you want to perform advanced performance analysis. We'll take a quick look at a few performance tools in upcoming sections.

Figure 11-1 shows three performance tools that came with the Red Hat Linux system I used for many of the examples in this chapter:

Figure 11-1. Red Hat Linux Performance Tools Screen Shot



man page

top - a

The three performance tools shown in this diagram are **xosview** in the lower right, the *System Monitor* across the top of the screen, and **top** shown in the lower left. The *System Monitor* provides bar charts across the top of the screen that indicate the amount of CPU, Memory, Swap, and LAN utilization taking place. There is then tabular data supplied for every process on the system. The *System Monitor* is a graphical version of **top** that I invoked with the command **gtop** for **g**raphical **top**. **xosview** is a small load meter that you can keep running that provides bar charts of system activity shown in the bottom right window. This is the **X** Windows **o**perating **s**ystem **view** program, hence the name **xosview**. You can't see the bar charts clearly in this diagram, because this is a color-based application and the book is printed in only black and white. The bar charts are, however, clear on the computer screen. The final, and most often used, tool on UNIX systems is the character version of **top** that is running in the bottom left **xterm**. **top** is found on many UNIX variants and supplies a lot of useful system information.

Among the useful **top** system data displayed is the following:

- Load averages in the last one, five, and fifteen minutes.
- Number of existing processes and the number of processes in each state.

- Percentage of time spent in each of the processor states per processor on the system.

This same information is included in the bottom of the *System Monitor* window, which is covered by the **top** and **xosview** windows.



Next in the **top** window is memory data, including used, free, and shared.

Data is also provided for individual processes in a format similar to **ps**, including the following:

PID - Process ID number.

USER - Name of the owner of the process.

PRI - Current priority of the process.

NI - Nice value, ranging from -20 to +20.

SIZE - Total size of the process in kilobytes.

RSS - Resident size of the process in kilobytes.

STATE - Current state of the process.

TIME - Number of system and CPU seconds the process has consumed.

%CPU - CPU percentage.

%MEM - Memory percentage.

COMMAND - Name of the command the process is currently running.



As with most of the commands we have been covering, **top** is different among UNIX variants. You may see some different fields on the various UNIX variants. I am usually confident when I sit down at any UNIX system that I can run **top** and quickly see how the system is running. Most versions of **top** I have run are character-based applications, so you don't even need a graphics terminal to run them. I have run **top** in this example in character mode within an X terminal.

The system used in this example has two CPUs. If you look carefully in the **xosview** window, you'll see that both *CPU0* and *CPU1* are 100 percent used. At the time this screen shot was obtained, I was compiling the Linux kernel on this system, which consumed all the CPU resources on the system for a short period of time. You can see from both the **top** and *System Monitor* windows that the program **cc1**, used to compile the kernel, was consuming a substantial amount of the CPU resources on the system.

Figure 11-1 helps illustrate how different tools can help with viewing how system resources are consumed. **xosview** provides a quick reference, graphical overview of how many system resources are being consumed. **top** and *System Monitor* can then be used to determine the specific process consuming the most system resources.

HP GlancePlus/UX

Using UNIX commands to get a better understanding of what your system is doing requires you to do a lot of work. In the first case, issuing UNIX commands gives you the advantage of obtaining data about what is taking place on your system that very second. Unfortunately, you can't always issue additional commands to probe more deeply into an area, such as a process, about which you want to know more.

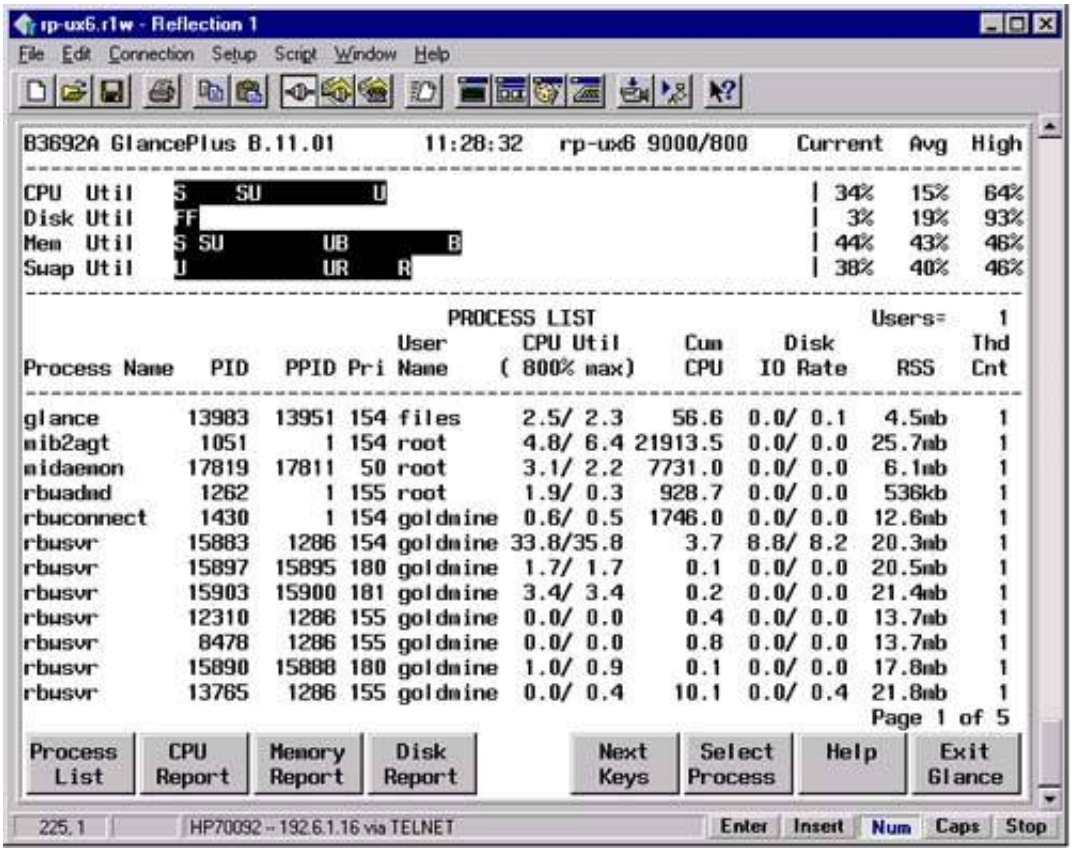
Now I'll describe another technique, a tool that can help get useful data in real time, allow you to investigate a specific process, and not bury you in reports. This tool is HP GlancePlus/UX (GlancePlus). This tool runs on several UNIX variants, including Solaris, HP-UX, and AIX.

GlancePlus can be run in character mode or in graphic mode. I chose to use the character-based version of GlancePlus, because this will run on any display, either graphics- or character-based, and the many colors used by the Motif version of GlancePlus do not show up well in a book. My examples are displayed much more clearly in the book when using the character mode. I recommend that you try both versions of GlancePlus to see which you prefer.

The system used in the examples has eight processors, 4 GBytes of RAM, and a substantial amount of EMC Symmetrix disk connected to it.

Figure 11-2 shows one of several interactive screens of GlancePlus. This one is the *Process List* screen, also referred to as the *Global* screen. This is the default screen when bringing up GlancePlus.

Figure 11-2. HP GlancePlus/UX *Process List* Screen Shot



Two features of the screen shown in Figure 11-2 are worth noticing immediately:



1. Four histograms at the top of the screen give you a graphical representation of your CPU, Disk, Memory, and Swap Utilization in a format much easier to assimilate than a column of numbers.
2. The "Process Summary" has columns similar to **ps -ef**, with which many system administrators are familiar and comfortable. GlancePlus, however, gives you the additional capability of filtering out processes that are using very few resources by specifying thresholds.

Using GlancePlus, you can take a close look at your system in many areas, including the following:

- Process List
- CPU Report
- Memory Report
- Swap Space
- Disk Report
- LAN Detail
- NFS by System
- PRM Summary (Process Resource Manager)
- I/O by File System
- I/O by Disk
- I/O by Logical Volume
- System Tables

Figure 11-2 is a GlancePlus screen shot.

Because the *Process List* shown in the example tells you where your system resources are going at the highest level, I'll start my description here. I am using a terminal emulator on my portable computer to display GlancePlus. I find that many system administrators use a PC and a terminal emulator to perform UNIX management functions. Keep in mind that the information shown on this screen can be updated at any interval you choose. If your system is running in a steady-state mode, you may want to have a long interval because you don't expect things to much change. On the other hand, you may have a dynamic environment and want to see the histograms and other information updated every few seconds. In either case, you can change the update interval to suit your needs. You can use the function keys at the bottom of the screen to go into other functional areas.

Process List Description

The *Process List* screen provides an overview of the state of system resources and active processes.

The top section of the screen (the histogram section) is common to the many screens of GlancePlus. The bottom section of the screen displays a summary of active processes.

Line 1 provides the product and version number of GlancePlus, the time, name of your system, and system type. In this case, we are running version 11.01 of GlancePlus.

Line 3 provides information about the overall state of the CPU. This tends to be the single most important piece of information that administrators want to know about their system - Is my CPU overworked?

The CPU Utilization bar is divided into the following parts:

1. "S" indicates the amount of time spent on "system" activities such as context switching and system calls.
2. "N" indicates the amount of time spent running "nice" user processes (those run at a low priority).
3. "U" indicates the amount of time spent running user processes.
4. "R" indicates real-time processes.
5. "A" indicates the amount of time spent running processes at a negative "nice" priority.

The far right of line 3 shows the percentage of CPU utilization. If your system is "CPU-Bound," you will consistently see this number near 100 percent. You get statistics for Current, Average (since analysis was begun), and High.

Line 4 shows Disk Utilization for the busiest mounted disk. This bar indicates the percentage of File System and Virtual Memory disk I/O over the update interval. This bar is divided into two parts:

1. "F" indicates the amount of file system activity of user reads and writes and other non-paging activities.
2. "V" indicates the percentage of disk I/O devoted to paging virtual memory.

The Current, Avg, and High statistics have the same meaning as in the CPU Utilization description.

Line 5 shows the system memory utilization. This bar is divided into three parts:

1. "S" indicates the amount of memory devoted to system use.
2. "U" indicates the amount of memory devoted to user programs and data.
3. "B" indicates the amount of memory devoted to buffer cache.

The Current, Avg, and High statistics have the same meaning as in the CPU Utilization description.

Line 6 shows Swap Util information, which is divided into two parts:

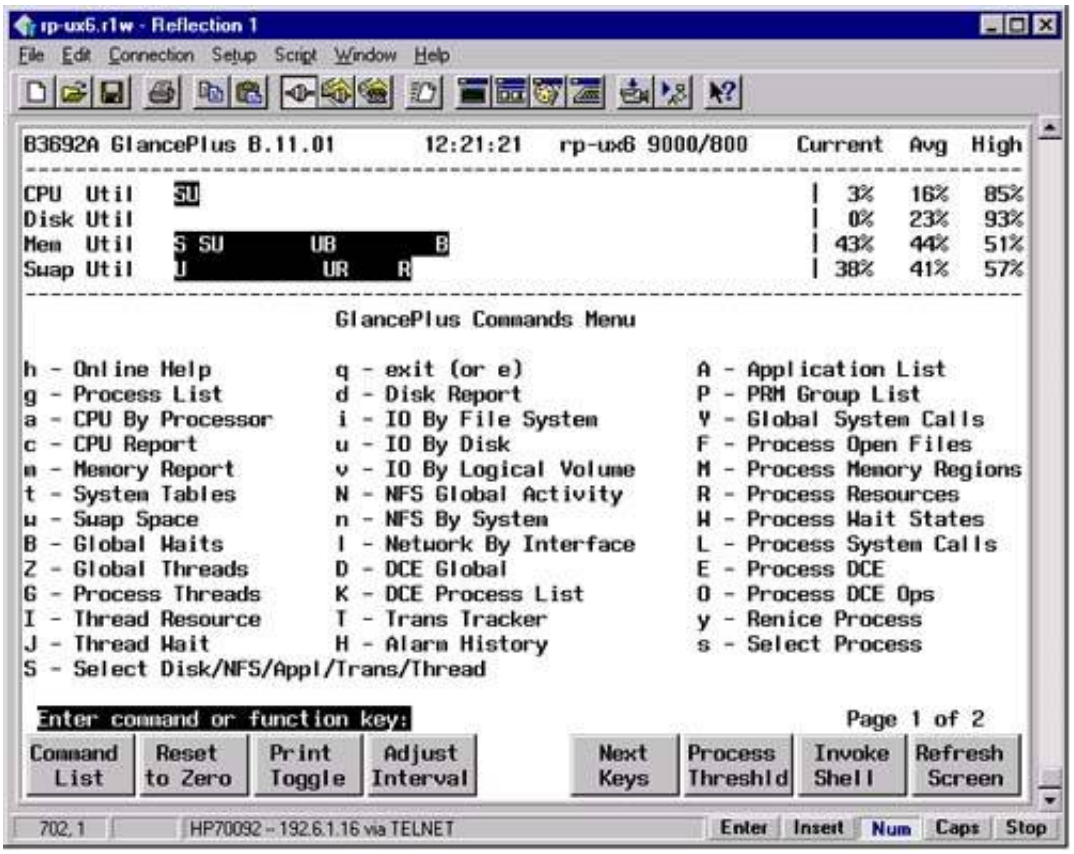
1. "R" indicates reserved, but not in use.
2. "U" indicates swap space in use.

All three of these areas (CPU, Memory, and Disk) may be further analyzed by using the F2, F3, and F4 function keys, respectively. Again, you may see different function keys, depending on the version of GlancePlus you are running. When you select one of these keys, you move from the *Process List* screen to a screen that provides more in-depth functions in the selected area. In addition, more detailed screens are available for many other system areas. Because most investigation beyond the *Process List* screen takes place on the CPU, Memory, and Disk screens, I'll describe these in more detail shortly.

The bottom of the *Process List* screen shows the active processes running on your system. Because there are typically many processes running on a UNIX system, you may want to consider using the **o** command to set a threshold for CPU utilization. If you set a threshold of five percent, for instance, then only processes that exceed the average CPU utilization of five percent over the interval will be displayed. There are other types of thresholds that can be specified such as the amount of RAM used (Resident Size). If you specify thresholds, you see only the processes you're most interested in, that is, those consuming the greatest system resources.

There is a line for each active process that meets the threshold requirements you defined. There may be more than one page of processes to display. The message in the bottom-right corner of the screen indicates which page you are on. You can scroll forward to view the next page with **f** and backwards with **b**. Usually only a few processes consume most of your system resources, so I recommend setting the thresholds so that only one page of processes is displayed. There are a whole series of commands you can issue in GlancePlus. Figure 11-7 toward the end of this section shows the commands recognized by GlancePlus.

Figure 11-7. HP GlancePlus/UX *Command List* Screen 1



Here is a brief summary of the process headings:

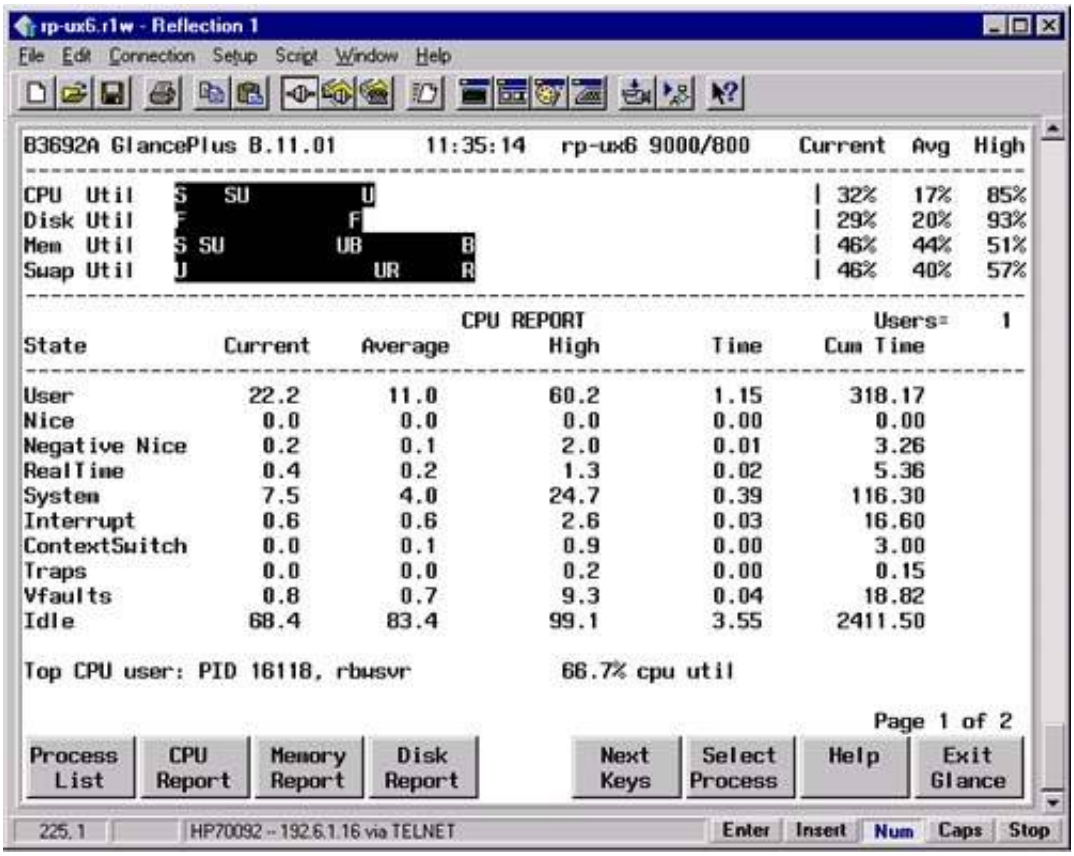
Process Name	The name or abbreviation used to load the executable program.
PID	The process identification number.
PPID	The PID of the parent process.
Pri	The priority of the process. The lower the number, the higher the priority. System-level processes usually run between 0 and 127. Other processes usually run between 128 and 255. "Nice" processes are those with the lowest priority and they have the largest number.
User Name	Name of the user who started the process.
CPU Util	The first number is the percentage of CPU utilization that this process consumed over the update interval. Note that this is 800% maximum for our eight-processor system. The second number is the percentage of CPU utilization that this process consumed since GlancePlus was invoked. Most system administrators leave GlancePlus running continuously on their systems with a low update interval. Since GlancePlus uses very little system overhead, there is virtually no penalty for this.
Cum CPU	The total CPU time used by the process. GlancePlus uses the "midaemon" to gather information. If the **midaemon** started before the process, you will get an accurate measure of cumulative CPU time used by the process.
Disk IO Rate	The first number is the average disk I/O rate per second over the last update interval. The second number is the average disk I/O rate since GlancePlus was started or since the process was started. Disk I/O can mean a lot of different things. Disk I/O could mean taking blocks of data off the disk for the first time and putting them in RAM, or it could be entirely paging and swapping. Some processes will simply require a lot more Disk I/O than others. When this number is very high, however, take a close look at whether or not you have enough RAM. Keep in mind that pageout activity, such as deactivation and swapping, are attributed to the *vhand* process.

RSS Size	The amount of RAM in KBytes that is consumed by the process. This is called the Resident Size. Everything related to the process that is in RAM is included in this column, such as the process's data, stack, text, and shared memory segments. This is a good column to inspect. Because slow systems are often erroneously assumed to be CPU-bound, I always make a point of looking at this column to identify the amount of RAM that the primary applications are using. This is often revealing. Some applications use a small amount of RAM but use large data sets, a point often overlooked when RAM calculations are made. This column shows all the RAM your process is currently using.
Block On	The reason the process was blocked (unable to run). If the process is currently blocked, you will see why. If the process is running, you will see why it was last blocked. There are many reasons a process could be blocked. After *Thd Cnt* is a list of the most common reasons for the process being blocked.
Thd Cnt	The total number of threads for this current process.
Abbreviation	Reason for the Blocked Process
CACHE	Waiting for a cache buffer to become available
DISK	Waiting for a disk operation to complete
INODE	Waiting for an inode operation to complete
IO	Waiting for a non-disk I/O to complete
IPC	Waiting for a shared memory operation to complete
LAN	Waiting for a LAN operation to complete
MESG	Waiting for a message queue operation to complete
NFS	Waiting for an NFS request to complete
PIPE	Waiting for data to or from a pipe
PRI	Waiting because a higher-priority process is running
RFA	Waiting for a Remote File Access to complete
SEM	Waiting for a semaphore to become available
SLEEP	Waiting because the process called **sleep** or **wait**
SOCKT	Waiting for a socket operation to complete
SYS	Waiting for system resources
TERM	Waiting for a terminal transfer
VM	Waiting for a virtual memory operation to complete
OTHER	Waiting for a reason GlancePlus can't determine

CPU Report Screen Description

If the *Process List* screen indicates that the CPU is overworked, you'll want to refer to the *CPU Report* screen shown in Figure 11-3. It can provide useful information about the seven types of states on which GlancePlus reports.

Figure 11-3. HP GlancePlus/UX *CPU Report* Screen Shot



For each of the seven types of states, there are columns that provide additional information. Following is a description of the columns:

Current	Displays the percentage of CPU time devoted to this state over the last time interval.
Average	Displays the average percentage of CPU time spent in this state since GlancePlus was started.
High	Displays the highest percentage of CPU time devoted to this state since GlancePlus was started.
Time	Displays the CPU time spent in this state over the last interval.
Cum Time	Displays the total amount of CPU time spent in this state since GlancePlus was started.

A description of the seven states follows:

User	CPU time spent executing user activities under normal priority.
Nice	CPU time spent running user code in nice mode.
Negative Nice	CPU time spent running code at a high priority.
Realtime	CPU time spent executing real-time processes that run at a high priority.
System	CPU time spent executing system calls and programs.
Interrupt	CPU time spent executing system interrupts. A high value here may indicate of a lot of I/O, such as paging and swapping.
ContSwitch	CPU time spent context switching between processes.
Traps	CPU time spent handling traps.

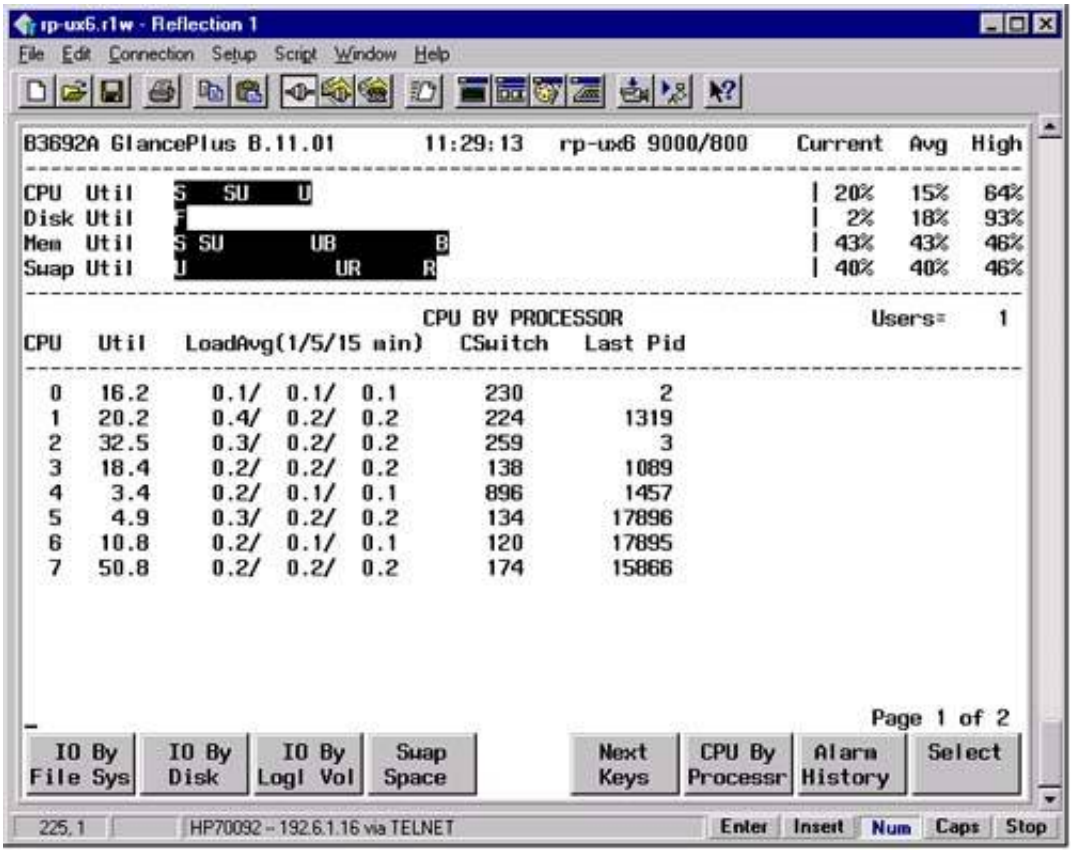
Vfaults	CPU time spent handling page faults.
Idle	CPU time spent idle.

The *CPU Report* screen also shows your system's run queue length or load average. This is displayed on the second page of the *CPU Report* screen. The Current, Average, and High values for the number of runnable processes waiting for the CPU are shown. You may want to get a gauge of your system's run queue length when the system is mostly idle and compare these numbers with those you see when your system is in normal use.

The final area reported on the *CPU Report* screen is load average, system calls, interrupts, and context switches. I don't inspect these too closely, because if one of these is high, it is normally the symptom of a problem and not the cause of a problem. If you correct a problem, you will see these numbers reduced.

You can use GlancePlus to view all the CPUs in your system, as shown in Figure 11-4. This is an eight-processor system.

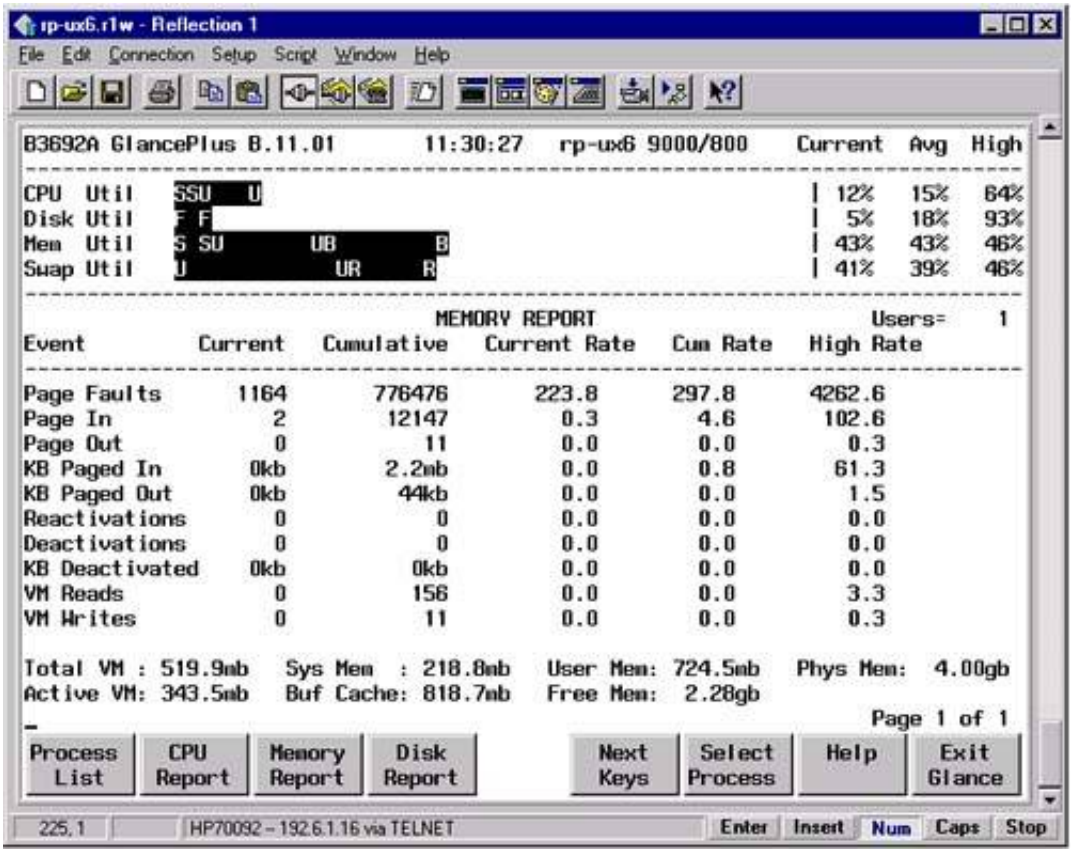
Figure 11-4. *All CPUs* Screen in GlancePlus



Memory Report Screen Description

The *Memory Report* Screen, shown in Figure 11-5, provides information on several types of memory management events. The statistics shown are in the form of counts, not percentages. You may want to look at these counts for a mostly idle system and then observe what takes place as the load on the system is incrementally increased. My experience has been that many more memory bottlenecks occur than CPU bottlenecks, so you may find this screen revealing.

Figure 11-5. HP GlancePlus/UX *Memory Report* Screen Shot



The following five statistics are shown for each memory management event:

Current	The number of times an event occurred in the last interval. The count changes if you update the interval, so you may want to select an interval you are comfortable with and stick with it.
Cumulative	The sum of all counts for this event since GlancePlus was started.
Current Rate	The number of events per second.
Cum Rate	Average of the rate over the cummulative collection interval.
High Rate	The highest rate recorded.

Following are brief descriptions of the memory management events for which statistics are provided:

Page Faults	Any address translation fault such as reclaims, pid faults, and so on.
Page In/Page Out	Pages of data moved from virtual memory (disk) to physical memory (page in), or vice versa.
KB Paged In	The amount of data paged in because of page faults.
KB Paged Out	The amount of data paged out to disk.

Reactivations/Deactivations

The number of processes swapped in and out of memory. A system low on RAM will spend a lot of time swapping processes in and out of RAM. If a lot of this type of swapping is taking place, you may have high CPU utilization and see some other statistics may increase as well. These may only be symptoms that a lot of swapping is taking place.

KB Reactivated	The amount of information swapped into RAM as a result of processes having been swapped out earlier due to insufficient RAM.
KB Deactivated	The amount of information swapped out when processes are moved to disk.
VM Reads	The total count of the number of vitual memory reads to disk. The higher this number, the more often your system is going to disk.
VM Writes	The total count of memory management I/O.

The following values are also on the Memory screen:

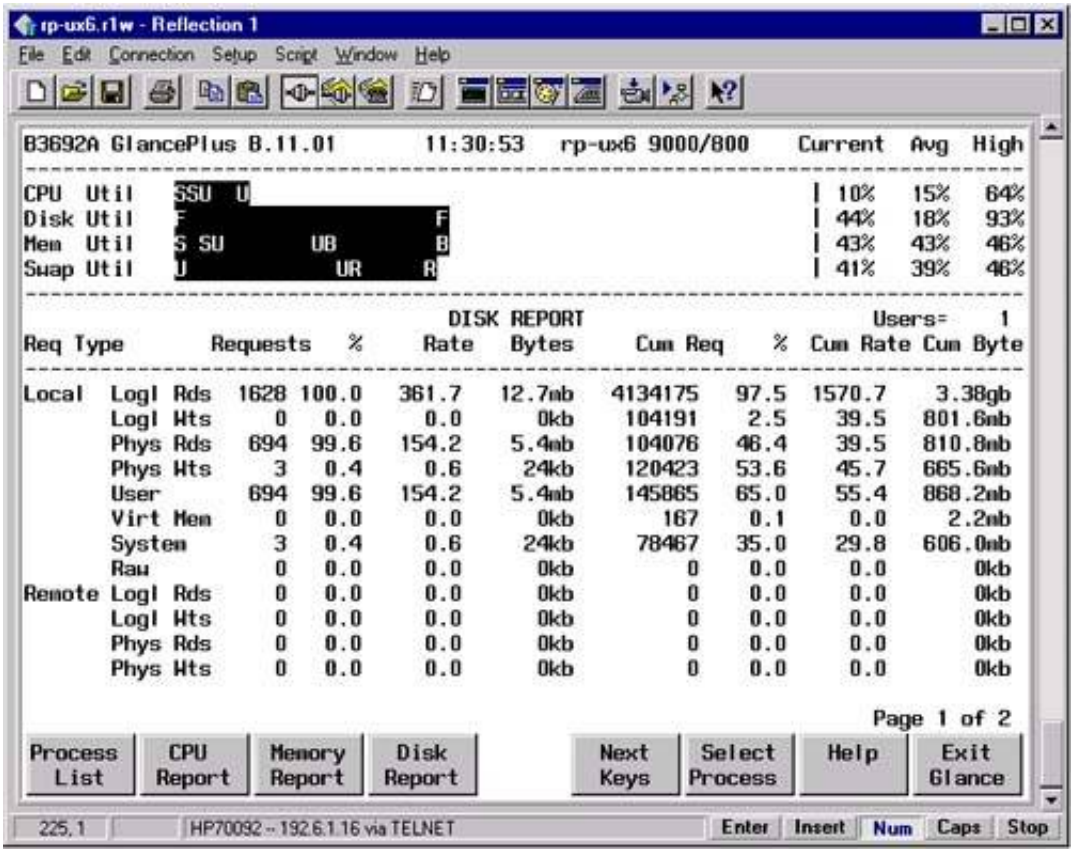
Total VM	The amount of total virtual memory used by all processes.
Active VM	The amount of virtual memory used by all active processes.
Sys Mem	The amount of memory devoted to system use.
Buf Cache Size	The current size of buffer cache.
User Mem	The amount of memory devoted to user use.
Free Memory	The amount of RAM not currently allocated for use.
Phys Memory	The total RAM in your system.

This screen gives you a lot of information about how your memory subsystem is being used. You may want to view some statistics when your system is mostly idle and when it is heavily used and compare the two. Some good numbers to record are "Free Memory" (to see whether you have any free RAM under either condition) and "Total VM" (to see how much virtual memory has been allocated for all your processes). A system that is RAM-rich will have available memory; a system that is RAM-poor will allocate a lot of virtual memory.

Disk Report Screen Description

The *Disk Report* screen appears in Figure 11-6. You may see groupings of "local" and "remote" information.

Figure 11-6. HP GlancePlus/UX *Disk Report* Screen Shot



There are eight disk statistics provided for eight events related to logical and physical accesses to all the disks mounted on the local system. These events represent all the disk activity taking place on the system.

Here are descriptions of the eight disk statistics provided:

Requests	The total number of requests of that type over the last interval.
%	The percentage of this type of disk event relative to other types.
Rate	The average number of requests of this type per second.
Bytes	The total number of bytes transferred for this event over the last interval.
Cum Req	The cumulative number of requests since GlancePlus started.
%	The relative percentage of this type of disk event since GlancePlus started.
Cum Rate	Average of the rate over the cumulative collection interval.
Cum Bytes	The total number of bytes transferred for this type of event since GlancePlus started.

Next are descriptions of the disk events for which these statistics are provided, which may be listed under "Local" on your system:

Logl Rds and Logl Wts

The number of logical reads and writes to a disk. Because disks normally use memory buffer cache, a logical read may not require physical access to the disk.

Phys Rds	The number of physical reads to the disk. These physical reads may be due to either file system logical reads or to virtual memory management.
Phys Wts	The number of physical writes to the disk. This may be due to file system activity or virtual memory management.

User	The amount of physical disk I/O as a result of user file I/O operations.
Virtual Mem	The amount of physical disk I/O as a result of virtual memory management activity.
System	Housekeeping I/O such as inode updates.
Raw	The amount of raw mode disk I/O.

A lot of disk activity may also take place as a result of NFS mounted disks. Statistics are provided for "Remote" disks as well.

Disk access is required on all systems. The question to ask is: What disk activity is unnecessary and slowing down my system? A good place to start is to compare the amount of "User" disk I/O with "Virtual Mem" disk I/O. If your system is performing much more virtual memory I/O than user I/O, you may want to investigate your memory needs.

GlancePlus Summary

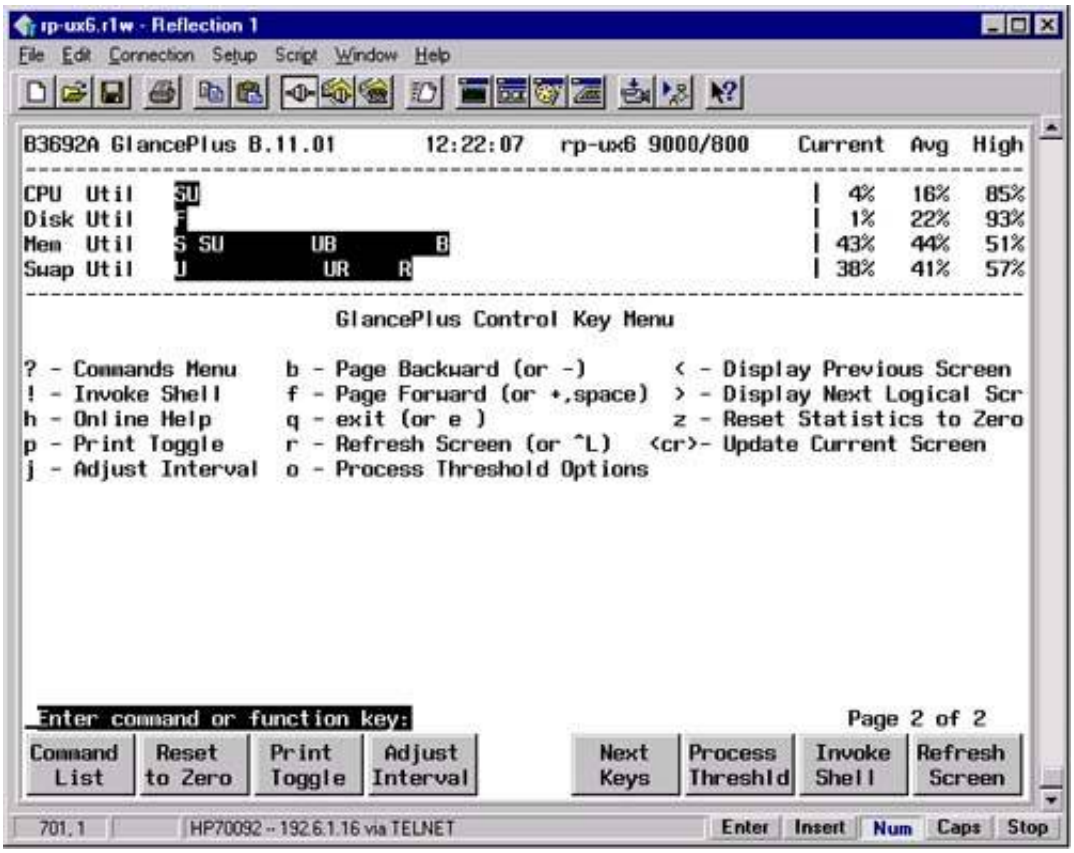
In addition to the Process List, or Global, screen and the CPU, Memory, and Disk screens described earlier, there are many other useful screens including the following:

Swap Space	Shows details of all swap areas. May be called by another name in other releases.
Netwk By Intrface	Gives details about each LAN card configured on your system. This screen may have another name in other releases.
NFS Global	Provides details on inbound and outbound NFS-mounted file systems. May be called by another name in other releases.
Select Process	Allows you to select a single process to investigate. May be called by another name in other releases.
I/O By File Sys	Shows details of I/O for each mounted disk partition.
I/O By Disk	Shows details of I/O for each mounted disk.
I/O By Logl Vol	Shows details of I/O for each mounted logical volume.
System Tables	Shows details of internal system tables.
Process Threshold	Defines which processes will be displayed on the Process List screen. May be called by another name, such as the Global screen, in other releases.

As you can see, although I described the four most commonly used screens in detail, you can use many others to investigate your system further.

There are also many commands that you can issue within GlancePlus. Figures 11-7 and 11-8 show the *Command List* screens in GlancePlus.

Figure 11-8. HP GlancePlus/UX *Command List* Screen 2



Using OpenView Performance Agent to Identify Bottlenecks

OpenView Performance Agent allows you to view many metrics related to system performance that can help you identify the source of bottlenecks in your system. You can use the graphical version of GlancePlus, called gpm, to specify the metrics you want to keep track of. You can then view them in the gpm interface and sort them a variety of different ways.

The following are the most important types of bottlenecks you can encounter on a system and the metrics associated with each type of bottleneck. This information was provided by Doug Grumann and Stephen Ciullo of Hewlett Packard, who are two performance experts.

1. CPU bottleneck Using OpenView Performance Agent:
    - Consistent High global CPU utilization with *GBL_CPU_TOTAL_UTIL*>90% and next bullet.
    - Significant *Run Queue* or *Load Average* indicated by *GBL_PRI_QUEUE* or *GBL_RUN_QUEUE*>3.
    - Look for processes blocked on priority with *PROC_STOP_REASON=PRI*.
1. System CPU bottleneck using OpenView Performance Agent (same as 1 with addition of first bullet):
    - Most of the CPU time spent in kernel mode with *GBL_CPU_SYS_MODE_UTIL*>50%.
    - Consistent High global CPU utilization with *GBL_CPU_TOTAL_UTIL*>90% and next bullet.
    - Significant *Run Queue* or *Load Average* indicated by *GBL_PRI_QUEUE* or *GBL_RUN_QUEUE*>3.
    - Look for processes blocked on priority with *PROC_STOP_REASON=PRI*.
1. Context switching bottleneck using OpenView Performance Agent (same as 2 with addition of first bullet):

- Significant CPU time spent switching with *GBL_CPU_CSWITCH*>30%.
- Most of the CPU time spend in kernel mode with *GBL_CPU_SYS_MODE_UTIL*>50%.
- Consistent High global CPU utilization with *GBL_CPU_TOTAL_UTIL*>90% and next bullet.
- Significant *Run Queue* or *Load Average* indicated by *GBL_PRI_QUEUE* or *GBL_RUN_QUEUE*>3.
- Look for processes blocked on priority with *PROC_STOP_REASON=PRI*.

1. User CPU bottleneck Using OpenView Performance Agent (same as 1 with addition of first bullet):
    - Most of the CPU time spent in user mode with *GBL_CPU_USER_MODE_UTIL*>50%.
    - Consistent High global CPU utilization with *GBL_CPU_TOTAL_UTIL*>90% and next bullet.
    - Significant *Run Queue* or *Load Average* indicated by *GBL_PRI_QUEUE* or *GBL_RUN_QUEUE*>3.
    - Look for processes blocked on priority with *PROC_STOP_REASON=PRI*.
1. Disk bottleneck Using OpenView Performance Agent:
    - At least one disk device with consistently high utilization with *BYDSK_UTIL*>50%.
    - Queue lengths greater than zero with *BYDSK_QUEUE*>0.
    - Processes or threads blocked on I/O for a variety of reasons with *PROC_STOP_REASON=CACHE, DISK* or *IO*.
    - Look for processes blocked on priority with *PROC_STOP_REASON=PRI*.
1. Buffer Cache bottleneck Using OpenView Performance Agent:
    - Moderate utilization of at least one disk with *BYDSK_UTIL*>25%.
    - Queue lengths greater than zero with *BYDSK_QUEUE*>0.
    - Low Buffer cache read hit percentage with *GBL_MEM_CACHE_HIT_PCT*<90%.
    - Processes or threads blocked on cache with *PROC_STOP_REASON=CACHE*.
1. Memory bottleneck Using OpenView Performance Agent:
    - High physical memory utiliztion with *GBL_MEM_UTIL*>95%.
    - Significant pageouts or any deactivations with *GBL_MEM_PAGEOUT_RATE>1* or *GBL_MEM_SWAPOUT_RATE>0*.
    - vhand processes consistently active with vhand's *PROC_CPU_TOTAL_UTIL*>5%.
    - Processes or threads blocked on virtual memory with *PROC_STOP_REASON=VM*.
1. Networking bottleneck Using OpenView Performance Agent:
    - High network packet rates with *GBL_NET_PACKET_RATE>2* average. Keep in mind this varies greatly depending on configuration.
    - Any output queuing taking place with *GBL_NET_OUTQUEUE>0.*
    - Higher than normal number of processes or threads blocked on networking with *PROC_STOP_REASON=NFS, LAN, RPC* or *SOCKET GBL_NETWORK_SUBSYSTEM_QUEUE*>average.
    - One CPU with a high system mode CPU utilization while other CPUs are mostly idle with *BYCPU_CPU_INTERRUPT_TIME>30*.
    - Using *lanadmin,* check for frequent incrementing of *Outbound Discards* or excessive *Collisions.*

In order to identify a problem on your system, you must first characterize your system when it is running smoothly and has no problems. Should your system start to perform poorly in some respect or another, you can compare the performance data of a smoothly running system to one with potential problems.

HP OpenView Performance Agent and HP OpenView Performance Analyzer/UX

There are performance tools that track and chart data over a long period of time. System Administrators often call this exercise "capacity planning." The goal of capacity planning is to view what system resources have been consumed over a long period of time and determine what adjustments or additions can be made to the system to improve performance and plan for the future. We'll use HP OpenView Performance Agent (what used to be MeasureWare Agent) and HP OpenView Performance Analyzer/UX (what used to be PerfView Analyzer) together to take a look at the performance of a system. These tools run on HP-UX and are similar to many advanced tools that run on other UNIX variants.

The OpenView Performance Agent is installed on individual systems throughout a distributed environment. It collects resource and performance measurement data on the individual systems. The OpenView Performance Analyzer/UX management console, which you would typically install on a management system, is then used to display the historical OpenView Performance Agent data. You could also set alarms to be triggered off by exception conditions using the OpenView Performance agent. For instance, if the OpenView Performance agent detects an exception condition, such as CPU utilization greater than 90%, it produces an alarm message. The alarm messages are then displayed with OpenView Performance Analyzer/UX. We're going to use the OpenView Performance Analyzer/UX in our upcoming examples; however, there are really three OpenView Performance components:

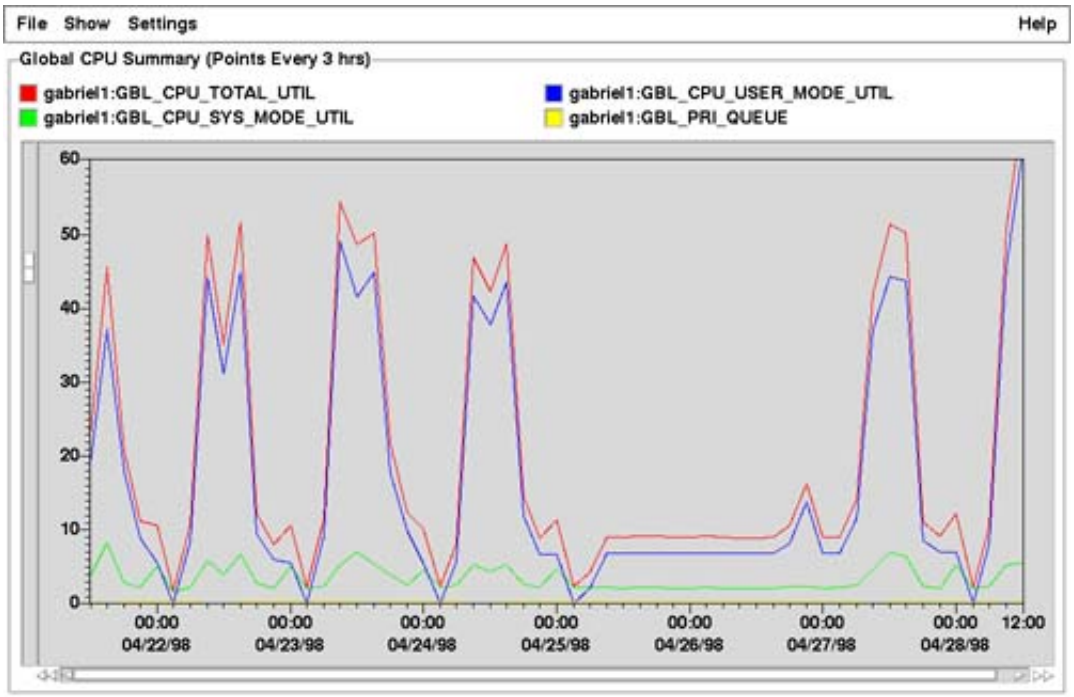
Monitor	Provides alarm monitoring capability by accepting alarms from OpenView Performance and displays alarms.
Planner	Provides forecasting capability by extrapolating OpenView Performance data for forecasts.
Analyzer	Analyzes OpenView Performance data from multiple systems and displays data. You can view the data from multiple systems simultaneously.

In our example, we will be working with a single system. We'll take the OpenView Performance data, collected over roughly a one-week period, and display some of it. In this example, we won't take data from several distributed systems and we'll use only one server in the example.

HP OpenView Performance Agent produces log files that contain information about the system resource consumption. The longer HP OpenView Performance Agent runs, the longer it records data in the log files. I am often called to review systems that are running poorly to propose system upgrades. I usually run HP OpenView Performance Agent for a minimum of a week so that I obtain log information over a long enough period of time to obtain useful data. For some systems, this time period is months. For other systems with a regular load, a week may be enough time.

After having run OpenView Performance for a week, I invoked OpenView Performance Analyzer/UX to see the level of system resource utilization that took place over the week. The graphs we'll review are CPU, Memory, and Disk. Figure 11-9 shows *Global CPU Summary* for the week:

Figure 11-9. *Global CPU Summary* Screen



You can adjust every imaginable feature of this graph with OpenView Performance Analyzer/UX. Unfortunately, the color in this graph is lost in the book. The colors used allow you to discern the parameters when viewing the graph on the computer screen. Total CPU utilization is always the top point in the graph and it is the sum of system and user mode utilization.

Figure 11-9 shows classic CPU utilization with prime hours reflecting high CPU utilization and non-prime hours reflecting low CPU utilization. In some respects, however, this graph can be deceiving. Because there is a data point occurs every three hours, hence the eight ticks per 24-hour period, you don't get a view of the actual CPU utilization during a much smaller window of time. We can't, for instance, see precisely what time in the morning the CPU becomes heavily used. We can see that it is between the second and third tick, but this is a long time period - between 6:00 and 9:00 am. The same lack of granularity is true at the end of the day. We see a clear fall-off in CPU utilization between the fifth and seventh ticks, but this does not give us a well-defined view. Figure 11-10 shows CPU utilization during a much shorter time window.

Figure 11-10. *Global CPU Summary* - Short Time Period

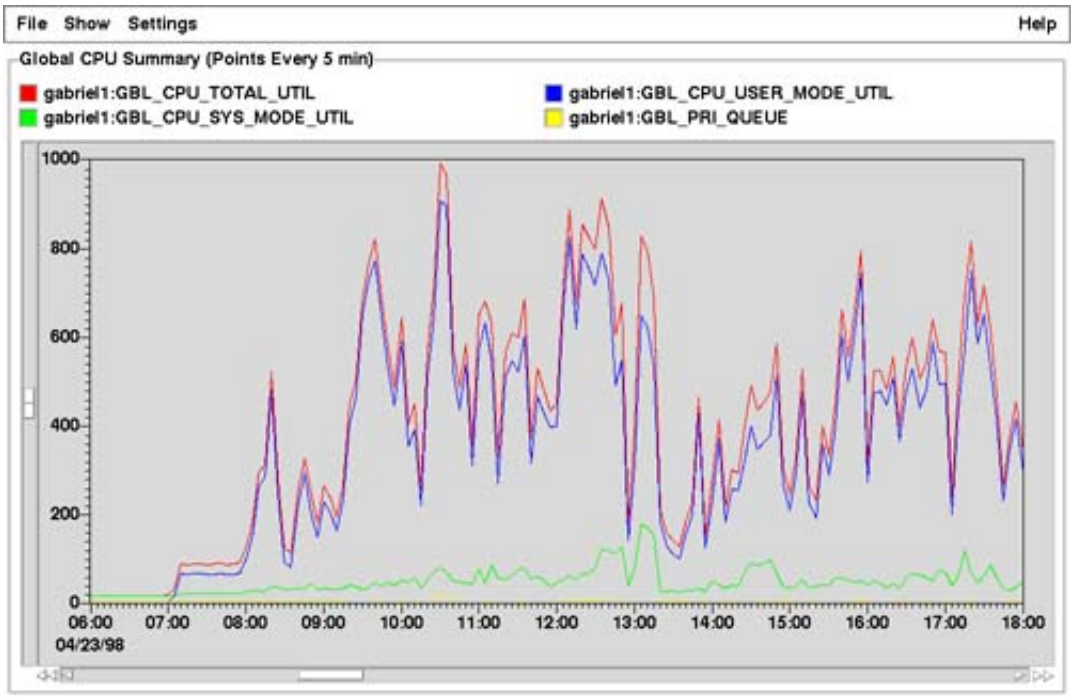
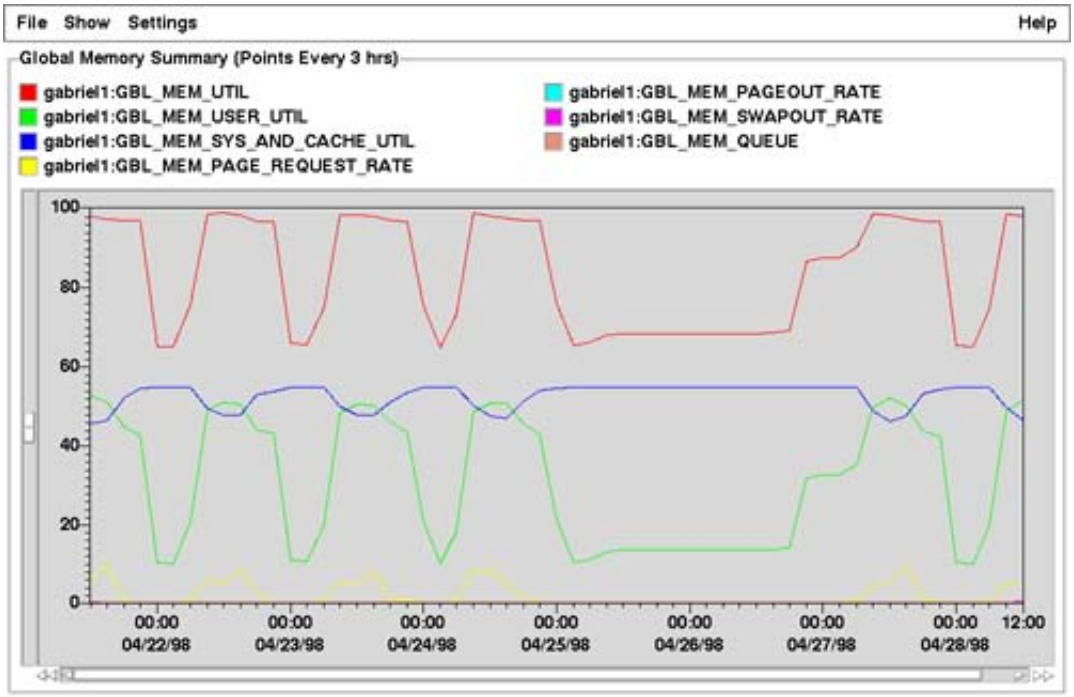


Figure 11-10 shows a finer granularity of CPU utilization during the shorter time window. The much finer granularity of this window makes clear the activity spikes that occur throughout the day. For instance, a clear login spike occurs at 8:30 am.

Memory utilization can also be graphed over the course of the week, as shown in Figure 11-11.

Figure 11-11. Global Memory Summary



The user memory utilization is the bottom line of the graph, which roughly corresponds to the CPU utilization shown earlier. User memory utilization is low during non-prime hours and high during prime hours.

System memory utilization is the middle line of the graph, which remains fairly steady throughout the week.

Total memory utilization is always the top line of the graph, and it is the sum of system and user utilization. It rises and drops with user utilization, because system memory utilization remains roughly the same.

The three-hour interval between data points on this graph may not give us the granularity we require. Figure 11-12 shows memory utilization during a much shorter time window.

Figure 11-12. *Global Memory Summary* - Short Time Period

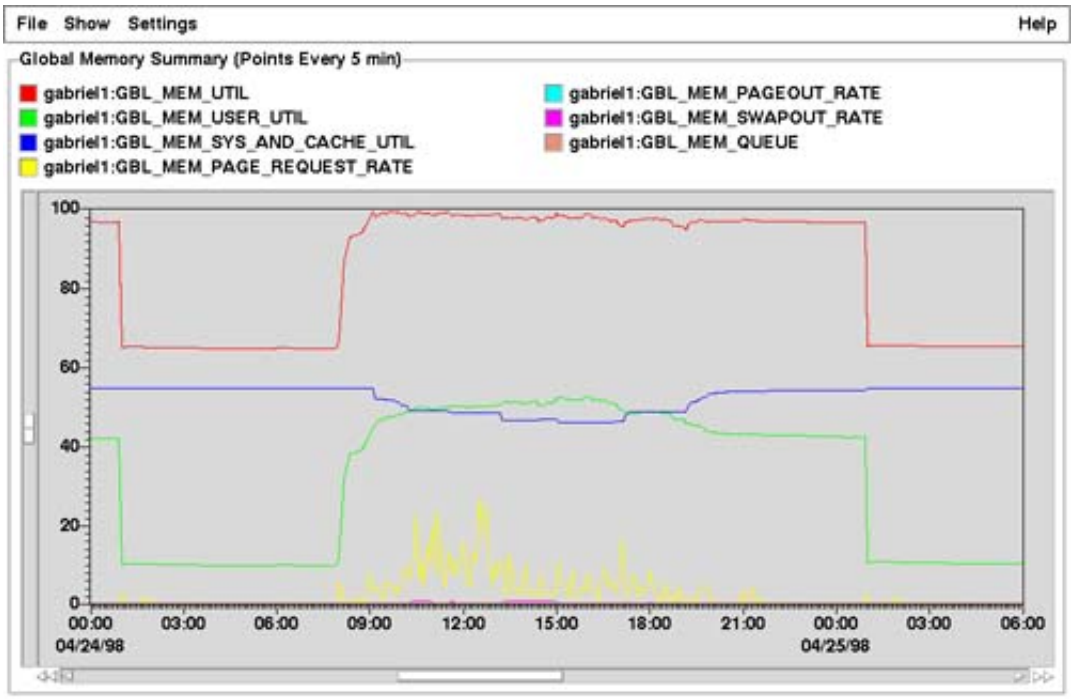
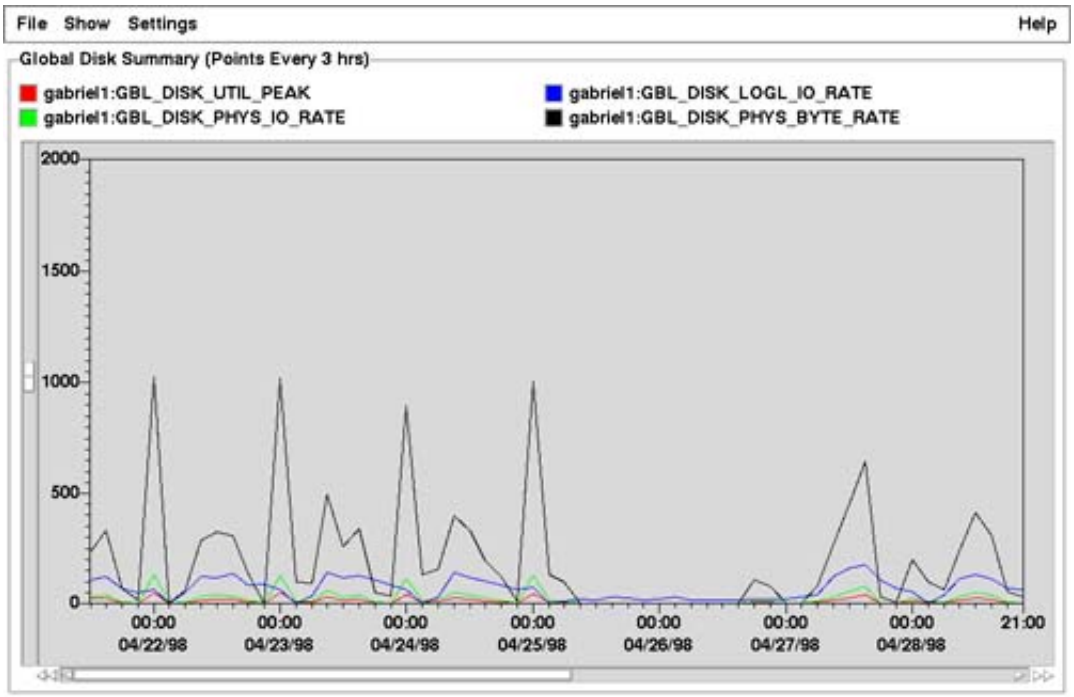


Figure 11-12 shows a finer granularity of memory utilization during the shorter time window. You can now see precisely how memory utilization is changing over roughly one day.

Disk utilization can also be graphed over the course of the week, as shown in Figure 11-13.

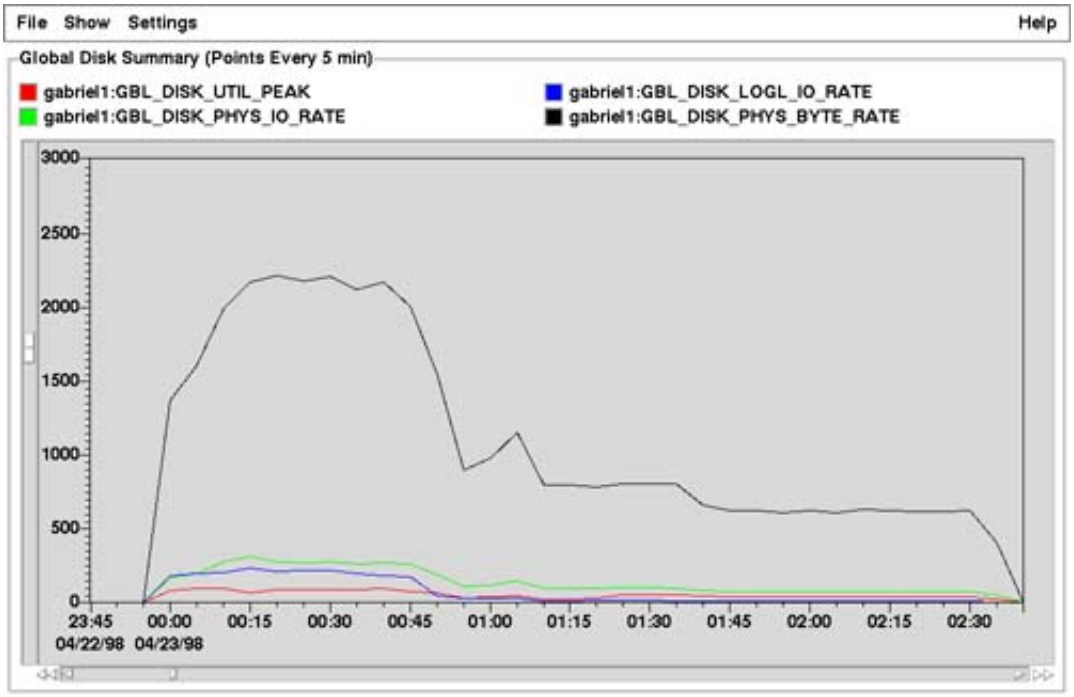
Figure 11-13. Global Disk Summary



Like the CPU and memory graph, this is an entire week of disk usage. Because many spikes occur on this graph, we would surely want to view and analyze much shorter time windows.

Figure 11-14 shows disk utilization during a much shorter time window.

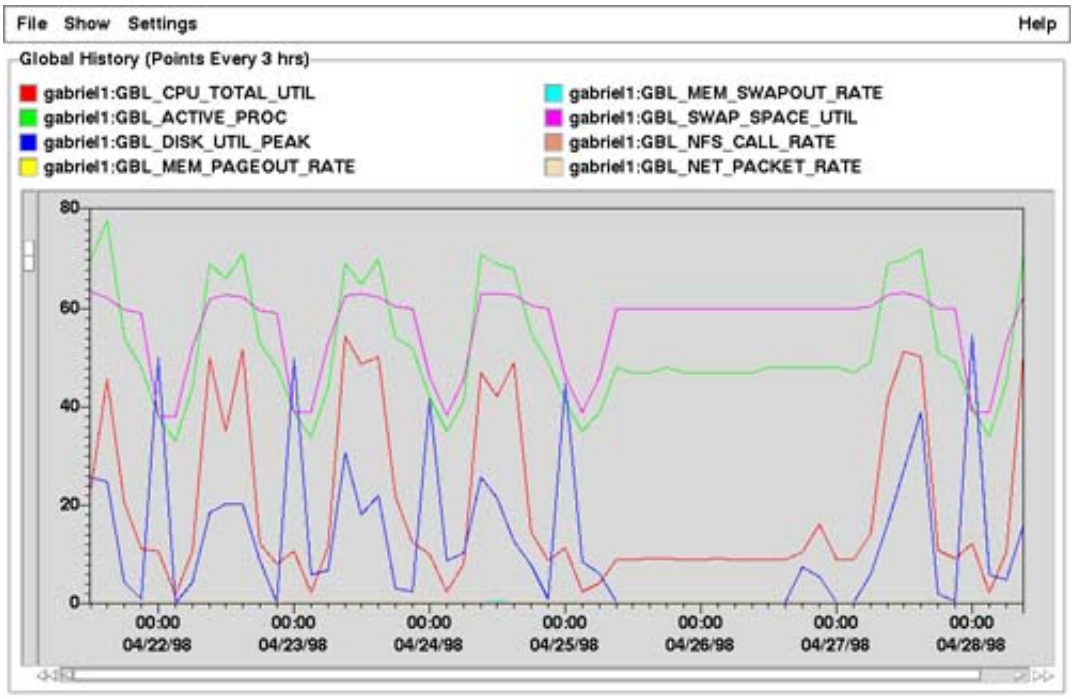
Figure 11-14. *Global Disk Summary* - Short Time Period



This much shorter time window, of roughly three hours, shows a lot more detail. There are tremendous spikes in disk activity occurring in the middle of the night. These could take place for a variety of reasons, including batch job processing or system backup.

You are not limited to viewing parameters related to only one system resource at a time. You can also view the way many system resources are used simultaneously, as shown in Figure 11-15.

Figure 11-15. *Global Summary History* Screen



Many system resources are present on this graph, including CPU, disk, and memory. You would surely want to view a much shorter time period when displaying so many system resources simultaneously.

Figure 11-16 shows the same parameters during a much shorter time window.

Figure 11-16. *Global Summary* - Short Time Period

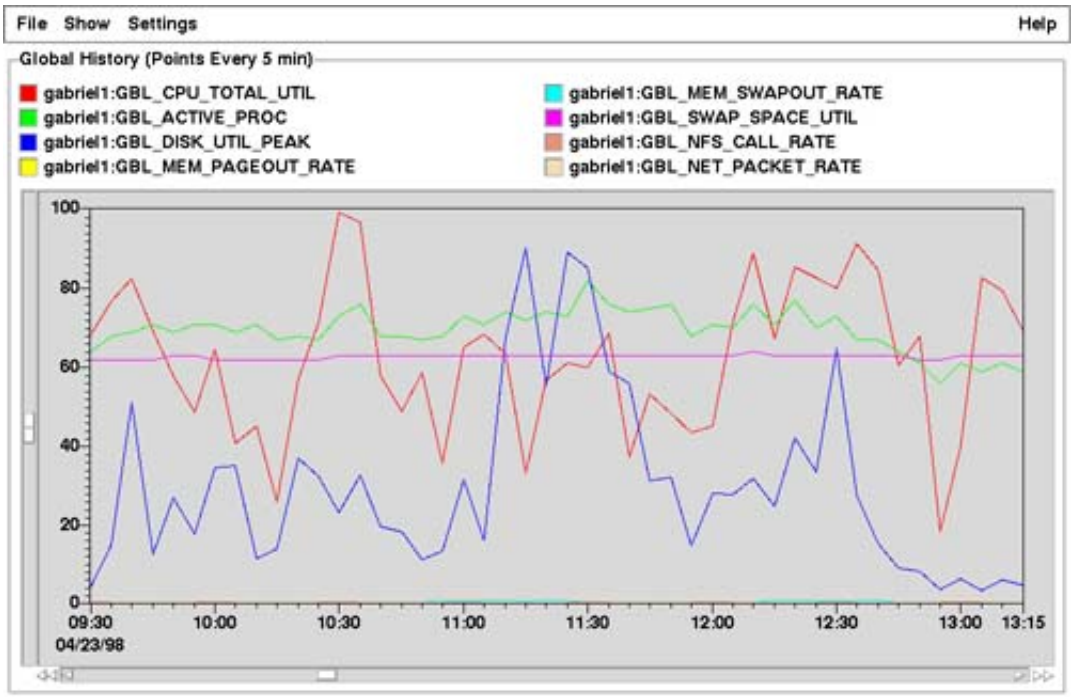


Figure 11-16 shows a finer granularity of the utilization of many system resources during the shorter time window. You can now view the ways in which various system resources are related to other system resources.

You can find the status of OpenView Performance Analyzer/UX running on your system with a useful command called **perfstat**. The following example shows issuing the **perfstat** command with the *-?* option to see all **perfstat** options:

```
# perfstat -?

usage: perfstat [options]

 Unix option  Function
 -----------  --------
    -?        List all perfstat options.
    -c        Show system configuration information.
    -e        Search for warnings and errors from
                performance tool status files.
    -f        List size of performance tool status files.
    -p        List active performance tool processes.
    -t        Display last few lines of performance tool
                status files.
    -v        List version strings for performance tool files.
    -z        Dump perfstat info to a file and tar tape.
```

Using the *-c* option, you get information about your system configuration, as shown in the following listing:

```
# perfstat -c

************************************************************
** perfstat for rp-ux6 on Fri May 15 12:20:06 EDT
************************************************************

system configuration information:

uname -a: HP-UX ux6 B.11.00 E 9000/800 71763 8-user license

mounted file systems with disk space shown:
Filesystem          kbytes    used   avail %used Mounted on
/dev/vg00/lvol3      86016   27675   54736   34% /
/dev/vg00/lvol1      67733   44928   16031   74% /stand
/dev/vg00/lvol8     163840   66995   90927   42% /var
/dev/vg00/lvol7     499712  358775  132155   73% /usr
/dev/rp06vgtmp/tmp 4319777 1099297 3134084   26% /tmp
```

```
/dev/vg00/lvol6     270336  188902   76405  71% /opt
/dev/vgroot1/var    640691   15636  605834   3% /newvar
/dev/vgroot1/usr    486677  356866  115210  76% /newusr
/dev/vgroot1/stand   67733   45109   15850  74% /newstand
/dev/vgroot1/root    83733   21181   54178  28% /newroot
/dev/vgroot1/opt    263253  188109   67246  74% /newopt
/dev/vg00/lvol5      20480    1109   18168   6% /home

LAN interfaces:
Name      Mtu      Network      Address      Ipkts     Opkts
lo0       4136     127.0.0.0    localhost    7442      7442
lan0      1500     192.60.11.0  rp-ux6       7847831   12939169


************* (end of perfstat -c output) ****************
```

Using the *-f* option shows the size of the performance tools status files, as shown in the following listing:

```
# perfstat -f

************************************************************
** perfstat for ux6 on Fri May 15 12:20:08 EDT
************************************************************

ls -l list of performance tool status files in /var/opt/perf:

-rw-rw-rw-   1 root  root   7812 May 10 19:35 status.alarmgen
-rw-r--r--   1 root  root      0 May 10 02:40 status.mi
-rw-rw-rw-   1 root  root   3100 May 10 02:40 status.perflbd
-rw-rw-rw-   1 root  root   3978 May 10 02:40 status.rep_server
-rw-r--r--   1 root  root   6079 May 11 23:30 status.scope
-rw-r--r--   1 root  root      0 Mar 31 07:26 status.ttd

************* (end of perfstat -f output) ****************
```

Using the *-v* option displays the version strings for the performance tools running, as shown in the following listing:

```
# perfstat -v

************************************************************
** perfstat for ux6 on Fri May 15 12:20:08 EDT
************************************************************

listing version strings for performance tool files:

NOTE:   The following software version information can be compared
with the version information shown in the /opt/perf/ReleaseNotes
file(s).

MeasureWare executables in the directory /opt/perf/bin
        scopeux  C.01.00     12/17/97 HP-UX 11.0+
            ttd  A.11.00.15  12/15/97 HP-UX 11.00
        perflbd  C.01.00     12/17/97 HP-UX 11.0+
       alarmgen  C.01.00     12/17/97 HP-UX 11.0+
     agdbserver  C.01.00     12/17/97 HP-UX 11.0+
        agsysdb  C.01.00     12/17/97 HP-UX 11.0+
     rep_server  C.01.00     12/17/97 HP-UX 11.0+
        extract  C.01.00     12/17/97 HP-UX 11.0+
        utility  C.01.00     12/17/97 HP-UX 11.0+
            mwa  A.10.52     12/05/97
       perfstat  A.11.01     11/19/97
         dsilog  C.01.00     12/17/97 HP-UX 11.0+
        sdlcomp  C.01.00     12/17/97 HP-UX 11.0+
          sdlexpt  C.01.00     12/17/97 HP-UX 11.0+
       sdlgendata  C.01.00     12/17/97 HP-UX 11.0+
          sdlutil  C.01.00     12/17/97 HP-UX 11.0+

Measureware libraries in the directory /opt/perf/lib
        libmwa.sl  C.01.00     12/17/97 HP-UX 11.0+
        libarm.a   A.11.00.15  12/15/97 HP-UX 11.00
        libarm.sl  A.11.00.15  12/15/97 HP-UX 11.00

Measureware metric description file in the directory /var/opt/perf
        metdesc  C.01.00     12/17/97
```

```
All critical MeasureWare files are accessible

     libnums.sl  B.11.00.15  12/15/97 HP-UX 11.00
     midaemon    B.11.00.15  12/15/97 HP-UX 11.00
         glance  B.11.01     12/16/97 HP-UX 11.00
            gpm B.11.01     12/16/97 HP-UX 11.00

************* (end of perfstat -v output) ***************
```

[Team LiB]

PREVIOUS NEXT



Manual Pages for Commands Used in Chapter 11

There are manual pages in Appendix A for the following commands used in this chapter:

find
grep
iostat
netstat
ps
top
vmstat





Chapter 12. Networking Background

This chapter covers the background of UNIX networking and the Chapter 13 covers networking configuration and using networking commands. If you don't have any UNIX background then you'll want to review this chapter before moving on to the next one. If you have networking background and want to know how to configure UNIX networking and use networking commands then you can jump directly to the Chapter 13.

The following is a list of topics covered in this chapter:

- TCP/IP and IEEE802.3 background
- Internet Protocol (IP) addressing (classes A, B, and C)
- Subnet Mask



An Overview of IEEE802.3, TCP/IP

In order to understand how the networking on your UNIX system works, you first need to understand the components of your network that exist on your UNIX system. Seven layers of network functionality exist on your UNIX system, as shown in Figure 12-1. I cover the bottom four layers at a cursory level so that you can see how each plays a part in the operation of your network and, therefore, be more informed when you configure and troubleshoot networking on your UNIX system. The top layers are the ones that most UNIX system administrators spend time working with because those layers are closest to the functionality to which you can relate. The bottom layers are, however, also important to understand at some level, so that you can perform any configuration necessary to improve the network performance of your system, which has a major impact on the overall performance of your system.

Figure 12-1. ISO/OSI Network Layer Functions

Layer Number	Layer Name	Data Form	Comments
7	Application		User applications here.
6	Presentation		Applications prepared.
5	Session		Applications prepared.
4	Transport	Packet	Port-to-port transportation handled by TCP.
3	Network	Datagram	Internet Protocol (IP) handles routing by going directly to either the destination or default router.
2	Link	Frame	Data encapsulated in Ethernet or IEEE 802.3 with source and destination addresses.
1	Physical		Physical connection between systems. Usually thinnet or twisted pair.

I start reviewing Figure 12-1 at the bottom with layer 1 and then describe each of the four bottom layers. This is the International Standards Organization Open Systems Interconnection (ISO/OSI) model. It is helpful to visualize the way in which networking layers interact.

Physical Layer

The beginning is the physical interconnection between the systems on your network. Without the **physical layer,** you can't communicate between systems, and all the great functionality that you would like to implement is not possible. The physical layer converts the data you would like to transmit to the analog signals that travel along the wire (I'll assume for now that whatever physical layer you have in place uses wires). The information traveling into a network interface is taken off the wire and prepared for use by the next layer.

Link Layer

In order to connect to other systems local to your system, you use the link layer that is able to establish a connection to all the other systems on your local segment. This is the layer where you have either IEEE 802.3 or Ethernet. Your UNIX system supports both of these "encapsulation" methods. This is called

encapsulation because your data is put in one of these two forms (either IEEE 802.3 or Ethernet). Data is transferred at the link layer in frames (just another name for data), with the source and destination addresses and some other information attached. You might think that because two different encapsulation methods exist, they must be very different. This assumption, however, is not the case. IEEE 802.3 and Ethernet are nearly identical. For this reason, many UNIX systems can handle both types of encapsulation. So with the bottom two layers, you have a physical connection between your systems and data that is encapsulated into one of two formats with a source and destination address attached. Figure 12-2 lists the components of an **Ethernet** encapsulation and makes comments about IEEE802.3 encapsulation where appropriate:

Figure 12-2. Ethernet Encapsulation

destination address	6 bytes	address data is sent to
source address	6 bytes	address data is sent from
type	2 bytes	this is the "length count" in 802.3
data	46-1500 bytes	38-1492 bytes for 802.3; the difference in these two data sizes (MTU) can be seen with the **ifconfig** command
crc	4 bytes	checksum to detect errors



One interesting item to note is the difference in the maximum data size between IEEE 802.3 and Ethernet of 1492 and 1500 bytes, respectively. This is the Maximum Transfer Unit (MTU). The **ifconfig** command covered shortly displays the MTU for your interface. The data in Ethernet is called a *frame* (the re-encapsulation of data at the next layer up is called a *datagram* in IP, and encapsulation at two levels up is called a *packet* for TCP).

Keep in mind that Ethernet and IEEE 802.3 will run on the same physical connection, but there are indeed differences between the two encapsulation methods. With your UNIX systems, you don't have to spend much, if any, time setting up your network interface for encapsulation.

Network Layer



Next we work up to the third layer, which is the network layer. This layer on UNIX systems is synonymous with the Internet Protocol (IP). Data at this layer is transported as *datagrams*. This is the layer that handles the routing of data around the network. Data that gets routed with IP sometimes encounters an error of some type, which is reported back to the source system with an Internet Control Message Protocol (ICMP) message. We will see some ICMP messages shortly. **ifconfig** and **netstat** are two UNIX commands that are commonly used to configure this routing.

Unfortunately, the information that IP uses does not conveniently fit inside an Ethernet frame, so you end up with fragmented data. This is really re-encapsulation of the data, so you end up with a lot of inefficiency as you work your way up the layers.

IP handles routing in a simple fashion. If data is sent to a destination connected directly to your system, then the data is sent directly to that system. If, on the other hand, the destination is not connected directly to your system, the data is sent to the default router. The default router then has the responsibility of getting the data to its destination. This routing can be a little tricky to understand, so I'll cover it in detail shortly.

Transport Layer

The *trasport level* is the next level up from the network layer. It communicates with *ports*. TCP is the most common protocol found at this level, and it forms packets that are sent from port to port. The port used by

a program is usually defined in **/etc/services**, along with the protocol (such as TCP). These ports are used by network programs such as **telnet, rlogin, ftp**, and so on. You can see that these programs, associated with ports, are the highest level we have covered while analyzing the layer diagram.

[Team LiB]

PREVIOUS NEXT



Internet Protocol (IP) Addressing and Subnet Mask

The Internet Protocol address (IP address) is either a class "A," "B," or "C" address (there are also class "D" and "E" addresses I will not cover). A class "A" network supports many more nodes per network than either a class "B" or "C" network. IP addresses consist of four fields. The purpose of breaking down the IP address into four fields is to define a node (or host) address and a network address. Figure 12-3 summarizes the relationships between the classes and addresses.

Figure 12-3. Comparison of Internet Protocol (IP) Addresses

Address Class	Networks	Nodes per Network	Bits Defining Network	Bits Defining Nodes per Network
A	a few	the most	8 bits	24 bits
B	many	many	16 bits	16 bits
C	the most	a few	24 bits	8 bits
Reserved	-	-	-	-

These bit patterns are significant in that the number of bits defines the ranges of networks and nodes in each class. For instance, a class A address uses 8 bits to define networks, and a class C address uses 24 bits to define networks. A class A address therefore supports fewer networks than a class C address. A class A address, however, supports many more nodes per network than a class C address. Taking these relationships one step further, we can now view the specific parameters associated with these address classes in Figure 12-4.

Figure 12-4. Address Classes

Address Class	Networks Supported	Nodes per Network	Address Range		
A	127	16777215	0.0.0.1	-	127.255.255.254
B	16383	65535	128.0.0.1	-	191.255.255.254
C	2097157	255	192.0.0.1	-	223.255.254.254
Reserved	-	-	224.0.0.0	-	255.255.255.255

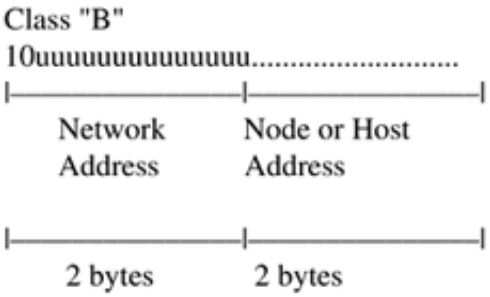
Looking at the 32-bit address in binary form, you can see how to determine the class of an address:

Class "A"
0uuuuuuu.....
|————————|————————————————|
Network Address Node or Host Address

|————————|————————————————|
1 byte 3 bytes

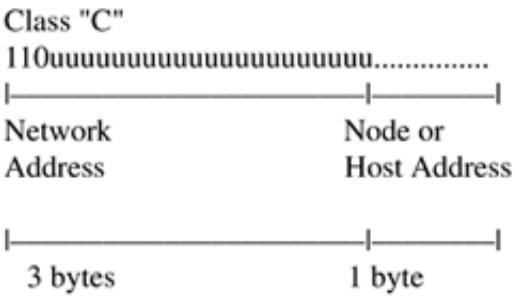
net.host.host.host

A class “A” address has the first bit set to 0. You can see how so many nodes per network can be supported with all the bits devoted to the node or host address. The first bit of a class A address is 0, and the remaining 7 bits of the network portion are used to define the network. Then a total of 3 bytes are devoted to defining the nodes with a network.



net.net.host.host

A class “B” address has the first bit set to a 1 and the second bit to a 0. More networks are supported here than with a class A address, but fewer nodes per network. With a class B address, 2 bytes are devoted to the network portion of the address and 2 bytes devoted to the node portion of the address.



net.net.net.host

A class “C” address has the first bit and second bit set to 1, and the third bit is 0. The greatest number of networks and fewest number of nodes per network are associated with a class C address. With a class C address, 3 bytes are devoted to the network and 1 byte is devoted to the nodes within a network.

These addresses are used in various setup files that are covered later when the **/etc/hosts** file is described. Every interface on your network must have a unique IP address. Systems that have two network interfaces must have two unique IP addresses.

Subnet Mask

Your UNIX system uses the subnet mask to determine whether an IP datagram is for a host on its own subnet, a host on a different subnet but the same network, or a host on a different network. Using subnets, you can have some hosts on one subnet and other hosts on a different subnet. The subnets can be separated by routers or other networking electronics that connect the subnets.

To perform routing, the only aspects of an address that your router uses are the net and subnet. The subnet mask is used to mask the host part of the address. Because you can set up network addresses in such a way that you are the only one who knows which part of the address is the host, subnet, and network, you use the subnet mask to make your system aware of the bits of your IP address that are for the host and which are for the subnet.

In its simplest form, what you are really doing with subnet masking is specifying which portion of your IP address defines the host, and which part defines the network. One of the most confusing aspects of working with subnet masks is that most books show the subnet masks in Figure 12-5 as the most common.

Figure 12-5. Subnet Masks

Address Class	Decimal	Hex
A	255.0.0.0	0xff000000
B	255.255.0.0	0xffff0000
C	255.255.255.0	0xffffff00

This way of thinking, however, assumes that you are devoting as many bits as possible to the network and as many bits as possible to the host, and that no subnets are used. Figure 12-6 shows an example of using subnetting with a class B address.

Figure 12-6. Class B IP Address and Subnet Mask Example

Address Class	Class B		
host IP address	152.128.	12.	1
breakdown	network	subnet	hostid
number of bits	16 bits	8 bits	8 bits
subnet mask in decimal	255.255.	255.	0
subnet mask in hexadecimal	0xffffff00		
Example of different host on same subnet	152.128.	12.	2
Example of host on different subnet	152.128.	13.	1

In Figure 12-6, the first two bytes of the subnet mask (255.255) define the network, the third byte (255) defines the subnet, and the fourth byte (0) is devoted to the host ID. Although this subnet mask for a class B address did not appear in the earlier default subnet mask figure, the subnet mask of 255.255.255.0 is widely used in class B networks to support subnetting.

How does your UNIX system perform the comparison using the subnet mask of 255.255.255.0 to determine that 152.128.12.1 and 152.128.13.1 are on different subnets? Figure 12-7 shows this comparison.

Figure 12-7. Example of Using Subnet Mask to Compare Addresses

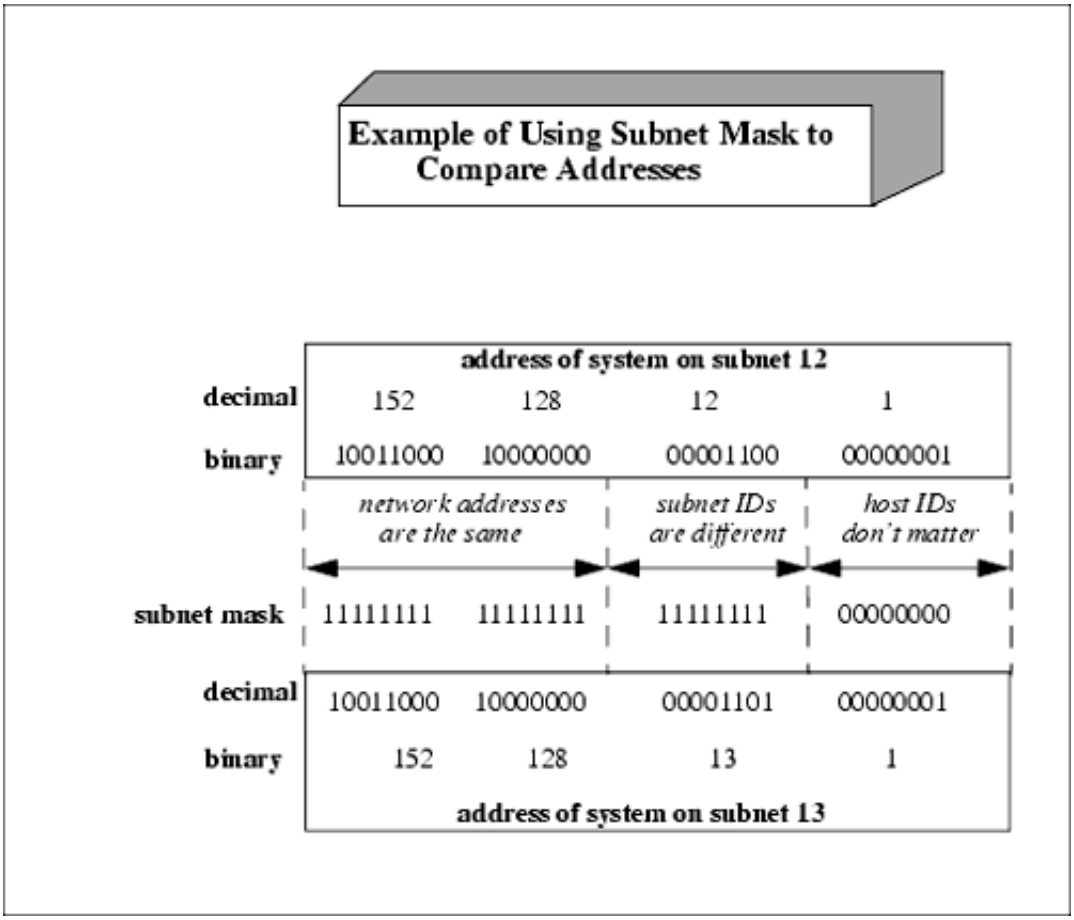
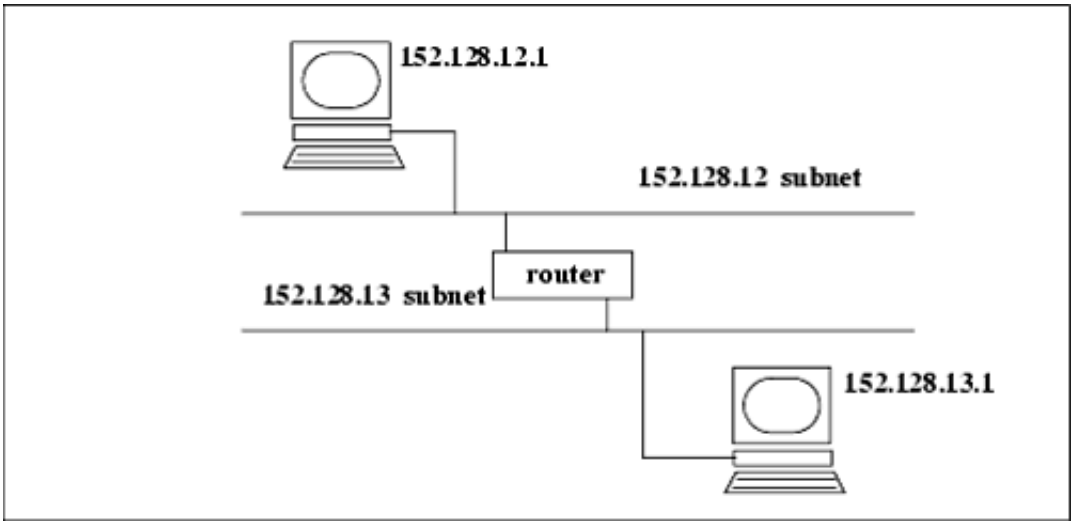


Figure 12-8 shows these two systems on the different subnets:

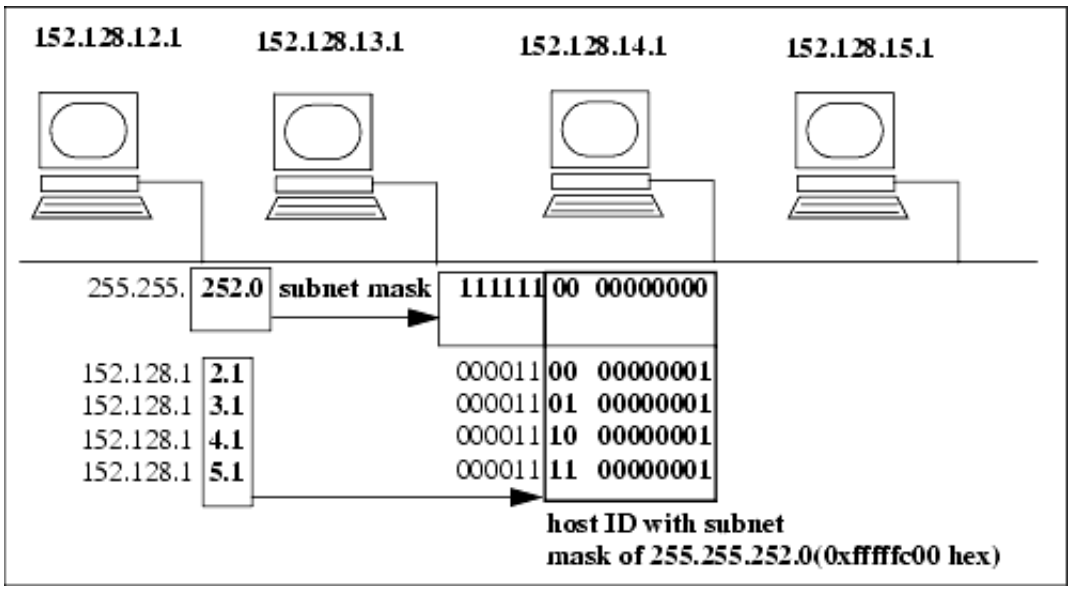
Figure 12-8. Class B Systems on Different Subnets



You don't have to use the 8-bit boundaries to delineate the network, subnet, and host ID fields. If, for instance, you want to use part of the subnet field for the host ID, you can do so. A good reason for this

approach would be to accommodate future expandability. You might want subnets 12, 13, 14, and 15 to be part of the same subnet today and make these into separate subnets in the future. Figure 12-9 shows this setup:

Figure 12-9. Future Expandability Using Subnet Mask



These systems are connected to the same subnet, even though part of the third byte, normally associated with the subnet, is used for the host ID. In the future, the subnet mask could be changed to 255.255.252.0 and have four separate subnets of 12, 13, 14, and 15. This arrangement would require putting routers in place to route to these separate subnets.

Chapter 13 covers configuring and using UNIX networking.



Chapter 13. Configuring and Using Networking

In this chapter we'll configure LAN cards and use a variety of networking commands. If you don't know the basics of UNIX networking please see the previous that covered UNIX networking background. In this chapter we'll focus on identifying and configuring LAN cards and use of many networking commands. The following is a bullet list of the topics we'll cover in this chapter:

- Configuring LAN Adapters (Identify LAN adapters, use **set_parms** to configure, and manually configure)
- Commonly Used Networking Commands such as **ping, netstat, route, ifconfig** and others.
- ARPA Services
- Berkeley commands
- Host name mapping
- DNS Setup
- Network File System (NFS) background

In the next section we cover LAN card configuration.



Configuring Networking and LAN Adapters

We'll cover several topics in this section. First we'll identify the LAN cards in our system. Next we'll use **set_parms** to configure the first LAN card and view all of the configuration files that were update. Then we'll manually configure the second LAN card.

Identify LAN Cards in System

The first step to take when configuring networking is to identify the networking cards in your HP 9000. You can have multiple LAN cards in a system and configure multiple IP addresses on the same LAN card if you wish. The following **ioscan** listing shows three LAN cards in a server:

[View full width]

```
# ioscan -funC lan
Class     I  H/W Path     Driver   S/W State   H/W Type     Description
============================================================================
lan       0  0/0/0/0      btlan    CLAIMED     INTERFACE    HP PCI 10/100Base-TX Core
                         /dev/diag/lan0  /dev/ether0     /dev/lan0
lan     1  0/4/0/0/6/0  btlan  CLAIMED     INTERFACE    HP A5838A PCI 100Base-TX/SCSI
➥ COMBO
                         /dev/diag/lan1  /dev/ether1     /dev/lan1
lan     2  0/4/0/0/7/0  btlan  CLAIMED     INTERFACE    HP A5838A PCI 100Base-TX/SCSI
➥ COMBO
                         /dev/diag/lan2  /dev/ether2     /dev/lan2
#
```

This output shows a LAN interface on the Core I/O card at hardware path *0/0/0/0*, and two cards that are a combination SCSI and LAN card at hardware paths *0/4/0/0/6/0* and *0/4/0/0/7/0*. These three interfaces are *lan0, lan1*, and *lan3*. All of these cards use the *btlan* driver. We can confirm that the *btlan* driver and other LAN drivers have been configured into the kernel for LAN interfaces with the **lsdev** command as shown in the following listing:

```
# lsdev -C lan
    Character     Block       Driver          Class
       14          -1         btlan           lan
       15          -1         maclan          lan
       45          -1         fcT1_cntl       lan
       50          -1         fddi4           lan
#
```

These drivers are listed by class (*-C*) option for the currently running kernel. Note that *btlan* is used for the driver of *lan0-2* in the earlier **ioscan** listing.

Use set_parms to Configure First LAN Interface

In this section we'll configure the first LAN card, *lan0*, immediately after HP-UX 11i has been loaded and the system reboots. The dialogue that takes place in the upcoming example is similar to what you'd see if you were to run **set_parms initial** to initially configure a LAN card. I'll skip over some of the dialogue such as setting the date and time and provide only the LAN-related information:

```
_______________________________________________________________________________
                        Welcome to HP-UX!

Before using your system, you will need to answer a few questions.

The first question is whether you plan to use this system on a network.

Answer "yes" if you have connected the system to a network and are ready
to link with a network.

Answer "no" if you:
```

```
     * Plan to set up this system as a standalone (no networking).

     * Want to use the system now as a standalone and connect to a
       network later.
```

```
Are you ready to link this system to a network?

Press [y] for yes or [n] for no, then press [Enter] y
```

```
If your network contains a DHCP (Dynamic Host Configuration
Protocol) server, you may want to use the DHCP server to
supply your networking setup information.

It is best to answer "Yes" to the following question only if
you're sure that DHCP service is available.
```

```
Do you wish to use DHCP to obtain networking information?

Press [y] for yes or [n] for no, then press [Enter] n
```

```
   Before you begin using this system, you need to obtain the
   following information from your local network administrator:

      * Your system name (host name).

      * Your Internet Protocol (IP) address.

      * Your time zone.

    If you do not have this information, you may stop now and restart
    your system once you have it.
```

```
                    .

                    .

                    .


Enter the system name, then press [Enter].  Just pressing [Enter] will
keep the (not recommended) name "unknown": asodevlab1

The hostname (or first component of the hostname) "asodevlab1"
contains more than 8 characters.  This is valid, but the system
name as reported by `uname' will be truncated to "asodevla".

Press [Enter] to continue...

You have chosen asodevlab1 as the name for this system.
Is this correct?

Press [y] for yes or [n] for no, then press [Enter] y
```

```
                    .

                    .

                    .
```

```
  Before setting an IP address or using DHCP, please choose which
  network interface card to use from the list below.  The card should
  now be physically connected to the network.  Other network interfaces
  may be configured using SAM after the system has booted.  A * indicates
  the lan that was used for installing HP-UX, if known.

    Interface   Hardware
    Name        Path             Type
    --------------------------------
    lan0        0/0/0/0          HP PCI 10/100Base-TX Core
    lan1        0/4/0/0/6/0      HP A5838A PCI 100Base-TX/SCSI COMBO
    lan2        0/4/0/0/7/0      HP A5838A PCI 100Base-TX/SCSI COMBO

_____________________________________________________________________________

 Enter the name of the lan interface you would like to configure followed by
   [Enter] or just press [Enter] to select the default interface (lan0): lan0

You have chosen lan0 as the lan interface to configure.
Is this correct?

Press [y] for yes or [n] for no, then press [Enter] y

_____________________________________________________________________________

Enter your IP address, then press [Enter] or just press [Enter] to select
the default address (127.0.0.1): 15.32.162.130

You have chosen 15.32.162.130 as the IP address for this system.
Is this correct?

Press [y] for yes, [n] for no or [c] to cancel then press [Enter] y

_____________________________________________________________________________

                    .

                    .

                    .


The subnetwork mask shown above is the system default value and possibly
should be changed.  Enter the new subnetwork mask, then press [Enter]
or just press [Enter] to use the default value above: 255.255.248.0


                    .

                    .

                    .
_____________________________________________________________________________

Enter the gateway address, then press [Enter] or just press [Enter]
to select the current gateway address (15.32.162.130): 15.32.160.1




_____________________________________________________________________________

  Additional Network Parameters: Subnetwork Mask and Default Gateway

  Enter the subnetwork mask and default gateway information.

  Example:

      Subnetwork mask:    255.255.255.0
      Gateway IP address: 15.32.162.1

  Current Settings:
```

```
      Subnetwork mask:    255.255.248.0
      Gateway IP address: 15.32.160.1
_______________________________________________________________________________

Are the parameters above correct?

Press [y] for yes, [n] for no or [c] to cancel then press [Enter] y
_______________________________________________________________________________

Note: Your system appears to have 3 network interfaces installed.
This procedure only configures the default network interface.
Use SAM to configure additional network interface cards.
_______________________________________________________________________________


Press [Enter] to continue...



                  .

                  .

                  .

#
```

We entered a lot of information during this initial setup that is similar to using **set_parms**. We did not us DHCP but instead, entered *hostname, IP address, subnet mask, default gateway*, and others.

You can run **set_parms_initial** to modify this information at any time. Let's look at some of the files that were updated and run some commands to see that networking-related updates were made. The first file we'll view is **/etc/rc.config.d/netconf**:

```
# cat /etc/rc.config.d/netconf
# netconf:  configuration values for core networking subsystems
#
# @(#)B.11.11_LR $Revision: 1.6.119.6 $ $Date: 97/09/10 15:56:01 $
#
# HOSTNAME:           Name of your system for uname -S and hostname
#
# OPERATING_SYSTEM:   Name of operating system returned by uname -s
#                     ---- DO NOT CHANGE THIS VALUE ----
#
# LOOPBACK_ADDRESS:   Loopback address
#                     ---- DO NOT CHANGE THIS VALUE ----
#
# IMPORTANT:  for 9.x-to-10.0 transition, do not put blank lines between
# the next set of statements

HOSTNAME="asodevlab1"
OPERATING_SYSTEM=HP-UX
LOOPBACK_ADDRESS=127.0.0.1

# Internet configuration parameters.  See ifconfig(1m), autopush(1m)
#
# INTERFACE_NAME:     Network interface name (see lanscan(1m))
#
# IP_ADDRESS:         Hostname (in /etc/hosts) or IP address in decimal-dot
#                     notation (e.g., 192.1.2.3)
#
# SUBNET_MASK:        Subnetwork mask in decimal-dot notation, if different
#                     from default
#
# BROADCAST_ADDRESS:  Broadcast address in decimal-dot notation, if
#                     different from default
#
# INTERFACE_STATE:    Desired interface state at boot time.
#                     either up or down, default is up.
#
# DHCP_ENABLE         Determines whether or not DHCP client functionality
#                     will be enabled on the network interface (see
#                     auto_parms(1M), dhcpclient(1M)). DHCP clients get
```

```
#                     their IP address assignments from DHCP servers.
#                     1 enables DHCP client functionality; 0 disables it.
#
# For each additional network interfaces, add a set of variable assignments
# like the ones below, changing the index to "[1]", "[2]" et cetera.
#
# IMPORTANT:  for 9.x-to-10.0 transition, do not put blank lines between
# the next set of statements

INTERFACE_NAME[0]=lan0
IP_ADDRESS[0]=15.32.162.130
SUBNET_MASK[0]=255.255.248.0
BROADCAST_ADDRESS[0]=""
INTERFACE_STATE[0]=""
DHCP_ENABLE[0]=0

# Internet routing configuration.  See route(1m), routing(7)
#
# ROUTE_DESTINATION:  Destination hostname (in /etc/hosts) or host or network
#                     IP address in decimal-dot notation, preceded by the word
#                     "host" or "net"; or simply the word "default".
#
# ROUTE_MASK:         Subnetwork mask in decimal-dot notation, or C language
#                     hexadecimal notation.  This is an optional field.
#                     A IP address, subnet mask pair uniquely identifies
#                     a subnet to be reached. If a subnet mask is not given,
#                     then the system will assign the longest subnet mask
#                     of the configured network interfaces to this route.
#                     If there is no matching subnet mask, then the system
#                     will assign the default network mask as the route's
#                     subnet mask.
#
# ROUTE_GATEWAY:      Gateway hostname (in /etc/hosts) or IP address in
#                     decimal-dot notation.  If local interface, must use the
#                     same form as used for IP_ADDRESS above (hostname or
#                     decimal-dot notation). If loopback interface, i.e.,
#             127.0.0.1, the ROUTE_COUNT must be set to zero.
#
# ROUTE_COUNT:        An integer that indicates whether the gateway is a
#                     remote interface (one) or the local interface (zero)
#             or loopback interface (e.g., 127.*).
#
# ROUTE_ARGS:         Route command arguments and options.  This variable
#                     may contain a combination of the following arguments:
#                     "-f", "-n" and "-p pmtu".
#
# For each additional route, add a set of variable assignments like the ones
# below, changing the index to "[1]", "[2]" et cetera.
#
# IMPORTANT:  for 9.x-to-10.0 transition, do not put blank lines between
# the next set of statements

# ROUTE_DESTINATION[0]=default
# ROUTE_MASK[0]=""
# ROUTE_GATEWAY[0]=""
# ROUTE_COUNT[0]=""
# ROUTE_ARGS[0]=""

# Dynamic routing daemon configuration.  See gated(1m)
#
# GATED:        Set to 1 to start gated daemon.
# GATED_ARGS:   Arguments to the gated daemon.
GATED=0
GATED_ARGS=""

#
# Router Discover Protocol daemon configuration.  See rdpd(1m)
#
# RDPD:         Set to 1 to start rdpd daemon
#

RDPD=0

#
# Reverse ARP daemon configuration.  See rarpd(1m)
```

```
#
# RARP:         Set to 1 to start rarpd daemon
#

RARP=0


# The NODENAME variable (limited to 8 chars) will be used to set the
# UUCP name (uname -S), since HOSTNAME is longer than 8 chars.
NODENAME=asodevla
ROUTE_GATEWAY[0]=15.32.160.1
ROUTE_COUNT[0]=1
ROUTE_DESTINATION[0]=default
#
```

All information related to *lan0* that we entered initially is present in this file. Note that variables related to *lan0* use a *[0],* which is significant in the next section when we manually add to this file information related to *lan1* and use *[1]*.

The **/etc/hosts** file (see the **/etc/hosts** overview later in this chapter) was also automatically updated to include the hostname and address we entered earlier. This file should be updated to include systems to which you'll connect as you'll see in the next section. The following is the **/etc/hosts** file for our system:

```
# cat /etc/hosts
# @(#)B.11.11_LRhosts $Revision: 1.9.214.1 $
# The form for each entry is:
# <internet address>    <official hostname> <aliases>
#
# For example:
# 192.1.2.34    hpfcrm  loghost
#
# See the hosts(4) manual page for more information.
# Note: The entries cannot be preceded by a space.
#   The format described in this file is the correct format.
#       The original Berkeley manual page contains an error in
#   the format description.
#

15.32.162.130   asodevlab1
127.0.0.1   localhost   loopback
#
```

The *hostname* of *asodevlab1* appears in **/etc/hosts**.

Let's now run form 2 of the **ifconfig** command to view parameters related to *lan0*:

```
#    ifconfig lan0
lan0: flags=843<UP,BROADCAST,RUNNING,MULTICAST>
    inet 15.32.162.130 netmask fffff800 broadcast 15.32.167.255
#
```

This **ifconfig** output shows the IP address we entered for *lan0* of *15.32.162.130*, and the subnet mask we entered, in hex form, of *255.255.248.0*. The interface is *UP* so it is operational.

We have not confirmed that the *default gateway* we specified. The following **netstat -rn** command is used to show *r*outing in *n*umber format for *lan0*:

```
# netstat -rn
Routing tables
Destination          Gateway           Flags   Refs Interface  Pmtu
15.32.162.130        15.32.162.130      UH       0  lan0       4136
127.0.0.1            127.0.0.1          UH       0  lo0        4136
15.32.160.0          15.32.162.130      U        2  lan0       1500
127.0.0.0            127.0.0.1          U        0  lo0           0
default              15.32.160.1        UG       0  lan0          0
#
```

The *default* route we specified of *15.32.160.1* is shown in the last line of this output. Any IP addresses to which we connect that are not on our subnet will use this *default* route.

The gateway we had specified as the default route in the **netconf** file of *15.32.160.1* is shown as the default gateway in this **netstat** output. Routes can be added and deleted with the **route** command (see the **route** overview later in this chapter.)

Using **set_parms**, set up a lot of host and networking-related information for us. In the next section we'll manually expand these files to include information related to the second networking interface.

Manually Configure Second LAN Interface

In the previous section we used **set_parms** to setup the first LAN card in our system, as well has hostname and other parameters, and in this section we'll perform the same work manually for the second LAN card.

The information we added to **netconf** includes the IP address, subnet mask, broadcast address, gateway, and all information related to *lan0. [0]* identifies entries made in the file for *lan0*.

The first step we'll take is to expand **/etc/rc.config.d/netconf** to include variable information for our second LAN card. We'll add the following block of information to **netconf**:

```
INTERFACE_NAME[1]=lan1
IP_ADDRESS[1]=15.32.162.131
SUBNET_MASK[1]=255.255.248.0
BROADCAST_ADDRESS[1]=""
INTERFACE_STATE[1]=""
DHCP_ENABLE[1]=0
```

The following is a full listing of **netconf** with our *lan1* block of information included:

```
# netconf:  configuration values for core networking subsystems
#
# @(#)B.11.11_LR $Revision: 1.6.119.6 $ $Date: 97/09/10 15:56:01 $
#
# HOSTNAME:           Name of your system for uname -S and hostname
#
# OPERATING_SYSTEM:   Name of operating system returned by uname -s
#                     ---- DO NOT CHANGE THIS VALUE ----
#
# LOOPBACK_ADDRESS:   Loopback address
#                     ---- DO NOT CHANGE THIS VALUE ----
#
# IMPORTANT:  for 9.x-to-10.0 transition, do not put blank lines between
# the next set of statements

HOSTNAME="asodevlab1"
OPERATING_SYSTEM=HP-UX
LOOPBACK_ADDRESS=127.0.0.1

# Internet configuration parameters.  See ifconfig(1m), autopush(1m)
#
# INTERFACE_NAME:     Network interface name (see lanscan(1m))
#
# IP_ADDRESS:         Hostname (in /etc/hosts) or IP address in decimal-dot
#                     notation (e.g., 192.1.2.3)
#
# SUBNET_MASK:        Subnetwork mask in decimal-dot notation, if different
#                     from default
#
# BROADCAST_ADDRESS:  Broadcast address in decimal-dot notation, if
#                     different from default
#
# INTERFACE_STATE:    Desired interface state at boot time.
#                     either up or down, default is up.
#
# DHCP_ENABLE         Determines whether or not DHCP client functionality
#                     will be enabled on the network interface (see
#                     auto_parms(1M), dhcpclient(1M)). DHCP clients get
#                     their IP address assignments from DHCP servers.
#                     1 enables DHCP client functionality; 0 disables it.
#
# For each additional network interfaces, add a set of variable assignments
# like the ones below, changing the index to "[1]", "[2]" et cetera.
#
# IMPORTANT:  for 9.x-to-10.0 transition, do not put blank lines between
# the next set of statements

INTERFACE_NAME[0]=lan0
IP_ADDRESS[0]=15.32.162.130
```

```
SUBNET_MASK[0]=255.255.248.0
BROADCAST_ADDRESS[0]=""
INTERFACE_STATE[0]=""
DHCP_ENABLE[0]=0

INTERFACE_NAME[1]=lan1
IP_ADDRESS[1]=15.32.162.131
SUBNET_MASK[1]=255.255.248.0
BROADCAST_ADDRESS[1]=""
INTERFACE_STATE[1]=""
DHCP_ENABLE[1]=0

# Internet routing configuration.  See route(1m), routing(7)
#
# ROUTE_DESTINATION:  Destination hostname (in /etc/hosts) or host or network
#                     IP address in decimal-dot notation, preceded by the word
#                     "host" or "net"; or simply the word "default".
#
# ROUTE_MASK:         Subnetwork mask in decimal-dot notation, or C language
#                     hexadecimal notation.  This is an optional field.
#                     A IP address, subnet mask pair uniquely identifies
#                     a subnet to be reached. If a subnet mask is not given,
#                     then the system will assign the longest subnet mask
#                     of the configured network interfaces to this route.
#                     If there is no matching subnet mask, then the system
#                     will assign the default network mask as the route's
#                     subnet mask.
#
# ROUTE_GATEWAY:      Gateway hostname (in /etc/hosts) or IP address in
#                     decimal-dot notation.  If local interface, must use the
#                     same form as used for IP_ADDRESS above (hostname or
#                     decimal-dot notation). If loopback interface, i.e.,
#                     127.0.0.1, the ROUTE_COUNT must be set to zero.
#
# ROUTE_COUNT:        An integer that indicates whether the gateway is a
#                     remote interface (one) or the local interface (zero)
#                     or loopback interface (e.g., 127.*).
#
# ROUTE_ARGS:         Route command arguments and options.  This variable
#                     may contain a combination of the following arguments:
#                     "-f", "-n" and "-p pmtu".
#
# For each additional route, add a set of variable assignments like the ones
# below, changing the index to "[1]", "[2]" et cetera.
#
# IMPORTANT:  for 9.x-to-10.0 transition, do not put blank lines between
# the next set of statements

# ROUTE_DESTINATION[0]=default
# ROUTE_MASK[0]=""
# ROUTE_GATEWAY[0]=""
# ROUTE_COUNT[0]=""
# ROUTE_ARGS[0]=""

# Dynamic routing daemon configuration.  See gated(1m)
#
# GATED:        Set to 1 to start gated daemon.
# GATED_ARGS:   Arguments to the gated daemon.

GATED=0
GATED_ARGS=""

#
# Router Discover Protocol daemon configuration.  See rdpd(1m)
#
# RDPD:         Set to 1 to start rdpd daemon
#

RDPD=0

#
# Reverse ARP daemon configuration.  See rarpd(1m)
#
# RARP:         Set to 1 to start rarpd daemon
#
```

```
RARP=0


# The NODENAME variable (limited to 8 chars) will be used to set the
# UUCP name (uname -S), since HOSTNAME is longer than 8 chars.
NODENAME=asodevla
ROUTE_GATEWAY[0]=15.32.160.1
ROUTE_COUNT[0]=1
ROUTE_DESTINATION[0]=default
#
```

This file has now been expanded to include information for *lan1* in the lines shown with *[1]*.

Similarly, the **/etc/hosts** file was expanded to include the host name and IP address of the second LAN card as shown in the following listing:

```
# cat /etc/hosts
# @(#)B.11.11_LRhosts $Revision: 1.9.214.1 $ $Date: 96/10/08 13:20:01 $
#
# The form for each entry is:
# <internet address>    <official hostname> <aliases>
#
# For example:
# 192.1.2.34    hpfcrm  loghost
#
# See the hosts(4) manual page for more information.
# Note: The entries cannot be preceded by a space.
#       The format described in this file is the correct format.
#       The original Berkeley manual page contains an error in
#       the format description.
#

15.32.162.130   asodevlab1      lab1
15.32.162.131   asodevlab2      lab2
127.0.0.1       localhost       loopback
#
```

The second LAN card and its IP address were added to the file as well as aliases for the two systems of *lab1* and *lab2*.

We can make these entries take effect by running **/sbin/init.d/net stop** and then **/sbin/init.d/net start** to reread the networking files, and then run **ifconfig** and **netstat -rn** to see if our additions were made for *lan1,* as shown in the following output:

```
# /sbin/init.d/net stop
# /sbin/init.d/net start

# ifconfig lan0
lan0: flags=843<UP,BROADCAST,RUNNING,MULTICAST>
        inet 15.32.162.130 netmask fffff800 broadcast 15.32.167.255

# ifconfig lan1
lan1: flags=843<UP,BROADCAST,RUNNING,MULTICAST>
        inet 15.32.162.131 netmask fffff800 broadcast 15.32.167.255
#

# netstat -rn
Routing tables
Destination           Gateway           Flags   Refs Interface  Pmtu
15.32.162.130         15.32.162.130     UH        0  lan0       4136
127.0.0.1             127.0.0.1         UH        0  lo0        4136
15.32.162.131         15.32.162.131     UH        0  lan1       4136
15.32.160.0           15.32.162.131     U         2  lan1       1500
15.32.160.0           15.32.162.130     U         2  lan0       1500
127.0.0.0             127.0.0.1         U         0  lo0           0
default               15.32.160.1       UG        0  lan0          0
#
```

The output of the two **ifconfig** commands shows that the two LAN cards have been configured. The gateway we had specified as the default route in the **netconf** file of *15.32.160.1* is shown as the default gateway in this **netstat** output. Routes can be added and deleted with the **route** command (see the **route** overview later in this chapter.) We'll use these interfaces for more advanced work in other chapters. As a test to ensure that these interfaces worked I was able to **ping**, as shown in the example below from

another server to *asodevlab2*, as well as to initiate a **telnet** session.

```
# ping 15.32.160.1 4096 10
PING 15.32.160.1: 4096 byte packets
4096 bytes from 15.32.160.1: icmp_seq=0. time=11. ms
4096 bytes from 15.32.160.1: icmp_seq=1. time=11. ms
4096 bytes from 15.32.160.1: icmp_seq=2. time=11. ms
4096 bytes from 15.32.160.1: icmp_seq=3. time=11. ms
4096 bytes from 15.32.160.1: icmp_seq=4. time=11. ms
4096 bytes from 15.32.160.1: icmp_seq=5. time=11. ms
4096 bytes from 15.32.160.1: icmp_seq=6. time=11. ms
4096 bytes from 15.32.160.1: icmp_seq=7. time=13. ms
4096 bytes from 15.32.160.1: icmp_seq=8. time=11. ms
4096 bytes from 15.32.160.1: icmp_seq=9. time=11. ms

----15.32.160.1 PING Statistics----
10 packets transmitted, 10 packets received, 0% packet loss
round-trip (ms)  min/avg/max = 11/11/13
#
```

We were able to **ping** from another server to asodev*lab2* in this example using a packet size of *4096* for *10* iterations.

We could also have configured *lan1* at the command line with **ifconfig** (see the **ifconfig** overview later in this chapter). **ifconfig** can be used to configure a LAN interface or to display information related to a LAN interface configuration. The following example shows running **ifconfig** to configure *lan1*:

```
# /etc/ifconfig lan1 inet 15.32.162.131 netmask 255.255.248.0
```

This is the first form of the **ifconfig** command in which LAN card parameters are specified.

As an added piece of information, make sure that you do not have duplicate files in **/etc/rc.config.d** such as **netconf** and **netconf.old**. The system isn't smart enough and will read all files and get confused. The work around is to reverse naming convention. For example, **netconf.old** needs to be **old.netconf**. If you do have duplicate files in this directory, you will need to reboot after changing the naming convention.

We have covered manually configuring a LAN card and the files related to doing so in this section. We have a third LAN card in this system as well that we could configure with **set_parms** or manually by performing the steps in this section.

The next section covers some frequently used networking commands including many of those we used in this section.

[Team LiB]

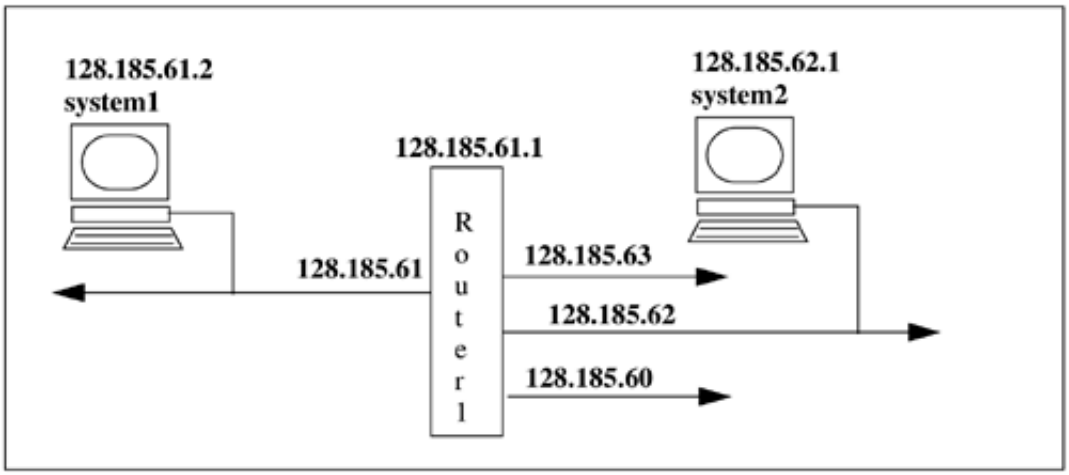
PREVIOUS NEXT

Commonly Used Networking Commands

Setting up a network is an intensive planning exercise for both network and system administrators. No two networking environments are alike. There is typically a lot of networking electronics to which your system is connected. There are many useful commands related to testing connectivity to other systems and networking configuration. Should you encounter a problem, you want to have an understanding of some networking commands that can be lifesavers. In addition, you can encounter some tricky aspects to networking setup if you have some networking hardware that your UNIX systems must interface to, such as routers, gateways, bridges, and so on. I give an example of one such case: connecting a UNIX system to a router. At the same time, I cover some of the most handy networking commands as part of this description. I also give examples of some commands on a variety of UNIX variants because many installations run more than one variant of UNIX.

Consider Figure 13-1, in which a UNIX system is connected directly to a router.

Figure 13-1. UNIX System and Router Example



Here we have a UNIX system connected to segment 128.185.61. This is a class "B" Internet address with subnetting enabled.

The **/etc/hosts** file needs to have in it the UNIX system with node ID 2, the router, and any other systems on this segment or segments on the other side of the router.

If the router is properly configured, we should be able to seamlessly connect from 61 to systems on segments 60, 62, and 63. The router should be configured to allow our system to connect to systems on other segments (60, 62, and 63) by going through the router. Some unforeseen configuration was required to make this simple network operate seamlessly. In this case, a problem occurred getting system1 to connect to systems on the other side of the router on 60, 62, and 63. Before discussing the additional configuration that needed to be done, I first show the **/etc/hosts** file and then use some very useful UNIX commands that show the state of the network. Here is the **/etc/hosts** file showing just the UNIX system and router:

```
$ cat /etc/hosts
127.0.0.1    localhosts loopback
128.185.61.1  router1# router
128.185.61.2  system1# UNIX system on 61
128.185.62.1  system2# UNIX system on 62
```

This host file is simple and allows system1 to connect to router1 and system2. The connection from system1 to system2 is accomplished by going through the router.

ping

Let's look at one of the most commonly used networking commands - **ping**. This command is used to determine whether or not a connection exists between two networking components. **ping** is a simple command that sends an ICMP echo packet to the host you specify once per second. You may recall that ICMP was covered earlier under the network, or third layer. **ping** stands for Packet InterNet Groper. **ping** differs somewhat among UNIX variants, mostly in the reporting that **ping** produces when no options are provided.

Some systems provide performance information when **ping** is issued with no options; others report that the system "is alive". The following is an example of checking the connection between the local system and another system on the network called *austin*:

```
martyp $ ping austin
austin is alive
martyp $
```

You can adjust the packet size and number of iterations on most UNIX variants as in the HP-UX example shown below, specifying a packet size of *4096* and interval of *5*:

```
# ping l2 4096 5
PING l2: 4096 byte packets
4096 bytes from 10.1.1.12: icmp_seq=0. time=2. ms
4096 bytes from 10.1.1.12: icmp_seq=1. time=2. ms
4096 bytes from 10.1.1.12: icmp_seq=2. time=2. ms
4096 bytes from 10.1.1.12: icmp_seq=3. time=2. ms
4096 bytes from 10.1.1.12: icmp_seq=4. time=2. ms

----l2 PING Statistics----
5 packets transmitted, 5 packets received, 0% packet loss
round-trip (ms)  min/avg/max = 2/2/2
#
```

AIX allows you to specify the interval with *-I* as well as other options, including packet size, and number of iterations. These options are shown for an AIX system in the following example:

```
martyp $ ping -I 5 austin 4096 10
PING austin: 4096 data bytes
4104 bytes from austin (128.185.61.5): icmp_seq=0. time=8. ms
4104 bytes from austin (128.185.61.5): icmp_seq=1. time=9. ms
4104 bytes from austin (128.15.61.5): icmp_seq=2. time=9. ms
4104 bytes from austin (128.15.61.5): icmp_seq=3. time=9. ms
4104 bytes from austin (128.15.61.5): icmp_seq=4. time=8. ms
4104 bytes from austin (128.15.61.5): icmp_seq=5. time=9. ms
4104 bytes from austin (128.15.61.5): icmp_seq=6. time=9. ms
4104 bytes from austin (128.15.61.5): icmp_seq=7. time=9. ms
4104 bytes from austin (128.15.61.5): icmp_seq=8. time=9. ms
4104 bytes from austin (128.15.61.5): icmp_seq=9. time=9. ms

----austin PING Statistics----

10 packets transmitted, 10 packets received, 0% packet loss
round-trip (ms)  min/avg/max = 9/9/15
martyp $
```

In this example, we **ping** *austin* every five seconds, with a packet size of *4096* bytes for a total of ten times.

Let's now get back to our example.

How do I know that I have a connection between system1 and the router and the other systems on the other side of the router? I use the **ping** command. Here is how I know that system1 is connected to router1:

$ ping router1

PING router1: 64 byte packets

64 bytes from 128.185.61.2: icmp_seq=0. time=0. ms

64 bytes from 128.185.61.2: icmp_seq=1. time=0. ms

64 bytes from 128.185.61.2: icmp_seq=2. time=0. ms

Each line of output here represents a response that was returned from the device that was pinged. This means that the device responded. You continue to get this response indefinitely and have to type **^c** (control c) to terminate the **ping**. If no output is produced, as shown below, then no response occurred and you may have a problem between your system and the device to which you are checking the connection:

```
$ ping system2
PING router1: 64 byte packets
```

In this scenario, you would see this message and that is as far as you would get. A **^c** will kill the **ping**, and you see that some number of packets were sent and none were received. I did indeed get this response when issuing the **ping** command, so I know that a problem exists with the connection between system1 and router1.

ping should be used only for testing purposes such as manual fault isolation, because it generates a substantial amount of network traffic. You do not want to use ping on an ongoing basis, such as in a script that is running continuously.

A nice variation of **ping** that I use is to specify a packet size of 4096 bytes, rather than the default of 64 bytes shown in the previous examples, and count the number of times **ping** transmits before terminating, rather than having to type **^c** to terminate **ping**. The following example shows this:

```
$ ping router1 4096 5

PING router1: 64 byte packets
4096 bytes from 128.185.51.2: icmp_seq=0. time=8. ms
4096 bytes from 128.185.51.2: icmp_seq=1. time=8. ms
4096 bytes from 128.185.51.2: icmp_seq=2. time=9. ms
4096 bytes from 128.185.51.2: icmp_seq=3. time=8. ms
4096 bytes from 128.185.51.2: icmp_seq=4. time=8. ms
```

Notice that the time required to transmit and receive a response, the round-trip time, is substantially longer than with only 64 bytes transmitted. I usually find that the round-trip time for 64 bytes is 0 ms, although this depends on a number of factors, including network topology and network traffic.

netstat

From the earlier description of the subnet mask, you can see that routing from one host to another can be configured in a variety of ways. The path that information takes in getting from one host to another depends on routing.



You can obtain information related to routing with the **netstat** command. The *-r* option to **netstat** shows the routing tables, which you usually want to know, and the *-n* option can be used to print network addresses as numbers rather than as names. With the *-v* option, you get additional information related to routing, such as the subnet mask. In the following examples, **netstat** is issued with the *-r* option (this is used when describing the **netstat** output), the *-rn* options, and the *-rnv* options, so you can compare the outputs:

```
# netstat -r
Routing tables
Dest/Netmask          Gateway           Flags  Refs       Use  Interface  Pmtu
o2                    o2                UH       0   1890905  lo0        4136
o2                    o2                UH       0       343  lan1       4136
o2                    o2                UH       0         0  lan0       4136
10.1.1.0              o2                U        2         0  lan0       1500
10.1.1.0              o2                U        2         0  lan1       1500
127.0.0.0             o2                U        0         0  lo0        4136
default               10.1.1.1          UG       0         0  lan1       1500
#
```

```
# netstat -rn
Routing tables
Dest/Netmask          Gateway           Flags  Refs       Use  Interface  Pmtu
127.0.0.1             127.0.0.1         UH       0   1891016  lo0        4136
10.1.1.10             10.1.1.10         UH       0       343  lan1       4136
10.1.1.110            10.1.1.110        UH       0         0  lan0       4136
```

```
10.1.1.0              10.1.1.110         U        2          0  lan0       1500
10.1.1.0              10.1.1.10          U        2          0  lan1       1500
127.0.0.0             127.0.0.1          U        0          0  lo0        4136
default               10.1.1.1           UG       0          0  lan1       1500
#
```

```
# netstat -rmv
Routing tables
Dest/Netmask                 Gateway            Flags  Refs       Use  Interface  Pmtu
127.0.0.1/255.255.255.255      127.0.0.1          UH      0    1891036  lo0        4136
10.1.1.10/255.255.255.255      10.1.1.10          UH      0        343  lan1       4136
10.1.1.110/255.255.255.255     10.1.1.110         UH      0          0  lan0       4136
10.1.1.0/255.255.255.0         10.1.1.110         U       2          0  lan0       1500
10.1.1.0/255.255.255.0         10.1.1.10          U       2          0  lan1       1500
127.0.0.0/255.0.0.0            127.0.0.1          U       0          0  lo0        4136
default/0.0.0.0                10.1.1.1           UG      0          0  lan1       1500
#
```

The first and second outputs show that our system, *o2*, has three interfaces: The first is the loopback interface called *lo0*. The second. is at *.10*, and the third is at *.110* (which we can see from the *-rn* output). The next two lines show that our destination of *10.1.1.0*, which is a network, can be accessed through either the card at *.10* or *.110*. The third output provides verbose information. The last line is for the default route. This entry says to send packets to *10.1.1.1* if a more direct route can't be found.



With **netstat**, some information is provided about the router. The *-r* option shows information about routing, but many other useful options to this command are also available. Of particular interest in this output is "Flags," which defines the type of routing that takes place. Here are descriptions of the most common flags from the UNIX manual pages:

1=U	Route to a *network* via a gateway that is the local host itself.
3=U	Route to a *network* via a gateway that is the remote host.
5=UH	Route to a *host* via a gateway that is the local host itself.
7=UG	Route to a *host* via a remote gateway that is a host.

Also, I use two forms of **netstat** to obtain network statistics, as opposed to routing information. The first is **netstat -i**, which shows the state of interfaces that are autoconfigured. Because I am most often interested in getting a summary of **lan0**, I issue this command. **netstat -i** gives a good rundown of *lan0*, such as the network it is on, its name, and so on.

The following example shows the output of **netstat -i** on an HP-UX system:

```
# netstat -i

Name  Mtu   Network         Address            Ipkts Ierrs    Opkts Oerrs  Coll
ni0*  0     none            none                   0     0        0     0     0
ni1*  0     none            none                   0     0        0     0     0
lo0   4608  loopback        127.0.0.1            232     0      232     0     0
lan0  1500  169.200.112     169.200.112.2    3589746     2    45630     0   104
```

```
# netstat -i
Name           Mtu Network           Address               Ipkts      Opkts
lan1          1500 10.1.1.0          o2                 59935480  163641547
lan0          1500 10.1.1.0          o2                   139173   12839358
lo0           4136 127.0.0.0         o2                  1892333    1892345
#
```

Here is a description of the fields in the **netstat** example:



Name	The name of your network interface (Name), in this case, *lan0*.
MTU	The "maximum transmission unit," which is the maximum packet size sent by the interface card.
Network	The network address of the LAN to which the interface card is connected (169.200).
Address	The host name of your system. This is the symbolic name of your system as it appears in the file **/etc/hosts**.

The statistical information includes:

Ipkts	The number of packets received by the interface card, in this case **lan0**.
Ierrs	The number of errors detected on incoming packets by the interface card (on some UNIX variants).
Opkts	The number of packets transmitted by the interface card.
Oerrs	The number of errors detected during the transmission of packets by the interface card (on some UNIX variants).
Collis	The number of collisions that resulted from packet traffic (on some UNIX variants).



netstat provides cumulative data since the node was last powered up; you might have a long elapsed time over which data was accumulated. If you are interested in seeing useful statistical information, you can use **netstat** with different options. You can also specify an interval over which to report statistics. I usually ignore the first entry, because it shows all data since the system was last powered up. Therefore, the data includes non-prime hours when the system was idle. I prefer to view data at the time the system is working its hardest. The following **netstat** example provides network interface information every five seconds:

```
# netstat -I lan0 5

(lan0)-> input           output        (Total)-> input           output
    packets  errs  packets  errs colls      packets  errs  packets  errs colls
    3590505     2    45714     0   104      3590737     2    45946     0   104
        134     0        5     0     0          134     0        5     0     0
        174     0        0     0     0          174     0        0     0     0
        210     0       13     0     0          210     0       13     0     0
        165     0        0     0     0          165     0        0     0     0
        169     0        0     0     0          169     0        0     0     0
        193     0        0     0     0          193     0        0     0     0
        261     0        7     0     0          261     0        7     0     0
        142     0        8     0     0          142     0        8     0     0
        118     0        0     0     0          118     0        0     0     0
        143     0        0     0     0          143     0        0     0     0
        149     0        0     0     0          149     0        0     0     0
```

With this example, you get multiple outputs of what is taking place on the LAN interface. As I mentioned earlier, you may want to ignore the first output, because it includes information over a long time period. This may include a time when your network was idle, and therefore the data is not important to you.

The following **netstat** example provides network interface information every five seconds on an HP-UX 11i system:

```
# netstat -I lan0 5
(lan0)-> input      output        (Total)-> input      output
        packets     packets                 packets     packets
        2500745     2236834                 3781914     3518003
```

```
          13          14                1190        1191
           6           4                1140        1138
         282         278                 282         278
          32          28                  32          28
           2           2                   2           2
          52          52                  52          52
```

Notice that delta values are shown and not totals as was the case in older releases of HP-UX.

You can specify the network interface on which you want statistics reported by using **-I interface**; in the case of the example, it was *-I lan0*. An interval of five seconds was also used in this example.

Yet another use of **netstat** is to show the state of network sockets. **netstat -a** produces a list of protocols, queues, local and remote addresses, and protocol states. All this information is useful for showing active communications, as shown in the following example:

```
# netstat -a
Active Internet connections (including servers)
Proto Recv-Q Send-Q  Local Address          Foreign Address        (state)
tcp        0      2  system1.telnet        atlm0081.atl.hp..1319  ESTABLISHED
tcp        0      0  *.1095                 *.*                    LISTEN
tcp        0      0  *.psmond               *.*                    LISTEN
tcp        0      0  *.mcsemon              *.*                    LISTEN
tcp        0      0  localhost.8886         localhost.1062         ESTABLISHED
tcp        0      0  localhost.1062         localhost.8886         ESTABLISHED
tcp        0      0  *.8886                 *.*                    LISTEN
tcp        0      0  *.8887                 *.*                    LISTEN
tcp        0      0  *.1006                 *.*                    LISTEN
tcp        0      0  *.978                  *.*                    LISTEN
tcp        0      0  *.22370                *.*                    LISTEN
tcp        0      0  *.389                  *.*                    LISTEN
tcp        0      0  *.8181                 *.*                    LISTEN
tcp        0      0  *.1054                 *.*                    LISTEN
tcp        0      0  *.1053                 *.*                    LISTEN
tcp        0      0  *.diagmond             *.*                    LISTEN
tcp        0      0  *.1045                 *.*                    LISTEN
tcp        0      0  *.1038                 *.*                    LISTEN
tcp        0      0  *.135                  *.*                    LISTEN
tcp        0      0  *.smtp                 *.*                    LISTEN
tcp        0      0  *.1036                 *.*                    LISTEN
tcp        0      0  *.appconn              *.*                    LISTEN
tcp        0      0  *.spc                  *.*                    LISTEN
tcp        0      0  *.dtspc                *.*                    LISTEN
tcp        0      0  *.recserv             *.*                    LISTEN
tcp        0      0  *.klogin              *.*                    LISTEN
tcp        0      0  *.kshell              *.*                    LISTEN
tcp        0      0  *.chargen             *.*                    LISTEN
tcp        0      0  *.discard             *.*                    LISTEN
tcp        0      0  *.echo                *.*                    LISTEN
tcp        0      0  *.time                *.*                    LISTEN
tcp        0      0  *.daytime             *.*                    LISTEN
tcp        0      0  *.printer             *.*                    LISTEN
tcp        0      0  *.auth                *.*                    LISTEN
tcp        0      0  *.exec                *.*                    LISTEN
tcp        0      0  *.shell               *.*                    LISTEN
tcp        0      0  *.login               *.*                    LISTEN
tcp        0      0  *.telnet              *.*                    LISTEN
tcp        0      0  *.ftp                 *.*                    LISTEN
tcp        0      0  *.795                 *.*                    LISTEN
tcp        0      0  *.792                 *.*                    LISTEN
tcp        0      0  *.*                   *.*                    CLOSED
tcp        0      0  *.787                 *.*                    LISTEN
tcp        0      0  *.783                 *.*                    LISTEN
tcp        0      0  *.779                 *.*                    LISTEN
tcp        0      0  *.portmap             *.*                    LISTEN
tcp        0      0  *.2121                *.*                    LISTEN
udp        0      0  *.1127                *.*
udp        0      0  *.177                 *.*
udp        0      0  *.1003                *.*
udp        0      0  *.*                   *.*
udp        0      0  *.*                   *.*
udp        0      0  *.*                   *.*
udp        0      0  *.*                   *.*
udp        0      0  *.nfsd                *.*
udp        0      0  *.976                 *.*
```

```
udp        0      0  *.22370                *.*
udp        0      0  *.1097                 *.*
udp        0      0  *.1095                 *.*
udp        0      0  *.1079                 *.*
udp        0      0  *.135                  *.*
udp        0      0  *.*                    *.*
udp        0      0  *.1045                 *.*
udp        0      0  *.snmp                 *.*
udp        0      0  *.1040                 *.*
udp        0      0  *.tftp                 *.*
udp        0      0  *.chargen              *.*
udp        0      0  *.discard              *.*
udp        0      0  *.echo                 *.*
udp        0      0  *.time                 *.*
udp        0      0  *.daytime              *.*
udp        0      0  *.ntalk                *.*
udp        0      0  *.bootps               *.*
udp        0      0  *.1023                 *.*
udp        0      0  *.787                  *.*
udp        0      0  *.798                  *.*
udp        0      0  *.797                  *.*
udp        0      0  *.1037                 *.*
udp        0      0  *.*                    *.*
udp        0      0  *.1036                 *.*
udp        0      0  *.1035                 *.*
udp        0      0  *.777                  *.*
udp        0      0  *.portmap              *.*
udp        0      0  *.1034                 *.*
udp        0      0  *.syslog               *.*
udp        0      0  *.2121                 *.*
Active UNIX domain sockets
Address  Type   Recv-Q Send-Q    Inode     Conn     Refs  Nextref Addr
  bb9c00 stream      0      0   af9000        0        0        0 /tmp/.AgentSoA
  ced700 dgram       0      0   c99400        0        0        0 /opt/dcelocalr
  ce9e00 dgram       0      0   d23000        0        0        0 /opt/dcelocalr
  b0d200 dgram       0      0   b87000        0        0        0 /opt/dcelocalr
  997a00 stream      0      0   b84800        0        0        0 /opt/dcelocal1
  b24e00 dgram       0      0   b84000        0        0        0 /opt/dcelocal1
  d59400 dgram       0      0   b66400        0        0        0 /var/tmp/psb_t
  d85c00 dgram       0      0   b67000        0        0        0 /var/tmp/psb_t
  c8b200 dgram       0      0   b12000        0        0        0 /opt/dcelocalr
  c8b400 stream      0      0   b78400        0        0        0 /opt/dcelocal5
  c8b300 dgram       0      0   b78000        0        0        0 /opt/dcelocal5
  c90900 dgram       0      0   d22400        0        0        0 /opt/dcelocalr
  c78c00 dgram       0      0   ba1000   c4a180        0        0 /opt/dcelocal0
  b1e900 dgram       0      0   9a4400        0   c32e80        0 /opt/dcelocald
  d64100 stream      0      0   d24c00        0        0        0 /opt/dcelocal5
  9e1600 dgram       0      0   9a4000   d4d940        0        0 /opt/dcelocal2
  d64200 dgram       0      0   cfc800        0   c32c80        0 /opt/dcelocal9
  d12d00 dgram       0      0   cfc000   c32c00        0        0 /opt/dcelocal1
  c5ee00 stream      0      0   b1c000        0        0        0 /opt/dcelocal4
  d19d00 dgram       0      0   ce4800        0        0        0 /opt/dcelocald
  cf0c00 dgram       0      0   a92800        0   af15c0        0 /opt/dcelocal7
  d2d600 dgram       0      0   a93800   c32c00        0   d4db80 /opt/dcelocal0
  c9b900 dgram       0      0   a93c00        0        0        0 /opt/dcelocald
  d6c800 stream      0      0   ba3000        0        0        0 /var/opt/OV/sT
#
```

A lot of information is given in this output. You can refer to the manual page in Appendix A if you want a detailed explanation of the fields.



The first line shows the *Proto tcp* to the *Local Address system1.telnet* as having a *(state)* of *ESTABLISHED*. This is the connection we have initiated to this system. We are sitting on *system1* with a telnet session open to the system on which we ran **netstat**.

Most of the remaining *tcp* protocol entries are listening. This means that they are listening for incoming connections, as indicated by the *LISTEN*. They have a wild card in the *Foreign Address* field, which will contain the address when a connection has been established. We are one of the few connections that has been made, as indicated by the *ESTABLISHED*.

All the send and receive queues, shown as *Recv-Q* and *Send-Q*, are empty as indicated by *0*.

The UNIX domain sockets at the end of the output are stream and datagram connections for a variety of services such as NFS.

This output gives you an appreciation of the immense amount of activity taking place from a networking perspective on your UNIX system. Networking and connectivity have been among the most advanced aspects of UNIX since its inception.

route



The information displayed with **netstat** comprises the routing tables for your system. Some are automatically created with the **ifconfig** command when your system is booted or when the network interface is initialized. Routes to networks and hosts that are not directly connected to your system are entered with the **route** command.



Routing changes can be made on the fly, as I did to change the *Flags* from *U* to *UG*:

```
$ /usr/sbin/route add default 128.185.61.1 3
```

First is the **route** command. Second, we specify that we wish to add a route; the other option is to delete a route. Third, we specify the destination, in this case, the default. This could be a specific host name, a network name, an IP address, or default that signifies the wild card gateway route that is shown in our example. Fourth is the gateway through which the destination is reached. In the above example, the IP address was used, but this could also be a host name. The 3 corresponds to the count that is used to specify whether the gateway is the local host or a remote gateway. If the gateway is the local host, then a count of 0 is used. If the gateway is a remote host, which is the case in the example, a count of >0 is used. This corresponds to *UG* for *Flags*. This manually changed the network routing table by adding a default route with the appropriate *Flags*. Issuing this command fixed the problem I encountered trying to get system1 to talk to the systems on the other side of the router (remember Figure 12-10?).

Before issuing **/usr/sbin/route** with the **add** option, you can first use the **delete** option to remove the existing default route, which is not working.

route commands usually appear in one of the system's startup files so that every time the system boots, **route** commands are issued. This ensures that the right connectivity information is in place every time the system starts.

ifconfig



The **ifconfig** command provides additional information on a LAN interface. The following example provides the configuration of a network interface:

```
$ /etc/ifconfig lan0
lan0:   flags=863<UP,BROADCAST,NOTRAILERS,RUNNING>
         inet 128.185.61.2 netmask ffff0000 broadcast 128.185.61.255
```

From this example, we can quickly see that the interface is up, it has an address of 128.185.61.2, and it has a netmask of *ffff0000*. Again, keep in mind that your network interface may have a different name, such as *le0*.

You can use **ifconfig** to get the status of a network interface as I have done here to assign an address to a network interface, or to configure network interface parameters. The network address you have falls into classes such as "A," "B," or "C," as mentioned earlier. You want to be sure that you know the class of your network before you start configuring your LAN interface. This example is a class "B" network, so the netmask is defined as ffff0000 (typical for a class "B" address), as opposed to ffffff00, which is typical for a class "C" network. The netmask is used to determine how much of the address to reserve for subdividing the net work into smaller networks. The netmask can be represented in hex, as shown above, or in decimal format, as in the **/etc/hosts** file. Here is the **ifconfig** command I issued to configure the interface:

```
$ /etc/ifconfig lan0 inet 128.185.61.2 netmask 255.255.0.0
```

- The *255.255.0.0* corresponds to the hex *ffff000* shown earlier for the class "B" subnet mask.
- *lan0* is the interface being configured.
- *inet* is the address family, which is currently the only one supported for this system.
- *128.185.61.2* is the address of the LAN interface for system1.
- **netmask** shows how to subdivide the network.
- *255.255.0.0* is the same as *ffff0000*, which is the netmask for a class "B" address.





I have made good use of **netstat, ping**, and **ifconfig** to help get the status of the network. **ifconfig, route**, and **/etc/hosts** are used to configure the network, should you identify any changes you need to make. The subnet examples show how flexible you can be when configuring your network for both your current and future needs. In simple networks, you may not need to use many of these commands or complex subnetting. In complex networks, or at times when you encounter configuration difficulties, you may have to make extensive use of these commands. In either case, network planning is an important part of setting up UNIX systems.

Most of the commands used throughout this chapter are a part of every system administrator's tool box. Networking is so vital to the use of UNIX systems, however, that having background in this area can help with your overall understanding of the system and how to use it more effectively.

rpcinfo

As a user, you may have a need to NFS mount a directory on another system or perform some other function that you haven't before used on your system. You can determine whether various pieces of functionality have been enabled by evaluating the daemons running on your system. **rpcinfo** allows you to query the Remote Procedure Call (RPC) port wrapper, called *rpcbind* on a system, including your local system, by issuing the command **rpc -p** *system_name*.

The following example shows issuing **rpcinfo -p** on our local system:

```
# rpcinfo -p
   program vers proto   port  service
    100000    2   tcp    111  portmapper
    100000    2   udp    111  portmapper
    100024    1   udp    777  status
    100024    1   tcp    779  status
    100021    1   tcp    783  nlockmgr
    100021    1   udp   1035  nlockmgr
    100021    3   tcp    787  nlockmgr
    100021    3   udp   1036  nlockmgr
    100020    1   udp   1037  llockmgr
    100020    1   tcp    792  llockmgr
    100021    2   tcp    795  nlockmgr
    100068    2   udp   1040  cmsd
```

```
    100068    3   udp   1040  cmsd
    100068    4   udp   1040  cmsd
    100068    5   udp   1040  cmsd
    100083    1   tcp   1036  ttdbserver
    100005    1   udp    976  mountd
    100005    1   tcp    978  mountd
    100003    2   udp   2049  nfs
    150001    1   udp   1003  pcnfsd
    150001    2   udp   1003  pcnfsd
    150001    1   tcp   1006  pcnfsd
    150001    2   tcp   1006  pcnfsd
#
```

Many daemons are running on the system that are important to the functionality I like to use. **mountd** and **nfs** are running, which indicates that another system or PC could NFS mount file systems on this computer. There is other setup required for the mount to take place, but at least the daemon is running to support this functionality. In addition, **pcnfsd** is running, meaning that we have support for Windows-based NFS access.

arp

The mechanism used to maintain a list of IP addresses and their corresponding MAC addresses is the *ARP cache*. The mapped addresses are only held in the cache for minutes, so if you want to see what addresses have been mapped recently, you can use the **arp** command as shown in the following example:

```
# arp -a
o2 (10.1.1.10) at 0:10:83:f7:a2:f8 ether
l1 (10.1.1.11) at 0:10:83:f7:2e:d0 ether
63.88.85.1 (63.88.85.1) at 0:30:94:b0:b8:a0 ether
l3 (10.1.1.200) at 0:10:83:fc:92:88 ether
tape1 (10.1.1.14) at 0:10:83:f7:e:32 ether
tape1 (10.1.1.14) at 0:10:83:f7:e:32 ether
tape1 (10.1.1.14) at 0:10:83:f7:e:32 ether
tape1 (10.1.1.14) at 0:10:83:f7:e:32 ether
63.88.85.18 (63.88.85.18) -- no entry
```

Current *arp* entries are displayed with the *-a* command. You can create an entry with the *-s* option if pinging the host does not do it for you. This will rarely need to be done; however, some system administrators neet to delete arp table entries to flush the arp cache. Use **arp -d** *hostname* or *IP address* to do this.

lanadmin

lanadmin is used to view and perform administration on network cards. Issuing **lanadmin** with no options brings you into the interactive interface as shown in the following example:

```
# lanadmin


          LOCAL AREA NETWORK ONLINE ADMINISTRATION, Version 1.0


               Copyright 1994 Hewlett Packard Company.
                      All rights are reserved.

Test Selection mode.

        lan      = LAN Interface Administration
        menu     = Display this menu
        quit     = Terminate the Administration
        terse    = Do not display command menu
        verbose  = Display command menu

Enter command: lan

LAN Interface test mode. LAN Interface PPA Number = 0

        clear    = Clear statistics registers
        display  = Display LAN Interface status and statistics registers
        end      = End LAN Interface Administration, return to Test Selection
        menu     = Display this menu
```

```
        ppa      = PPA Number of the LAN Interface
        quit     = Terminate the Administration, return to shell
        reset    = Reset LAN Interface to execute its selftest
        specific = Go to Driver specific menu

Enter command: d

                      LAN INTERFACE STATUS DISPLAY


PPA Number                      = 0
Description               = lan0 Hewlett-Packard 10/100 TX Half-Duplex  TT = 1500
Type (value)                    = ethernet-csmacd(6)
MTU Size                        = 1500
Speed                           = 100000000
Station Address                 = 0x1083ffcaae
Administration Status (value)   = up(1)
Operation Status (value)        = down(2)
Last Change                     = 237321866
Inbound Octets                  = 0
Inbound Unicast Packets         = 0
Inbound Non-Unicast Packets     = 0
Inbound Discards                = 0
Inbound Errors                  = 0
Inbound Unknown Protocols       = 0
Outbound Octets                 = 820
Outbound Unicast Packets        = 20
Outbound Non-Unicast Packets    = 0
Outbound Discards               = 1
Outbound Errors                 = 0
Outbound Queue Length           = 0
Specific                        = 655367

Press <Return> to continue
```

In this example, we issued **lanadmin** and specified that we wanted to go into the *lan* interface administration and that we wanted to *d*isplay information about the interface.

To switch to a different LAN interface or *PPA* just enter the *PPA* number for it. This is usually the same as the LAN number and can be found in the **lanscan** output.

lanadmin can also be used to perform such tasks as to change the MTU or speed of a LAN interface with the *-M* and *-s* options, respectively. To see the speed at which you LAN interface is set you can issue the following command:

```
# lanadmin -x 0
Current Config                  = 10 Half-Duplex AUTONEG

#
```

The **lanadmin** command was issued for LAN interface *0* which is shown to be at ten MBits per second and half-duplex.

nslookup and nsquery

nslookup is used to resolve a host name into an IP address. You issue **nslookup** *hostname* and **nslookup** will access either the **/etc/resolv.conf** file or **/etc/hosts** to resolve the host name. The following example shows a system using **/etc/hosts** to produce the IP address of system *l2*:

```
# nslookup l2
Using /etc/hosts on:  l3
looking up FILES
Name:    l2
Address:  10.1.1.12

#
```

You can also run **nslookup** in interactive mode by issuing the command with no command-line arguments. The following example shows issuing the command with no command line arguments to get into interactive mode and then typing help to get information on commands you can issue:

```
# nslookup l2
```

```
> help
NAME              - print address information about NAME
IP-ADDRESS        - print hostname information about IP-ADDRESS
policy            - print switch policy information
server NAME       - set default server to NAME, using current default server
lserver NAME      - set default server to NAME, using initial server
set OPTION         - sets the OPTION
    all            -  print options, current server and host
    [no]swtrace -  print lookup result and lookup switch messages
>
```

Used in conjunction with **nslookup** is the **nsquery** command. **nsquery** is used to verify a hostname or IP lookup as well as verify lookups of usernames and groups based on the policies of **/etc/nsswitch.conf**. It is a more trustworthy tool than **nslookup** because it implements resolver timeouts and policies for DNS, NIS, NIS+, and local files. It can perform lookup of a name, user ID, group ID, or IP address.

To query information about the user *hackley* in the **passwd** file, you would issue **nsquery passwd hackley**.

To query the **group** file for users, you would issue **nsquery group users**.

To search the **hosts** file for *www.hp.com* you would issue **nsquery hosts www.hp.com**. Depending on the configuration of **nsswitch.conf** the **hosts** file and then **dns** would be queried if *www.hp.com* were not found in **hosts**.

To search **hosts** for *192.16.16.204,* you would issue **nsquery hosts 192.16.16.204**.

ndd

ndd is used to perform network tuning and view information about network parameters. To view information about all supported tunable parameters with **ndd**, you would issue **ndd -h supported**. You can get the value of a parameter using the *-get* option; you can set the value of a parameter with the *-set* option.

Changes made with the **ndd** command are not permanent meaning that they will not be in place after a reboot of the system. To make permanent changes you would edit the **/etc/rc.config.d/nddconf** file.

An example of using **ndd** would be to work with the ICMP source quench. This is often disabled for security purposes. To check the current value you would use **ndd -get /dev/ip_send_source_quench**. To disable this with **ndd** you would use **ndd -set /dev/ip ip_send_source_quench 0**.

To make this a permanent change, you would create the following **/etc/rc.config.d/ndd** entries:

```
TRANSPORT_NAME[0]=ip
NDD_NAME[0]=ip_send_source_quench
NDD_VALUE[0]=0
```

In general using files in **/etc/rc.confg.d** is a good practice since these changes are normally permanent.

[Team LiB]

PREVIOUS NEXT

Using Networking

The ISO/OSI model is helpful for visualizing the way in which the networking layers interact. The model does not, however, tell you how to use the networking. Two widely used networking services that may be running on your system(s) and are worth taking a look at are ARPA and NFS.

The products we're about to cover in the upcoming sections fall under the umbrella of "Internet Services." The first networking product to try on your system is what is sometimes called ARPA Services - what I have been calling ARPA. ARPA is a combination of "ARPA Services" and "Berkeley Services." ARPA Services supports communications among systems running different operating systems, and Berkeley Services supports UNIX systems. The terms *ARPA* and *Berkeley* aren't used as often as they once were; however, since the roots of these command are in ARPA and Berkeley I'll use these terms in the following sections. The following sections are a list of the most common ARPA and Berkeley commands. Although many programs can be run under each of these services, the following are the most commonly used services in the UNIX world. In some cases, there are examples that show how these commands are used. For most of the examples, the local host is **system1** and the remote host is **system2**.

Internet Services ARPA (Communication among Systems w/ Different OS)

File Transfer Protocol (ftp) Transfer a file, or multiple files, from one system to another. This is often used when transferring files between a UNIX workstation and a Windows PC, VAX, etc. The following example shows copying the file **/tmp/krsort.c** from system2 (remote host) to the local directory on system1 (local host):

	Comments
$ ftp system2	Issue ftp command
Connected to system2.	
system2 FTP server (Version 4.1) ready.	
Name (system2:root): root	Log in to system2
Password required for root.	
Password:	Enter password
User root logged in.	
Remote system type is UNIX.	
Using binary mode to transfer files.	
ftp> **cd /tmp**	**cd** to **/tmp** on system2
CWD command successful	
ftp> **get krsort.c**	Get krsort.c file
PORT command successful	
Opening BINARY mode data connection for **krsort.c**	
Transfer complete.	
2896 bytes received in 0.08 seconds	
ftp> **bye**	Exit ftp
Goodbye.	
$	

In this example, both systems are running UNIX; however, the commands you issue through **ftp** are operating system-independent. The **cd** for change directory and **get** commands used above work for any operating system on which **ftp** is running. If you become familiar with just a few **ftp** commands, you may find that transferring information in a heterogeneous networking environment is not difficult.

Chances are that you are using your UNIX system(s) in a heterogeneous environment and may therefore use **ftp** to copy files and directories from one system to another. Because **ftp** is so widely used, I describe some of the more commonly used **ftp** commands:

ascii Set the type of file transferred to ASCII. This means that you are transferring an ASCII file

from one system to another. **binary** is usually the default.

Example: **ascii**

binary Set the type of file transferred to binary. This means that you are transferring a binary file from one system to another. If, for instance, you want to have a directory on your UNIX system that holds applications that you copy to non-UNIX systems, then you want to use binary transfer.

Example: **binary**

cd Change to the specified directory on the remote host.

Example: **cd /tmp**

dir List the contents of a directory on the remote system to the screen or to a file on the local system, if you specify a local file name.

get Copy the specified remote file to the specified local file. If you don't specify a local file name, then the remote file name will be used.

lcd Change to the specified directory on the local host.

Example: **lcd /tmp**

ls List the contents of a directory on the remote system to the screen or to a file on the local system, if you specify a local file name.

mget Copy multiple files from the remote host to the local host.

Example: **mget *.c**

put Copy the specified local file to the specified remote file. If you don't specify a remote file name, then the local file name will be used.

Example: **put test.c**

mput Copy multiple files from the local host to the remote host.

Example: **mput *.c**

bye/quit Close the connection to the remote host.

Example: **bye**

Other **ftp** commands are available in addition to those I have covered here. If you need more information on these commands or wish to review additional **ftp** commands, the UNIX manual pages for **ftp** are helpful.

telnet Used for communication with another host using the telnet protocol. Telnet is an alternative to using **rlogin**, described later. The following example shows how to establish a telnet connection with the remote host, system2:

	Comments
$ telnet system2	
Connected to system2.	Telnet to system2
AIX version 4 system2	
login: **root**	Log in as root on system2
password:	Enter password
Welcome to system2. - rs6000 aix 4.3.1.0	
$	AIX prompt on system2

Internet Services Berkeley (Communication between UNIX Systems)

Remote Copy (rcp)

This program is used to copy files and directories from one UNIX system to another. To copy **/tmp/krsort.c** from system1 to system2, you could do the following:

```
$ rcp   system2:/tmp/krsort.c  /tmp/krsort.c
```

Some networking configuration needs to be made to files to get this level of functionality. In this example, the user who issues the command is considered "equivalent" on both systems and has permission to copy files from one system to the other with **rcp**. (These terms are described shortly).

Remote login (rlogin)

Supports login to a remote UNIX system. To remotely log in to system2 from system1, you would do the following:

```
$ rlogin system2
password:
Welcome to system2
$
```

If a password is requested when the user issues the **rlogin** command, the users are not equivalent on the two systems. If no password is requested, then the users are indeed equivalent. You can also issue **rlogin** *system* **-l** *user* to specify the *system* and *user* as part of the command.

Remote shell (remsh)

With the **remsh** command, you can sit on one UNIX system and issue a command to be run remotely on a different UNIX system and have the results displayed locally. This command is called **rsh** on Linux and some other UNIX variants. In this case, a **remsh** is issued to show a long listing of **/tmp/krsort.c**. The command is run on system2, but the result is displayed on system1, where the command was typed:

```
$ remsh system2 ll /tmp/krsort.c
-rwxrwxrwx 1 root sys 2896 Sept 1 10:54 /tmp/krsort.c
$
```

In this case, the users on system1 and system2 must be equivalent, or else permission is denied to issue this command.

Remote who (rwho)

Find out who is logged in on a remote UNIX system. Here is the output of issuing **rwho**:

```
$ rwho
root      system1:ttyu0      Sept 1 19:21
root      system2:console    Sept 1 13:17
tomd      system2:ttyp2      Sept 1 13:05
|          |      |         |      |> time of login
|          |      |         |> day of login
|          |      |
|          |      |> terminal line
|          |> machine name
|
|> user name
```

For **rwho** to work, the **rwho** daemon (**rwhod**) must be running.

Other "r" commands, in addition to those covered are available. Also, variations of these commands occur going from one UNIX variant to another, so you may not run exactly the same "r" command on your UNIX system.

[Team LiB]

PREVIOUS NEXT

Host Name Mapping

The most important decision related to networking is how host name mapping is implemented on your system in Internet Services ARPA. Three techniques are available for host name mapping:

- Berkeley Internet Named Domain (BIND)
- Network Information Service (NIS)
- UNIX file **/etc/hosts**

The most common and simplest way to implement host name mapping is with **/etc/hosts**, so I cover this technique in the next section. Keep in mind that there are probably networking manuals for your UNIX variant devoted to many networking topics including NFS, ARPA, and others. These manuals serve as a good reference material if you need to know more about networking than is covered here.

Using the **/etc/hosts** file, as you are about to see, becomes very difficult for environments where there are many systems deployed. With this solution there is one **/etc/hosts** file that must be kept up-to-date and propagated to all other systems.

The Domain Name System (DNS) is widely used in large environments. DNS uses Berkeley Internet Name Domain Service (BIND) to resolve names to addresses. There are name servers that fill a request for name data. This is the server side to BIND. There is a client side to BIND, called the resolver, that accesses the name server(s) to resolve names. Using this client/server model, it is much easier to maintain naming information, because it only needs to be kept in a few places as opposed to one for each system.

Clients use a file called **/etc/resolv.conf** to configure the resolver. The name server and its corresponding address are the keys to resolving information.

This solution makes it much easier to maintain system names and addresses in large environments. DNS and BIND are primarily a system administration exercise to setup. From a user standpoint, you don't need to know much about them. What I will instead focus on in the upcoming sections are some of the programs in which users are more interested. I will supply some background so that the way in which the programs are used has more meaning. In general, though, I'll concentrate on the user aspect of these networking topics, as opposed to the system administration aspect of them.

/etc/hosts

This file contains information about the other systems to which you are connected. It contains the Internet address of each system, the system name, and any aliases for the system name. If the **/etc/hosts** file is modified to contain the names of the systems on your network, they have provided the basis for **rlogin** to another system. Although you can now **rlogin** to other UNIX systems, you cannot yet **rcp** or **remsh** to another system. Although adding **remsh** and **rcp** functionality is easy, it does indeed compromise security, so it is not always set up on all systems. Here is an example **/etc/hosts** file:

```
127.0.0.1     localhost loopback
15.32.199.42  a4410827
15.32.199.28  a4410tu8
15.32.199.7   a4410922
15.32.199.21  a4410tu1
15.32.199.22  a4410tu2
15.32.199.62  a4410730
15.32.199.63  hpxterm1
15.32.199.64  a4410rd1
15.32.199.62  a4410750hp1
```

This file is in the following format:

<internet_address> *<official_hostname>* *<alias>*

The Internet Protocol address (IP address) is a class "A," "B," or "C" address. A class "A" network supports many more nodes per network than either a class "B" or "C" network. The purpose of breaking down the IP address into four fields is to define a node (or host) address and a network address. These were described in detail in figures 9-3 through 9-6.

Assuming that the above **/etc/hosts** file contains class "C" addresses, the rightmost field is the host or node address, and the other three fields comprise the network address.

You could use either the official_hostname or the alias from the **/etc/hosts** file when issuing one of the ARPA or Berkeley Internet Services commands described earlier. For instance, either of the following ARPA commands work:

```
$ telnet a4410750
```

or

```
$ telnet hp1
```

Similarly, either of the following Berkeley commands works:

```
$ rlogin a4410750
```

or

```
$ rlogin hp1
```

/etc/hosts.equiv

Your system may be setup so that user's don't have to issue a password when they **rlogin** to a remote system, they can set up equivalent hosts by editing this file. As I mentioned earlier, this is technique sometimes considered a security risk, so it is not always employed. The login names must be the same on both the local and remote systems for **/etc/hosts.equiv** to allow the user to bypass entering a password. You can either list all the equivalent hosts in **/etc/hosts.equiv** or list the host and user name you wish to be equivalent. Users can now use **rcp** and **remsh**, because they are equivalent users on these systems. I usually just enter all the host names on the network. Here is an example of **/etc/hosts.equiv**:

```
a4410730
a4410tu1
a4410tu2
hpxterm1
a4410827
a4410750
```

Keep in mind the potential security risks of using **/etc/hosts.equiv**. If a user can log into a remote system without a password, you have reduced the overall level of security on your network. Even though users may find it convenient to not have to enter a password when logging into a remote system, you have given every user in **/etc/hosts.equiv** access to the entire network. If you could ensure that all the permissions on all the files and directories on all systems were properly set up, then you wouldn't care who had access to what system. In the real UNIX world, however, permissions are sometimes not what they are supposed to be. Users have a strong tendency to "browse around," invariably stumbling upon a file they want to copy to which they really shouldn't have access.

/.rhosts

This file is the **/etc/hosts.equiv** for a superuser. If you log in as root, you want to have this file configured with exactly the same information as **/etc/hosts.equiv**. If you do, however, you have compounded your network security risk by allowing superuser on any system to log in to a remote system without a root password. If you are the undisputed ruler of your network and you're 100 percent certain that no security holes exist, then you may want to set up **/.rhosts** so that you don't have to issue a password when you log in remotely to a system as superuser. From a security standpoint, however, you should know that this setup is frowned upon.

I have described the process of setting up the appropriate files to get the most commonly used ARPA Services up and running. There is sometimes even more advanced functionality, such as DNS/BIND, required. You system may have DNS/BIND or similar functionality set up that gives you access to some or all of the commands covered throughout this section.

[Team LiB]

Set Up DNS Servers Using BIND 8.1.2 and 9.2 on HP-UX 11i

This procedure documents how to setup a Master (primary), a Slave (secondary), and a Caching-only DNS server on HP-UX 11.0/11i platforms with the latest supported BIND version 8.1.2 release for HP-UX. BIND 8.1.2 is loaded on 11.0 via a patch from http://software.hp.com and is provided on the 11i release media. BIND 9.2 is available on software.hp.com for download. For a more detailed discussion on BIND 8.1.2 and 9.2, please refer to the O'Reilly and Associates 3rd Edition book DNS and BIND, by Albitz and Liu. BIND 8.1.2 and 9.2 are available on the 11.0/11i platforms only and not on 10.20. 10.20 uses the older 4.9.7 version. It is important to be aware that the older 4.9.X versions of BIND have a different configuration file (boot file) format. There are also some terminology differences. The "boot" file is now called a "config" or "conf" file. HP Primary nameservers are now called Master nameservers; Secondary name servers are now called Slaves.

In most cases, DNS implementation is fairly straightforward, i.e., resource records and database files, etc. The real test comes in planning. Considerations such as whether the DNS system will be protected behind a firewall, whether or not you wish your resolvers to "peer" out on the Internet, and nslookup remote domains and addresses must be considered. Some administrators choose to turn off looking out on the internet to reduce the network traffic it creates. If you are directly attached to the Internet, you will need to register at least two nameservers with an authorized Internet Domain Name provider. A list of these providers can be found at http://www.icann.org/registrars/accredited-list.html. If you are not directly attached, i.e., sitting behind a firewall, an ISP is a good choice for DNS services and will do most of the work for you.

When you have completed your planning you need to configure your DNS server. What names should you use? Some people use planets, locational references, and sometimes people use names for their network computers and printers. The decision is up to you, but do not use underscores in your hostnames (see "check-names" boot file options), and make the names easy to remember. Here are the steps to creating a Master (Primary) DNS server for your network:

1. Populate **/etc/hosts** with all of the hosts that you want to administer, separated by network segment and domains, and keep everything in an orderly fashion so that a tool such as hosts_to_named can safely and efficiently administer database files for you, as in the following example:

   ```
   /etc/hosts
   15.17.186.159 wtec712-rtr

   127.0.0.1 localhost loopback
   # 812 Bogus Domain BASEBALL.HP.COM

   # NL - EAST DIVISION
   10.1.1.1 atlantabraves atlantabraves.baseball.hp.com

   10.1.1.2 newyork newyorkmets.baseball.hp.com
   # NL - CENTRAL DIVISION

   10.1.2.1 houstonastros houstonastros.baseball.hp.com

   10.1.2.2 chicagocubs chicagocubs.baseball.hp.com

   10.1.2.3 stlouiscardinals stlouscardinals.baseball.hp.com
   ```

2. Create a *param* file with the parameters for your domain as shown in the following example for **/tmp/param**:

   /tmp/param

   ```
   -d baseball.hp.com   <--------- Your domain.
   -n 10.1.1            <--------- Your subnet(s).
   -n 10.1.2
   -n 10.1.3
   -n 15.17.186
   -H /etc/hosts.dnstest <--------- hosts file you will use.
   -r                    <--------- If this nameserver is a
                                    root NS. Our examples
                                    below are NOT for root NS.
   ```

3. Run **hosts_to_named** with your newly created **param** file from the directory in which you want to place the database. Note that the **param** file is not necessary, but, it is a good idea; the options can be run from the command line instead of including a file. Another common error with **hosts_to_named** is not running it in a "clean" directory. If you need to preserve results before you run it, **mkdir /etc/named.data.old**; **mv /etc/named.data/* /etc/named.data.old** to preserve the prior configuration.

   ```
   # cd /etc/named.data
   ```

```
# hosts_to_named -f /tmp/param

Translating /tmp/hosts.dns to lower case ...

Collecting network data ...

10.1.1
10.1.2
10.1.3
15.17.186

Creating list of multi-homed hosts ...
Creating "A" data (name to address mapping) for net 10.1.1 ...
Creating "PTR" data (address to name mapping) for net 10.1.1 ...
Creating "A" data (name to address mapping) for net 10.1.2 ...
Creating "PTR" data (address to name mapping) for net 10.1.2 ...
Creating "A" data (name to address mapping) for net 10.1.3 ...
Creating "PTR" data (address to name mapping) for net 10.1.3 ...
Creating "A" data (name to address mapping) for net 15.17.186 ...
Creating "PTR" data (address to name mapping) for net 15.17.168 ...
Creating "MX" (mail exchanger) data ...
Building default named.boot file ...

Building default db.cache file ...
WARNING: db.cache must be filled in withthe name(s) and address(es)
of the rootserver(s)
Building default boot.cacheonly for caching only servers ...
done
```

4. **hosts_to_named** will produce both a BIND 4.X **named.boot** file and a BIND 8.X and 9.X **named.conf** file. Match the db files created in **/etc/named.data** with those found in **/etc/named.data/named.conf.** Here is an example for a non-root name server from 8.1.2:

```
// generated by named-bootconf.pl

options {
check-names response fail;   // do not change this
check-names slave warn;
directory "/etc/named.data"; // running directory for named
/*
* If there is a firewall between you and nameservers you want
* to talk to, you might need to uncomment the query-source
* directive below. Previous versions of BIND always asked
* questions using port 53, but BIND 8.1 uses an unprivileged
* port by default.
*/
// query-source address * port 53;
};

//
// type domain source file
//

zone "0.0.127.IN-ADDR.ARPA" {
type master;
file "db.127.0.0";
};

zone "baseball.hp.com" {
type master;
file "db.baseball";
};

zone "1.1.10.IN-ADDR.ARPA" {
type master;
file "db.10.1.1";
};

zone "2.1.10.IN-ADDR.ARPA" {
type master;
file "db.10.1.2";
};

zone "3.1.10.IN-ADDR.ARPA" {
type master;
file "db.10.1.3";
};
```

```
zone "186.17.15.IN-ADDR.ARPA" {
type master;
file "db.15.17.186";
};

zone "." {
type hint;
file "db.cache";
};

(wtec712-rtr)named.data- ls

boot.cacheonly db.10.1.2 db.15.17.186 named.boot
conf.cacheonly db.10.1.3 db.baseball named.conf
db.10.1.1 db.127.0.0 db.cache params-file
```

As you can see, **hosts_to_named** created reverse lookup (IP addresses to names) db files with one parent domain, **baseball.hp.com**. Each of the nodes in our network will have the fully-qualified name **shortname.baseball.hp.com**.

5. Since many installations still use underscores in hostnames, we highly recommend that you allow for this by modifying the *check-names* options lists in the **named.conf** file that **hosts_to_named** generates. For example, you may want to change these lines:

```
options {
check-names response fail           // do not change this
check-names slave warn
.
.
.

to:

options {
check-names response ignore    //change "fail" to "ignore"
check-names slave ignore       //change "warn" to "ignore"
check-names master ignore      //add this whole new line
.
.
.
```

Note: BIND 9.X does not support the *check-names* option and will notify **syslog** about it.

6. Fill in **db.cache** with the addresses of the root name servers. If you are directly connected to the Internet and will be querying the root nameservers at the NIC, obtain an updated list from the site ftp://internic.net/domain/named.cache.

Next, copy the file to **db.cache** after you have downloaded. If you are not directly connected to the Internet and have to go through a firewall to query the root name servers, make the firewall your root name server and point your **db.cache** entry at the firewall in the same fashion that you would specify the root name servers. It is very important to configure **db.cache** correctly or services such as mail and name resolution will be affected. Here is an example of this:

```
; FILL IN THE NAMES AND ADDRESSES OF THE ROOT SERVERS

;

; . 99999999 IN NS root.server.

; root.server. 99999999 IN A ??.??.??.??

;
. 99999999 IN NS firewall.baseball.hp.com.

firewall.baseball.hp.com. 99999999 IN A 15.17.186.99
```

In this case, **firewall.baseball.hp.com**. is my firewall and since my name server cannot directly query the root nameservers, **db.cache** is directed to my firewall.

7. If you are going to have a Master (primary) name server that will not talk to the internet in any way, shape, or form, then you need to setup your name server as a root name server by doing a couple of things. Either use the *-r* option in your **params** file to **hosts_to_named** or make the following changes:

In your **named.conf** file, change:

```
zone "." {

type hint;
```

```
file "db.cache";
```

to:

```
zone "." {

type master;

file "db.root";
```

The root name server database file **db.root** would contain:

```
.   IN  SOA  m3107ced.baseball.hp.com.  root.m3107ced.baseball.hp.com.

(

             1        ; Serial
             10800    ; Refresh every 3 hours
             3600     ; Retry every hour
             604800   ; Expire after a week
             86400 )  ; Minimum ttl of 1 day

    IN NS m3107ced.baseball.hp.com.
m3107ced.baseball.hp.com. IN A 15.50.73.92
```

What we have done is set up an internal root name server, **db.root**, with one record, **m3107ced.baseball.hp.com**.

8. The next consideration is where to send queries for domains for which you are not authoritative for. If your domain is **baseball.hp.com** and someone asks for **jughead.ibm.com**, what happens to the request? Well, if you've configured a root nameserver with no forwarder statements, the answer is nothing. The query will fail with host not found. This might be a good thing if you do not want your internal systems querying internet domains. Security and network congestion are usually the reason. What if you want to resolve Internet names and addresses, however? Easy, configure a forwarders statement in your **/etc/named.conf** to point to the firewall or whichever system is talking directly to the root name servers, as shown in the following BIND 8.1.2 example:

```
options {
check-names response ignore ;   // change fail to ignore

check-names slave ignore         // change warn to ignore

check-names master ignore       // add this line


forwarders 15.253.24.10 15.253.32.10 15.253.24.10 15.253.32.10

directory "/etc/named.data"; // running directory for named

/*
* If there is a firewall between you and nameservers you want
* to talk to, you might need to uncomment the query-source
* directive below. Previous versions of BIND always asked
* questions using port 53, but BIND 8.1 uses an unprivileged
* port by default.
*/
// query-source address * port 53;
};
```

What's happening here is that queries for domains that we are not authoritative for, basically anything outside **baseball.hp.com**, we are sending to the forwarders to let them handle it. As you can see, there are two forwarders, each listed twice. The reason for this is that forwarders tend to be quite busy and by specifying two of them, you prevent the query from timing out. Be sure to copy **/etc/named.data/named.conf** to **/etc/named.conf**. All versions of BIND will look for the boot file in **/etc/** by default, so don't forget to copy it to **/etc** when you are ready.

9. Configure **/etc/resolv.conf** and **/etc/nsswitch.conf** on your name server and all clients that will be pointed at the name server:

```
/etc/resolv.conf

domain baseball.hp.com
search baseball.hp.com atl.hp.com hp.com rose.hp.com cup.hp.com
external.hp.com

nameserver 15.50.73.92 # authoritative name server 4 atl.hp.com
nameserver 15.51.240.8 # non-authoritative cache only servers
```

/etc/resolv.conf is pretty simple: the domain statement identifies which domain the system is part of; the search statements are used to simplify typing when there is more than one domain. When you issue a query for say, *jughead*, it will search for *jughead* first in **baseball.hp.com**, then **atl.hp.com**, **rose.hp.com**, and finally, **cup.hp.com**. **/etc/nsswitch.conf** modifies the switch order you will use to look up hosts and IP addresses. There are four possible sources for this information: 1) **/etc/hosts**, 2) **nis**, 3) **nisplus**, 4) and **dns** By default, the hard coded order is **dns nis** files. To modify the switch order, you need to copy in a fresh **/etc/nsswitch.conf** file from **/usr/newconfig/etc/nsswitch.hp_defaults** and modify the *hosts* entry:

```
# /etc/nsswitch.hp_defaults:

#

# An example file that could be copied over to /etc/nsswitch.conf; it

# uses NIS (YP) in conjunction with files.

#

passwd: compat

group: compat

hosts: files [NOTFOUND=return] dns

networks: nis [NOTFOUND=return] files

protocols: nis [NOTFOUND=return] files

rpc: nis [NOTFOUND=return] files

publickey: nis [NOTFOUND=return] files

netgroup: nis [NOTFOUND=return] files

automount: files nis

aliases: files nis

services: nis [NOTFOUND=return] files
```

As you can see, the *hosts* line has been modified so that we consult the **/etc/hosts** file first then continue on to *dns* if the query is unsuccessful. There are many ways to modify the switch order and many ways to mess things up. Use discretion when changing the switch order and consult the man pages on *switch* for more information.

10. Start up the DNS name server process as follows:

```
# /usr/sbin/named.
# ps -eaf|grep named

root 8074 1 0 08:42:08 ? 0:00 /usr/sbin/named <--- check to make
                                                  sure it is running..

root 8077 8072 2 08:42:13 ttyp7 0:00 grep named
```

After the name server is started, you use can **sig_named** to perform various functions. After you modify any of the *db* files or **/etc/named.boot**, you need to tell **named** to refresh its databases. You can accomplish this using either of the two following commands:

```
# sig_named restart

# kill -HUP `/var/run/named.pid`
```

This will reload the databases, which you can verify by viewing the end of file **/var/adm/syslog/syslog.log** as shown below:

[View full width]

```
Oct 12 08:49:13 m3107ced named[8074]: primary zone "0.0.127.IN-ADDR.ARPA" loaded (serial 1)

Oct 12 08:49:13 m3107ced named[8074]: primary zone "baseball.hp.com" loaded (serial 1)

Oct 12 08:49:13 m3107ced named[8074]: primary zone "1.1.10.IN-ADDR.ARPA" loaded(serial 1)

Oct 12 08:49:13 m3107ced named[8074]: primary zone "2.1.10.IN-ADDR.ARPA" loaded(serial 1)

Oct 12 08:49:13 m3107ced named[8074]: primary zone "3.1.10.IN-ADDR.ARPA" loaded(serial 1)
```

```
Oct 12 08:49:13 m3107ced named[8074]: primary zone "1.168.192.IN-ADDR.ARPA" loaded (
➥ serial 1)

Oct 12 08:49:13 m3107ced named[8074]: primary zone "2.168.192.IN-ADDR.ARPA" loaded (
➥ serial 1)

Oct 12 08:49:13 m3107ced named[8074]: primary zone "3.168.192.IN-ADDR.ARPA" loaded (
➥ serial 1)

Oct 12 08:49:13 m3107ced named[8074]: Ready to answer queries.
```

As you can see, **named** loaded each of our databases, or zones, and is ready to answer queries.

11. Test queries both by name and IP address.

```
# nslookup atlantabraves
Using /etc/hosts on: wtec712-rtr
looking up FILES
Trying DNS

Name: atlantabraves.baseball.hp.com

Address: 10.1.1.1

# nslookup 10.1.1.1

Using /etc/hosts on: wtec712-rtr
looking up FILES
Trying DNS
Trying DNS
Name: atlantabraves.baseball.hp.com
Address: 10.1.1.1
```

Looking up *atlantabraves* by name and IP address worked and generally we are done.

We always check **/var/adm/syslog/syslog.log** for messages from **named**. **named** logs a lot of seemingly unimportant chatter, but it always deserves at least a short look.

A Word on Slave (Secondary) Name Servers

Once you have created a Master (primary) DNS server, you have completed all the hard work. Creating Slaves and cache-only servers is simple. Let's walk through this process step-by-step:

1. Use **ftp** to copy the **named.conf** file and **db.cache** from the Master (primary).
2. Edit the **named.conf** file as follows:

   - Change each instance of "master" to "slave", except for the loopback domain db.127.0.0 and the cache entry.

   - Add a "masters" entry for each zone with the IP address of the Master DNS server.

   Using our BIND 8.1.2 example from above:

```
# cat /etc/named.conf

// generated by named-bootconf.pl

options {

check-names response fail; // do not change this
check-names slave warn;
directory "/etc/named.data"; // running directory for named

/*
* If there is a firewall between you and nameservers you want
* to talk to, you might need to uncomment the query-source
* directive below. Previous versions of BIND always asked
* questions using port 53, but BIND 8.1 uses an unprivileged
* port by default
*/
// query-source address * port 53;
};

//
```

```
// type domain source file
//
zone "0.0.127.IN-ADDR.ARPA" {
type master;
file "db.127.0.0";
};

zone "baseball.hp.com" {
type slave;
file "db.baseball";
masters ( 15.17.186.159);
};

zone "1.1.10.IN-ADDR.ARPA" {
type slave;
file "db.10.1.1";
masters ( 15.17.186.159);
};

zone "2.1.10.IN-ADDR.ARPA" {
type slave;
file "db.10.1.2";
masters ( 15.17.186.159);
};

zone "3.1.10.IN-ADDR.ARPA" {
type slave;
file "db.10.1.3";
masters ( 15.17.186.159);
};

zone "186.17.15.IN-ADDR.ARPA" {
type slave;
file "db.15.17.186";
masters ( 15.17.186.159);
};

zone "." {
type hint;
file "db.cache";
};
```

3. Now, all you need to do, once your **named.boot** has been copied to **/etc/named.boot** and your **/etc/named.data** directory has been created, is kick off a zone transfer.

   To kick off a zone transfer, all you need to do is **kill named** with

   ```
   # sig_named restart
   ```

   or

   ```
   # kill -HUP `/var/run/named.pid`
   ```

   to start the transfer.

   Take a look at **/etc/named.data** and you should see all the database files there now. Also, check **/var/adm/syslog/sylog.log** to make sure that the zones were loaded properly, as shown in the following example:

   [View full width]

   ```
   Oct 12 09:21:34 stimpy named[1893]: secondary zone "1.1.10.IN-ADDR.ARPA" loaded (serial 1)

   Oct 12 09:21:35 stimpy named[1893]: secondary zone "baseball.hp.com" loaded (ser ial 1)

   Oct 12 09:21:35 stimpy named[1893]: secondary zone "2.1.10.IN-ADDR.ARPA" loaded (serial 1)

   Oct 12 09:21:35 stimpy named[1893]: secondary zone "1.168.192.IN-ADDR.ARPA" loaded (
   ➥ serial 1)

   Oct 12 09:21:35 stimpy named[1893]: secondary zone "3.1.10.IN-ADDR.ARPA" loaded (serial 1)

   Oct 12 09:21:36 stimpy named[1893]: secondary zone "2.168.192.IN-ADDR.ARPA" loaded (
   ➥ serial 1)

   Oct 12 09:21:36 stimpy named[1893]: secondary zone "3.168.192.IN-ADDR.ARPA" loaded (
   ➥ serial 1)
   ```

Notice the serial number entries. They should match on the Master (primary) and Slaves (secondaries).

A Word On Cache-Only Nameservers

Why in the world would you want a cache-only name server? In a word, performance. If you want to maintain a local cache but do not want to manage database files, this is the way to go. It will act as any other name server, responding to queries in the same fashion, except queries that are built in the cache will be non-authoritative. Any query that comes back with a non-authoritative reply is a query received from cache. Is that a bad thing? No, but be aware that the data may have changed on the Master (primary) and the cache replies may be outdated. The Time To Live (TTL) flag for each query is a way to manipulate the time, in seconds, that a name server may cache the answer to a query, versus having to contact an authoritative name server. The default TTL for records is usually 86400 seconds, or 24 hours. You may want to play with this value, depending on how frequently or infrequently the records are updated. A good rule of thumb is to have at least one name server per subnet, and cache-only name servers are an excellent choice.

To configure a cache-only nameserver, copy down the **conf.cacheonly** file and db.cache from the Master (primary) nameserver as shown in the following BIND 8.1.2 example:

```
// generated by named-bootconf.pl

options {
check-names response fail; // do not change this
check-names slave warn;
directory "/tmp/testhack"; // running directory for named
/*
* If there is a firewall between you and nameservers you want
* to talk to, you might need to uncomment the query-source
* directive below. Previous versions of BIND always asked
* questions using port 53, but BIND 8.1 uses an unprivileged
* port by default.
*/
// query-source address * port 53;
};

//
// type domain source file
//
zone "0.0.127.IN-ADDR.ARPA" {
type master;
file "db.127.0.0";
};

zone "." {
type hint;
file "db.cache";
};
```

Please don't forget to fill in **db.cache** with the name server(s) for which you will be caching data for.

A Final Word on Name Server Setup

This section in no way offers a comprehensive discussion on setting up DNS. It is only intended to act as a cookbook after all your planning has been completed. Please refer to the book *HP Installing and Administering Internet Services* at http://docs.hp.com. DNS is pretty straightforward, but any syntax errors or problems with **/etc/named.boot** can have dramatic consequences. The **syslog** can be your best friend when zones are not transferring or there are problems with the data.



Network File System (NFS)

NFS allows you to mount disks on remote systems so that they appear as though they are local to your system. Similarly, NFS allows remote systems to mount your local disk so that it looks as though it is local to the remote system. We'll go through a simple NFS server and client setup in this section after NFS background is supplied. You can use standard-mounted directories or automounted directories with NFS. Our upcoming example uses standard-mount directories but you can find out more about automounted directories and all NFS-related topics in *Installing and Administering NFS Services* from docs.hp.com. Configuring NFS to achieve this functionality is simple. Here are the steps to go through in order to configure NFS:

1. Configure NFS startup files to specify whether your system will be an NFS Client, NFS Server, or both.
2. Start NFS server processes.
3. Specify which of your local file systems can be mounted by remote systems.
4. Specify the remote disks you want to mount and view as if they were local to your system.

As with Internet Services, you could enable other aspects to NFS, but again, I cover what I know to be the NFS functionality that nearly every UNIX installation uses.

Because NFS may be setup on your system to meet the needs of many users, you may want to understand the terminology associated with NFS. The following are commonly used NFS terms:

Node	A computer system that is attached to or is part of a computer network.
Client	A node that requests data or services from other nodes (servers).
Server	A node that provides data or services to other nodes (clients) on the network.
File System	A disk partition or logical volume.
Export	Makes a file system available for mounting on remote nodes using NFS.
Mount	Accesses a remote file system using NFS.
Mount Point	The name of a directory on which the NFS file system is mounted.
Import	Mounts a remote file system.

Some of the specific configuration tasks and related files are different among UNIX variants. The following are some general tasks and examples related to configuring NFS. Your system administrator, of course, has to deal with the specifics of configuration on the UNIX variants.

Your system must be an NFS client, NFS server, or both. There are also daemons that must be running to support NFS. Both of these tasks are performed somewhat differently among the UNIX variants.

Your system then imports remote file systems to which you have local access and exports local file systems that are accessed by other systems.

A remote file system that you are mounting locally has an entry similar to the one that follows in **/etc/fstab, /etc/filesystems**, or whatever file is used to mount file systems:

```
system2:/opt/app3  /opt/app3   nfs rw,suid 0 0
```

In this case, we are mounting **/opt/app3** on *system2* locally as **/opt/app3**. This is an NFS mount with the permissions shown.

You can use the **showmount** command to show all remote systems (clients) that have mounted a local file system. This command is supported on most UNIX variants. **showmount** is useful for determining the file systems that are most often mounted by clients with NFS. The output of **showmount** is particularly easy to read, because it lists the host name and the directory that was mounted by the client. You have the three following options to the **showmount** command:

-a prints output in the format "name:directory"

-d lists all the local directories that have been remotely mounted by clients

-e prints a list of exported file systems

Example NFS Configuration

Let's take a look at the steps to setup an NFS server and client. We'll perform the following steps on our NFS server, with hostname *asodevlab1*, to export the directory **/home/frame**:

1. Configure NFS startup files by setting the following variables to *1* in **/etc/rc.config.d/nfsconf**:

   ```
   NFS_CLIENT=1
   NFS_SERVER=1
   NUM_NFSD=16
   NUM_NFSIOD=16
   PCNFS_SERVER=1
   ```

   All of the startup variables have been set to *1* thereby enabling all NFS-related functionality at startup. I left the number of daemons (*NUM_NFSD* and *NUM_NFSIOD*) with default values.

2. Next we'll start the NFS server and check to see what NFS-related processes are running, as shown in the following output:

   ```
   # /sbin/init.d/nfs.server start
       starting NFS SERVER networking

       starting up the rpcbind daemon
           rpcbind already started, using pid: 837
       Reading in /etc/exports
       starting up the mount daemon
           rpc.mountd already started, using pid: 1938
       starting up the NFS daemons
         nfsd(s) already started, using pid(s): 1970 1971 1973 1974 1975 1958 1959 1960 196
   1 1962 1963 1964 1965 1966 1967 1968 1976
       starting up the Status Monitor daemon
           rpc.statd already started, using pid: 884
       starting up the Lock Manager daemon
           rpc.lockd already started, using pid: 890
   # ps -ef | grep nfs
       root   842     0  0  Apr 29  ?         0:00 nfskd
       root  1970  1959  0  Apr 29  ?         0:00 /usr/sbin/nfsd 16
       root  1971  1959  0  Apr 29  ?         0:00 /usr/sbin/nfsd 16
       root  1973  1959  0  Apr 29  ?         0:00 /usr/sbin/nfsd 16
       root  1974  1959  0  Apr 29  ?         0:00 /usr/sbin/nfsd 16
       root  1975  1959  0  Apr 29  ?         0:00 /usr/sbin/nfsd 16
       root  1958     1  0  Apr 29  ?         0:00 /usr/sbin/nfsd 16
       root  1959     1  0  Apr 29  ?         0:00 /usr/sbin/nfsd 16
       root  1960  1959  0  Apr 29  ?         0:00 /usr/sbin/nfsd 16
       root  1961  1960  0  Apr 29  ?         0:00 /usr/sbin/nfsd 16
       root  1962  1960  0  Apr 29  ?         0:00 /usr/sbin/nfsd 16
       root  1963  1959  0  Apr 29  ?         0:00 /usr/sbin/nfsd 16
       root  1964  1960  0  Apr 29  ?         0:00 /usr/sbin/nfsd 16
       root  1965  1960  0  Apr 29  ?         0:00 /usr/sbin/nfsd 16
       root  1966  1960  0  Apr 29  ?         0:00 /usr/sbin/nfsd 16
       root  1967  1960  0  Apr 29  ?         0:00 /usr/sbin/nfsd 16
       root  1968  1960  0  Apr 29  ?         0:00 /usr/sbin/nfsd 16
       root  1976  1959  0  Apr 29  ?         0:00 /usr/sbin/nfsd 16
       root 23825     0  0 12:00:10 ?         0:00 nfsktcpd
       root 24527 23078  0 12:44:29 pts/ta    0:00 grep nfs
   #
   ```

   Now that we have the server started, we can proceed to export file systems in step 3.

3. We'll update the **/etc/exports** file to include a directory that will be available to all systems, as shown below:

   ```
   # cat /etc/exports
   /home/frame     #exported read/write to all systems
   #
   ```

   We have exported **/home/frame** to all systems with read/write access.

4. Run the **exportfs** command in order to make **/home/frame** available to other systems:

```
# exportfs -av //home/frame
```

You can also run **showmount -e** to see the mounted file system. We have performed all of the setup required on our server to export this file system. Next we'll proceed to the client part of the setup.

The first step is to update the **/etc/rc.config.d/nfsconf** conf file to enable the NFS client. On our client, with hostname of *m4415mxp*, we'll enable both the NFS server and client as we did on the server.

1. Configure NFS startup files by setting the following variables to *1* in **/etc/rc.config.d/nfsconf**:

```
NFS_CLIENT=1
NFS_SERVER=1
NUM_NFSD=16
NUM_NFSIOD=16
PCNFS_SERVER=1
```

   All of the startup variables have been set to *1* thereby enabling all NFS-related functionality at startup. I left the number of daemons (*NUM_NFSD* and *NUM_NFSIOD*) with default values. I also ran the **server start** command on the client in our example as well so that all of the NFS daemons are running. The NFS client is enabled by default in the **nfsconf** file.

2. Next we'll update the **/etc/fstab** file to include the mount

```
# cat /etc/fstab
# System /etc/fstab file.  Static information about the file systems
# See fstab(4) and for further details on configuring devices
/dev/vg00/lvol3 / vxfs delaylog 0 1
/dev/vg00/lvol1 /stand hfs defaults 0 1
/dev/vg00/lvol4 /home vxfs delaylog 0 2
/dev/vg00/lvol5 /tmp vxfs delaylog 0 2
/dev/vg00/lvol6 /opt vxfs delaylog 0 2
/dev/vg00/lvol7 /usr vxfs delaylog 0 2
/dev/vg00/lvol8 /var vxfs delaylog 0 2
asodevlab1:/home/frame /home/frame nfs rw,suid 0 0
#
```

The **/home/frame** directory will be automatically mounted at the next boot. We have to run the **mount** command as shown below to manually mount the file system at this time:

```
# mount /home/frame
# bdf
Filesystem          kbytes    used   avail %used Mounted on
/dev/vg00/lvol3     409600   69466  318923   18% /
/dev/vg00/lvol1     299157   26357  242884   10% /stand
/dev/vg00/lvol8    4706304  146362 4275401    3% /var
/dev/vg00/lvol7    1036288  748400  269946   73% /usr
/dev/vg00/lvol5     409600    1388  382763    0% /tmp
/dev/vg00/lvol6    3047424  749463 2154357   26% /opt
/dev/vg00/lvol4     409600    1210  382871    0% /home
asodevlab1:/home/frame
                     20480    1128   18144    6% /home/frame
# ls -l /home/frame
total 20
-r--r--r--   1 root       sys      959 May  6 12:14 copy.l.pm.Z
-r--r--r--   1 root       sys     2570 May  6 12:14 copy1.xwd
-r--r--r--   1 root       sys     2574 May  6 12:14 copy_done.xwd
-r--r--r--   1 root       sys     2575 May  6 12:14 copy_sched.xwd
#
```

We manually mounted **/home/frame**, ran **bdf** to get information on all mounted file systems, and then ran **ls -l** to see its contents. Keep in mind that our client is accessing data on a server. We can write to this directory and the data will be written on the server.

The **nfsstat -m** command can also be used to verify active NFS client mounts and options. Useful data on the NFS server is also provided by this command.

There are many NFS mount options and trade-offs related to standard-mounting vs. automounting that are not covered in this book that are covered in *Installing and Administering NFS Services*. This example; however, shows how simple it is to get NFS mounts up-and-running quickly on two systems. With *PCNFSD* running, the PC NFS daemon, you can also mount this directory from a PC.

[Team LiB]

Manual Pages for Commands Used in Chapter 13

There are manual pages in Appendix A for the following commands used in this chapter:

ifconfig
netstat

Chapter 14. Common Desktop Environment (CDE) and GNOME

Introduction

Why a Graphical User Interface (GUI)?

CDE Basics

Customizing CDE

CDE - Advanced Topics

X, Motif, and CDE Configuration Files

How Configuration Files Play Together

Specifying Appearance and Behavior

The Sequence of Events When CDE Starts

CDE and Performance

Font Paths

CDE and Virtual Partitions

Custom Greeting for Different Virtual Partitions

GNOME



Introduction

This chapter covers the Common Desktop Environment (CDE) and GNOME. Most of the chapter covers CDE but there is a brief introduction to GNOME because at the time of this writing GNOME can be downloaded from the HP Web site and runs on many HP servers. Information about GNOME and a download of GNOME for HP-UX can be found on www.hp.com/go/gnome. The following three major topics are covered in this chapter:

- Detailed CDE topics including X and Motif.
- CDE and Virtual Partitions (vPars.)
- An introduction to GNOME.

CDE represents the effort of major UNIX vendors to unify UNIX at the desktop level. CDE is widely used by X terminal and workstation users on many UNIX systems. Because you may be managing many UNIX variants that run CDE, I'll cover CDE on IBM's AIX systems, Hewlett Packard's HP-UX systems, and Sun Microsystems' Solaris systems. This chapter provides an introduction to CDE: It touches on the basics of CDE's look and feel, describes making changes to the CDE environment, and includes a bit of background about the X, Motif, and CDE relationships. CDE versions used to write this chapter are: AIX CDE 1.0, HP-UX CDE 2.1.0, and Solaris CDE 1.3. Newer releases will have enhanced features, but in general, they should still work the same.

Several features make it easy to customize CDE. The Style Manager, which every user has access to, makes it easy to customize CDE on an individual user basis. Sooner or later, however, you may want to provide some common denominator of CDE functionality for your users. If, for instance, you have an application that most users will run, you can set up environment variables, prepare pull-down menus, provide suitable fonts, an so on, that will make your users more productive. Users can then perform additional customizations such as defining File Manager characteristics and selecting backgrounds.

The following is a list of CDE-specific topics I'll cover in the upcoming sections:

1. Why a Graphical User Interface (GUI)?
2. CDE Basics
3. Customizing CDE
4. CDE: Advanced Topics
    - The Relationship Among X, Motif, and CDE
    - X, Motif, and CDE Configuration Files
    - The Sequence of Events When CDE Starts
    - CDE and Performance

First, I'll provide you with the reasoning behind providing a graphical interface rather than the more common, but also more cumbersome, line-by-line terminal interface. An overview of the CDE desktop workspaces follows. This is divided into two sections: AIX and HP-UX (because they are so similar) and Solaris. Each will give an overview of the front panel features. Next I'll guide you through making some CDE customizations. These customizations will give you a working basis for making more advanced changes on your own. I'll show you how to make some basic, simple changes, and then more complex changes, ending with modifying the login screen with a new logo and new welcome messages. Finally, I'll delve into the more advanced topics of X, Motif, and CDE relationships, configuration file usage and location, what happens internally when CDE starts up, and some CDE performance tips.

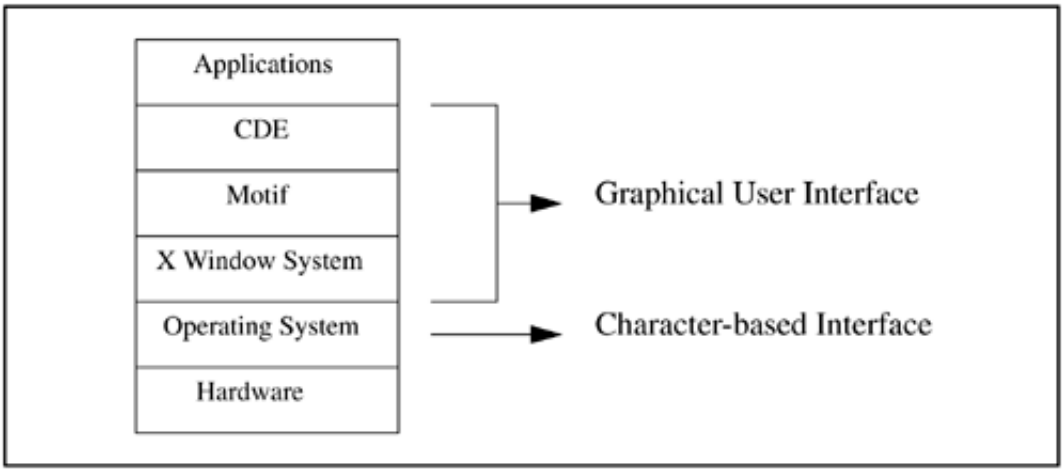


Why a Graphical User Interface (GUI)?

For computers to be used on every desktop, they had to be made easier to use. A new method of accessing computer power was required, one that avoided the command-line prompt, didn't require users to memorize complex commands, and didn't require a working knowledge of technological infrastructures such as networking. Not that this information was unimportant; far from it. The information was both too important and too specialized to be of use to the average worker-bee computer user. A knowledge of their applications was all that was important for these users. After all, so the reasoning goes, to drive a car one doesn't have to be a mechanic, so why should a computer user have to understand computer technology? The graphical user interface (GUI) makes computers accessible to the application end-user.

Figure 14-1 illustrates the relationship among the computer hardware, the operating system, and the graphical user interface. The computer is the hardware platform on the bottom. The operating system, the next layer up, represents a character-based user interface. To control the computer at this level, users must type commands at the keyboard. The next several layers, beginning with the X Window System, represent the graphical user interface. To control the computer at these levels, users manipulate graphical controls with a mouse.

Figure 14-1. User Interface Components



GUIs replaced memorization with exploration. A user could now use pull-down menus, push buttons, sliding scroll bars, and other direct manipulation to use a computer. Typing operating system commands to perform a function is greatly reduced. With a GUI, to use a computer is both easier to learn and easier to use.

Although fairly inexpensive in terms of dollars (CDE is bundled "free" with the operating system), GUIs are not without cost in terms of RAM usage and performance. Despite this performance expense, GUIs have become a permanent part of the computing environment. The benefits of their utility are worth the cost.

Beyond the graphical controls that reduce training, make mundane tasks simpler to do, and generally ease the stress of using a computer, two other benefits of GUIs are worth mentioning: multiple windows per display and client-server topology.

The benefit of multiple windows that GUIs provide is that each window (literally a rectangular area surrounded by a window frame) contains a separate application. The user can work with multiple windows open. CDE goes one step further: its multiple workspaces allow users to separate application windows by task into specific workspaces. For instance, in a workspace named "Mail," users may have application windows showing the list of incoming electronic mail, a mail message they are currently reading, and a message they are composing for later transmission. In another workspace called "Financials," they could be working on several spreadsheets, each in its own window.

Client-server topology enables the computing resources spread around a network to be accessed efficiently to meet computing needs. In a client-server topology, powerful computers on the network are dedicated to a specific purpose (file management on a file server and running applications on an application server).

Users working on less powerful client computers elsewhere on the network access the files or applications remotely. A file server reduces system administration by centralizing file backup, enabling the system administrator to back up only the file server, not each individual client computer. This setup also ensures that files will be backed up at regular intervals. An application server reduces operating costs by reducing the number and size of storage disks required and the size of RAM required on each client computer. A single version of an application resides and runs on the application server and is accessed by multiple users throughout the network.

Although this topology sounds complicated, the CDE GUI makes it easy. To access a file, users "drag and drop" a file icon from the file manager window. To start an application, users double-click the application icon. To print a file, users drag the file to the icon of the appropriate printer in the front panel and drop it there. Users don't have to know where these files and applications are, what directories they are in, what computers they are on, or how they are accessed. The underlying infrastructure and control you have put in place, along with the power of the GUI, allow users to concentrate on their work and not on the mechanics of their computer.

[Team LiB]

PREVIOUS NEXT

CDE Basics

Because most systems come with a set of CDE user guides, I'm only going to give an overview of what CDE looks like and the main areas we will be working with when we do some customizations. Along the way, I'll point out similarities and differences among the different flavors of CDE.

The CDE login screen presents you with several choices before you even log in. Under the area where you enter your user name are four buttons: **OK, Start Over, Options**, and **Help**. **OK** is just the same as pressing *Enter* when you enter your login and then password. The **Start Over** button clears your user login and allows you to start over. **Options** provides you with some initial session configuration that would need to be made before you log in. **Language** allows you to change your default language. Suppose that you need to test your company's software in another language. Assuming that the proper language is preloaded on your system, you can swap between the languages for testing by simply choosing CDE to come up in a different language. **Session** allows you to choose to come up in the CDE desktop session or into a **Failsafe** session, which is an X session but without the CDE desktop. Solaris also has options to log in to their **OpenWindow Desktop** or into the **User's Last Desktop**. **Command Line Login** allows you to log in without CDE being invoked. This would be just a regular terminal mode session. **Reset Login Screen** does just what its name suggests. And finally, **Help** lists all these features with a brief description of each.

The login screen itself displays a CDE logo, an operating system logo such as AIX, or possibly your company's logo. A welcome message appears along with a place to enter your user login. After entering your user name, you are prompted for your password. Once entered and verified as correct, CDE is started and the desktop is displayed. On Solaris systems, the first time you log in, you are presented with a choice as to whether you want to log into CDE or OpenWindows Desktop.

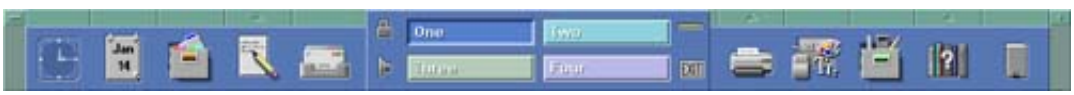
The CDE desktop is comprised of four desktop workspaces and a front panel shared by each. The front panel is an easy-to-use interface to various applications, commands, and tools. The various components are easily accessed by the simple point-and-click method. Front panel components are a collection of objects, subpanels, and access to desktop workspaces. Some objects are used only to display items such as the clock, whereas others, when clicked, either bring up an application, such as the calendar or dtmail, or perform an action, such as the lock or the exit action icons. Subpanels pop up a menu of objects that can be accessed. These objects can also simply display items or bring up applications. By default, on AIX and HP-UX, you see Personal Printer subpanels, Personal Applications subpanels, and Help subpanels. On Solaris, all panels contain subpanels. Subpanels can be added to the other panels on AIX and HP-UX as you will see in the next section "Customizing CDE." You can tell that these have subpanels, because they each have a little arrow above the panel where you click to pop it up. In the center of the front panel are four workspaces. These provide areas in which to perform related tasks, enabling the user to separate work and not clutter up the desktop.

CDE on AIX and HP-UX is very similar and, in general, what is displayed on the front panel on one is the same as on the other. Solaris, however, while retaining the basic CDE look and feel, has greatly expanded what is included on the default front panel and the subpanels behind it. I'm going to give an overview of CDE as found on HP-UX and AIX first. Then I'll point out the Solaris enhancements to CDE.

CDE on AIX and HP-UX

The front panel is divided into 11 main areas: 5 panels, 1 workspaces area, and 5 more panels. I'll give an overview of each area from left to right, beginning with the Clock and ending with the Trash Can, as shown in Figure 14-2.

Figure 14-2. Front Panel



Clock - As you would expect, this displays the current time.

Calendar - The Calendar icon displays the current date, and when clicked, brings up an appointment calendar. The calendar allows you to set appointments and reminders and to create a task list.

Appointments can be set as a one-time-only events or as recurring. You can be notified of an appointment by a beep, a pop-up message, or an e-mail. The calendar and associated appointments can be displayed by the day, the week, or the month. A yearly calendar can also be displayed, but without appointments.

File Manager - The File Manager opens a window that displays your home directory and associated files. From here, you can do any number of basic file manipulations, such as copy a file, move a file, delete a file, or execute a program or script. More infrequently used operations, such as creating a symbolic link or changing file permissions or ownership, can also be done here. Removing a file from within File Manager moves it to the Trash Can rather than permanently deleting the file. This way, if you decide that you need it back, you can simply retrieve it from the Trash Can rather than having it restored from a backup - a more time-consuming operating. However, note that the Trash Can is automatically "emptied" at the end of every session. So when you log off, the files are permanently removed.

Personal Application - The Personal Application subpanel contains the CDE's text editor, **dtpad,** and terminal emulator, **dtterm**. It also contains the Icon Editor. The **dtpad** is an easy-to-use, full-screen text editor. As in a PC-based word processor, **dtpad** allows you to move the cursor anywhere to add, change, and delete text, unlike the popular **vi**, which is a line-by-line text editor. CDE's dtterm is a good basic terminal emulator. The Icon Editor opens bitmap (.bm) or pixmap (.pm) files and allows you to edit them.

The Personal Applications subpanel is the first place we find the Install Icon application. This is where new icons are added to the subpanel. This allows the application associated with the icon to be executed when the icon is double clicked. We'll be using this when we modify the desktop in "Customizing CDE."

Mail - The CDE mailer, **dtmail**, is invoked when this icon is clicked. From here, mail messages can be composed, messages replied to, and messages forwarded. Most advanced mail features that you've come to think of as basic features are included: items such as adding attachments to messages, replying to just the sender or all recipients, and setting automatic messages saying that you're on vacation. Another feature is when a new message arrives, the **dtmail** icon changes to show a letter popping into the mailbox.

Workspace Area - The next four items comprise the workspace area.

Lock Button - The Lock button allows you to lock your session while you're away from your desk. This security feature keeps others from viewing or accessing your work when you're not there. It saves you from logging off and on every time you need to step away. Your login password, or root's password, must be entered to unlock it again.

Workspace Switch - Four in number, these are the separate workspaces created by default. These allow you to organize your work so that your workspace doesn't get cluttered up. By using the workspaces, you can keep work on different tasks, applications, or systems separated from each other. You change from one workspace to another simply by clicking on the workspace number: One, Two, Three, or Four. You'll notice that no matter what workspace you are in, the Front Panel follows you. In "Customizing CDE," you'll see how easily you can increase the number of workspaces and to change the names.

Activity Light - The Activity light, quite simply, blinks when the system is busy doing work.

Exit button - The Exit button is where you log out of CDE, terminating your CDE session. Upon exiting, depending on how you have CDE configured, you are prompted to resume your current session or return to your home session. If you choose to resume your current session, the next time you login, the desktop looks exactly the same as it does when you log out - as closely as possible. Some things, such as remote logins, are not possible, but others are, such as having an application automatically executed. If you choose to return to your home session, the next time you log in, you are returned to a known, preset configuration. This configuration is set in the Style Manager panel, discussed shortly.

Personal Printers - This subpanel contains printers that you have configured on your system, including the default printer designated as such and the Print Manager. The front panel icon is that of the default printer. To print one of your documents, simply drag it from the File Manager and drop it on the printer icon. The Print Manager allows you to view queued print files and remove them before they print. However, this works only with printers directly managed by your system. In today's networked offices, printers are usually shared and the print manager function is on a server, probably in the next building.

Style Manager - The Style Manager is one of those places where you can either get really creative getting your desktop to look just like you want or waste a lot of time - depending on your point of view. Here is where you personalize your login to the system to your own preferences. You can change your font size, your background, your mouse speed, and whether or not your session automatically locks after a certain amount of non-activity, or idle time. This is where you can set your home session, as referred to earlier, to come back to every time you log in or set your system to return to the current session or choose the option of being asked every time you log out. One other configuration you can make here is whether your window focus follows your mouse or whether you have to click on a window before it is the active window. On AIX systems, you can toggle on or off whether the workspaces are displayed on the front panel.

Application Manager - When opened, the Application Manager displays folders with useful applications and actions. Although they differ among manufacturers and even operating systems for that matter, AIX, HP-UX, and Solaris all have some basic features: Desktop Applications, Desktop Tools, Information, and System Administration. Desktop Applications contains such applications and tools as Calculator, Man Page Viewer, Icon Editor, and Create Action. Desktop Tools includes such tools as **xterm**, **xwd** capture, compress files, and reload resources. System Administration contains operating system-specific applications such as SAM in HP-UX, SMIT in AIX, and Admintool in Solaris, besides more generic actions such as change password. Take the time to look around here. You'll find many items you may want to incorporate on your customized CDE front panel. I, for one, have found the Man Page Viewer to be an invaluable resource and moved it to my front panel, where it can be readily accessed.

Help - The Help subpanel is a compilation of the Help Manager, a Desktop Introduction, Front Panel help, and an On-Item Front Panel help mechanism. The Help Manager is the main online help facility for CDE. This is a comprehensive help system with topic trees and the ability to search the index using keywords or pattern matching, backtrack where you've been in the help manager, and view the history of items for which you requested help. The Desktop Introduction is an overview of CDE and how it works. The Front Panel help facility gives information about how to use the front panel icons, subpanels, and workspaces. The On-Item Front Panel help mechanism allows you to click on the front panel item about which you wish assistance. Along with the Help Subpanel, Help can also be requested by pressing the F1 function key. If installed on your system, AIX may also include Basic Desktop Customization help and Base Library.

Trash Can - The Trash Can, used with the File Manager, holds files and folders that you have deleted during the current session. Using the facility allows you to quickly retrieve files that you should not have removed. The Trash Can can be "emptied" at any time to permanently remove items. Also, the Trash Can is "emptied" when you log out of your session.

CDE on Solaris

As mentioned earlier, Solaris has embellished CDE, adding many more subpanels and items to the subpanels, as shown in Figure 14-3. Where Solaris has greatly changed the panel and subpanels, I'll give an overview of what you'll find, as I did in the preceding section on AIX and HP-UX. Where they are the same, I'll simply note that they are the same.

Figure 14-3. Front Panel



Like AIX and HP-UX, Solaris' front panel is divided into 11 main areas: 5 panels, 1 workspaces area, and 5 more panels. I'll give an overview of each area from left to right, beginning with the World and ending with the Trash Can.

Links - As you might guess by the World icon, this is where you access the world. Solaris' Web browser, HotJava, lives here. There are also actions included to access Personal Bookmarks for the web browser, and a Find Web Page search engine. As indicated on the front panel icon, the world has a clock on it, too.

Cards - The calendar is the same as on AIX and HP-UX. However, included in the subpanel is Find Card, which is a Rolodex-type address manager.

Files - Although the title is slightly different, this is where the File Manager resides. In the subpanel are special icons to perform file actions associated with: Properties, Encryption, Compress File, Archive, and Find File. Solaris has also included actions to manage your floppy disk drive and CD-ROM.

Applications - Renamed simply Applications, here is where you'll not only find the Text Editor, but also Text Note and Voice Note. For the audio enabled, Voice Note allows you to play, record, or save audio files with WAV, AU, and AIFF formats. On this subpanel, you'll also find the Applications subpanel found under Application Manager on the AIX and HP-UX systems. As on the others, this includes Desktop Applications, Desktop Tools, Information, and System Administration. These are basically the same on all systems.

Mail - This subpanel, in addition to **dtmail**, has been enhanced to include a Suggestion Box. A very clever idea, this automatically opens up a message, pre-addressed to Sun Microsystems, Inc., so that you can send them your suggestions.

Workspace Area - The next five items comprise the workspace area:

- *Lock button* - This is the same as on AIX and HP-UX.
-
- *Workspace switch* - Same as on AIX and HP-UX.
-
- *Progress Indicator* - This is the same as the Busy Indicator on AIX and HP-UX.
-
- *Exit button* - Same as on AIX and HP-UX.
-
- *Personal Printers* - Same as on AIX and HP-UX.

Tools - The Style Manager is located here and is displayed on the front panel. You'll also find easy access to the CDE error log, and Find Process allows you to view all processes running on your system and kill selected ones. The Customized Workspace Menu and Add Item to Menu actions allow you to easily modify the Workspace Menu. The Workspace Menu is accessed by placing the mouse over a blank area of the desktop and pressing the right mouse key. Whereas AIX and HP-UX come with a generic menu, Solaris has incorporated the front panel and subpanel actions into the Workspace Menu as another way to access these actions.

Hosts - Here, Solaris differs greatly from AIX and HP-UX. The Hosts subpanel contains system-related actions. On the front panel, you'll find the Performance Monitor icons indicating how busy your CPU and disk drives are. From the subpanel, this Host opens up the *dtterm* terminal emulator, and Console opens up a *dtterm* specifically for displaying console messages. System Information provides system information, including system name, hardware model, network IP address and domain, physical and virtual memory, operating system version, and date and time last rebooted. Find Host is the same as Find Card in the Cards subpanel.

Help - This is the same as on AIX and HP-UX.

Trash - The Trash Can is the same, with the addition of a subpanel icon to "empty" the Trash Can.

This concludes the overview of the look and feel of CDE. Armed with this knowledge, you can easily navigate around your desktop environment with confidence and ease. Next, you'll learn how to change that look and feel to conform to your work environment.

[Team LiB]

PREVIOUS NEXT

Customizing CDE

Before you modify any CDE configuration files, first develop a strategy. I know that I've mentioned this before, but it's important enough to mention again.

The following questions should get you started:

1. What are your users' needs?
2. Which of those needs can be met by reconfiguring CDE?
3. At what level should these changes be made (system-wide, groups of users, individual users only)?
4. Which CDE files do you need to modify (names and locations)?
5. What are the changes, and what is their order within the file?

It's also a good idea to have handy a binder containing man pages for each of the CDE components (for looking up resources and their values) and a copy of each of the CDE configuration files.

Now that you have a good understanding as to how CDE works, I'll lead you through making changes and customizing the system for either the entire user community or each individual user.

I'll precede each change with a discussion of what is involved in making the change. I'll be making some basic, simple changes as well as some more advanced changes to show you how versatile CDE can be.

One thing I need to mention is that all these changes can be made on any system. And knowing how to do these tasks "the hard way" increases your understanding of what these tools are doing for you in the background. So be sure to give each one a try.

Making Changes Using Style Manager

Font Size

When we first log in to CDE and the workspace comes up, as shown in Figure 14-4, one of the first things many users change is the size of the font. Initially set to 4, most users want a bigger font. This change is easy.

1. **Click on the Style Manager icon** on the front panel. This action brings up the Style Manager.
2. **Click on Font.**
3. **Highlight 5**.
4. **Click on OK**.

Figure 14-4. Style Manager



Backdrop and Colors

The Style Manager is also where we can change the backdrop and colors:

1. **Click on Backdrop** and we are presented with a variety of backdrop choices.

2. After we see one we like, such as Pebbles, we can **click on apply** and then **click on close** to change our backdrop. Notice, however, that this action changes the backdrop only for the workspace that we are in.

   To change the other workspaces, we can either go to them and bring up the Style Manager in that workspace or, because we already have Style Manager up, we can click in the top-right corner on the little "**-**" on the window itself, above "File," and access the pull-down menu. From here, we can choose Occupy All Workspaces. Then we can simply go to the other workspaces, and Style Manager is already up and ready for us there. The Backdrop area is the only place we have to worry about moving to other workspaces.

3. To change colors, **click on the Color icon**. We are given a list of different color schemes from which to choose. And if one isn't quite to our liking, we can easily modify the color, hue, brightness, and contrast. We can even grab a color from somewhere else, such as an image off the Internet, to include in the color scheme. Once we have the colors we like, we can save the scheme with its own name.

Adding Objects to or Removing Objects from the Front Panel

The front panel can make life just a little easier for us by including frequently used items on it. One of the most frequent actions is opening up a terminal window. And although CDE comes with **dtterm** as the default terminal window, we sometimes might want to use an **xterm** window. We'll add it to the Personal Applications subpanel, where dtterm lives on AIX and HP-UX systems. Solaris users can do the same by putting the **xterm** on the Hosts subpanel, where the This Host terminal emulator lives.

You have two ways to add objects to the CDE front panel:

- Drag and drop them into a slideup subpanel and then make them the default for that subpanel.
- Modify the **/etc/dt/appconfig/types/C/dtwm.fp** configuration file. This approach will be used later when we create actions and file types.

The basic actions to add a control button through drag and drop are as follows:

- Drag the application icon you want as a front panel button from an application manager view and drop the icon onto the installation section (the top section) of the appropriate subpanel.
- Place the mouse pointer over the icon and press mouse button 3 to display the subpanel menu.
- Select *Copy to Main Panel*.

1. **Click on the up arrow** of the Personal Applications subpanel on AIX or HP-UX or on the Hosts subpanel on Solaris so that it pops up.
2. **Click on the Application Manage**r icon, where Desktop Applications and Desktop Tools live.
3. **Double-click on Desktop_Tools**. Here you'll find **Xterm**.
4. **Drag and drop the Xterm** icon from Desktop_Tools to the Install Icon box at the top of the Personal Applications subpanel on HP-UX and AIX, or Hosts on Solaris.

Now if we want to include **xterm** on the front panel:

1. **Right-click on the Xterm** icon in the Personal Applications or Hosts subpanel.
2. **Select Copy to Main Panel** or **Promote to Front Panel**, depending on which CDE you are using.

Adding Another Workspace

Another easy change to the front panel is to add another workspace. CDE comes with a default of four workspaces, but this is easy to change. We'll add one more and call it "Web View." This can be the window where we'll access the Internet.

1. **Place the mouse in the Workspace area of the front panel** and **press the right mouse button**.

2. **Select Add Workspace** from the pull-down menu. The workspace "New" has been added.
3. **Right-click on the new workspace labeled "New"** and select **Rename**. **Type Web View** and **press Return**.

If we want to delete a Workspace, simply right-click on the Workspace to be removed and then select delete.

Making these kinds of changes is easy, and they help personalize the workspace for the individual user.

Changing the Front Panel in Other Ways

In addition to adding and removing buttons, you can shape the front panel in other ways. These other ways use Workspace Manager resources to modify default values. The following resources relate to the front panel:

- **clientTimeoutInterval** - Length of time the Busy light blinks and the pointer remains an hourglass when a client is started from the front panel.
- **geometry** - x and y coordinate location of the front panel.
- **highResFontList** - Font to use on a high-resolution display.
- **lowResFontList** - Font to use on a low-resolution display.
- **mediumResFontList** - Font to use on a medium-resolution display.
- **name** - Name of the front panel to use when multiple front panels are in **dtwm.fp**.
- **pushButtonClickTime** - Time interval distinguishing two single mouse clicks from a double click (to avoid double launching an application accidentally).
- **waitingBlinkRate** - Blink rate of the front panel Busy light.
- **workspaceList** - List of workspace names.
- **title** - Title to appear on a workspace button.

Like all other workspace manager resources, these front-panel resources have the following syntax:

```
Dtwm*screen*resource: value
```

For example, suppose that instead of the default four workspaces, all that your users need is a front panel with six workspaces named Mail, Reports, Travel, Financials, Projects, and Studio. Further, they prefer a large font and have decided upon New Century Schoolbook 10-point bold. As system administrator, you'd make everyone happy with the following resource specifications:

```
Dtwm*0*workspaceList: One Two Three Four Five Six
Dtwm*0*One*title: Mail
Dtwm*0*Two*title: Reports
Dtwm*0*Three*title: Travel
Dtwm*0*Four*title: Financials
Dtwm*0*Five*title: Projects
Dtwm*0*Six*title: Studio
Dtwm*0*highResFontList:
             -adobe-new century schoolbook-bold-r-normal\
             --10-100-75-75-p-66-iso8859-1
```

The screen designation is usually *0*, except for displays capable of both image and overlay planes. The order of screens in the **X*screens** file is what determines the screen number; the first screen, typically the image plane, is designated as 0. Note also the inclusion of workspace names (One, Two, Three, Four, Five, and Six) in the six title resource specifications.

These changes can be added to the **sys.resources** file, which is discussed in detail in the section

"Advanced CDE Topics" later in this chapter. These changes can also be made by use of the **EditResources** action to insert the new resource lines into each user's **RESOURCE_MANAGER** property and then restart the Workspace Manager.

The obvious disadvantage is that you have to physically go to each user's work area and take over the machine for a few minutes. However, on the plus side, the changes are immediate and are automatically saved in the correct **dt.resources** for users who restore their current session. You also avoid having your changes overwritten, which could happen if you modify the right **dt.resources** file at the wrong time, while the user is still logged in.

Modifying Things in Slide-up Subpanels

Subpanels are defined in **dtwm.fp** after the front panel and front-panel control definitions. To associate a subpanel with a front panel control button, the front-panel control name is listed as the container name in the subpanel definition.

Note: **/etc/dt** is where global changes are made. **$HOME/.dt** is where local or individual user changes are made.

To add a slideup subpanel to the front panel, follow these steps:

1. Copy the file **/usr/dt/appconfig/types/C/dtwm.fp** to either **/etc/dt/appconfig/types/C/dtwm.fp** or **$HOME/.dt/types/dtwm.fp**.
2. Decide with which control button the slide-up is to be associated.
3. Create the subpanel definition file in **dtwm.fp**. This will take the following form:

```
SUBPANEL   SubPanelName
{
CONTAINER_NAME  AssociatedFrontPanelControlButton
TITLE  SubPanelTitle
}
```

4. Create subpanel control definitions for the subpanel. These will take the following form:

```
CONTROL  ControlName
{
TYPE  icon
CONTAINER_NAME   SubPanelName
CONTAINER_TYPE   SUBPANEL
ICON  BitmapName
PUSH_ACTION   ActionName
}
```

As with front panel control buttons, it's easier to copy and modify an existing subpanel file than to start from scratch.

Changing the Default Printer Name Display

We'll now make an easy change to one of the slide-up subpanels. If we pop up the Personal Printers subpanel, it shows that we have a default printer configured, but not the name of it. Let's go back into the front panel file, **dtwm.fp**, and change that:

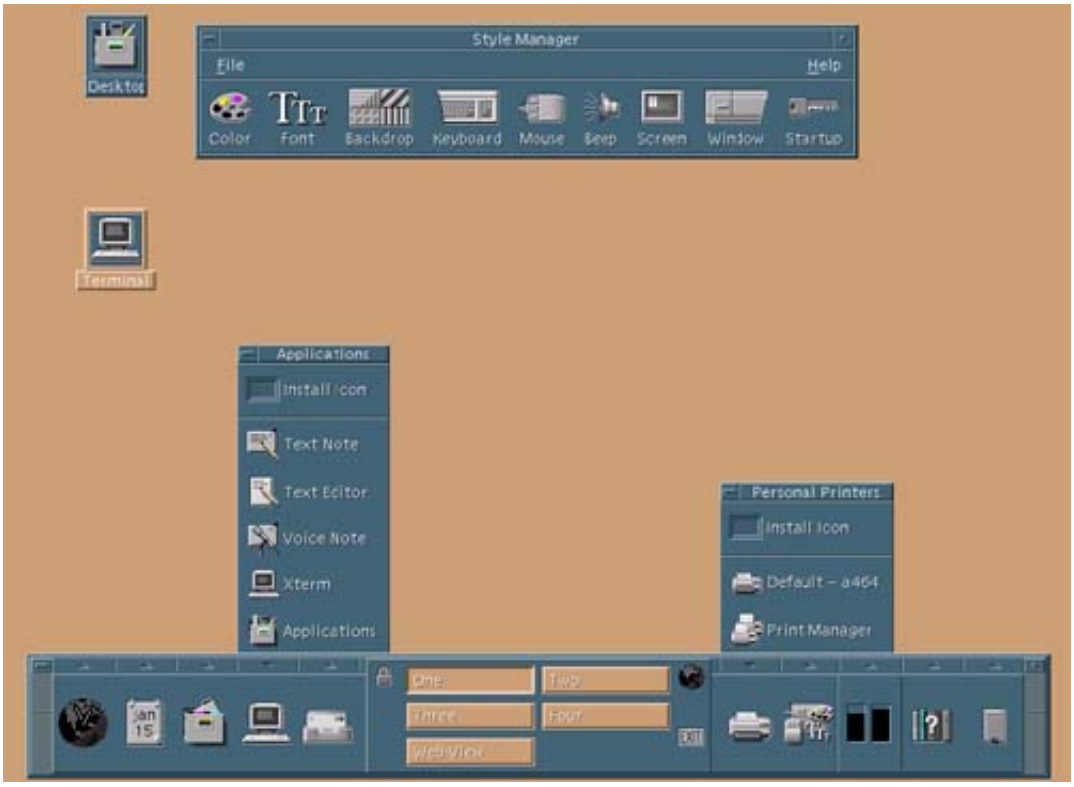
1. Bring up your favorite editor, such as **dtpad** or **vi**, and edit the file **/$HOME/.dt/types/dtwm.fp**.
2. Scroll down to CONTROL Printer. You'll see that the LABEL is Default. Change or add to that the name of your default printer. My printer is called *a464*.

   ```
   LABEL                    Default - a464
   ```

3. **Save the file** and **restart the Workspace Manager**. Position the mouse over a blank area on your workspace and press the right mouse button. Select Restart Workspace Manager. On Solaris, press the right mouse button to bring up the Customized Workspace Menu. From here, select Windows, where you'll find the Restart Workspace Manager key.

Pop up the Personal Printers subpanel to see what the default printer really is. Of course, if we change the default, we'll have to change the front panel again. But we know how to do that now, don't we! Figure 14-5 shows the Front Panel.

Figure 14-5. Front Panel with Changes



Front Panel Animation

Animation for front panel or slide-up subpanel drop zones is created by displaying a progressive series of bitmaps. By convention, the bitmaps are in **/usr/dt/appconfig/icons**. The list of bitmaps to display is contained in animation definitions at the end of **dtwm.fp**.

To create an animation sequence for a drop zone:

1. Create a progressive series of bitmaps.
2. Add a list of these bitmap files to the appropriate configuration file using the following syntax:

```
ANIMATION AnimationName
{
 bitmap0
 bitmap1
 bitmap2
 bitmap3
 bitmap4
 bitmap5
}
```

3. Add a line to the appropriate control definition using the syntax:

```
DROP_ANIMATION AnimationName
```

Adding Items to the Workspace Menu

The workspace menu is defined in the **sys.dtwmrc** file. As mentioned in the overview of the front panel for Solaris, the Workspace Menu is accessed by placing the mouse over a blank area of the desktop and pressing the right mouse key. Where AIX and HP-UX come with a generic menu, Solaris has incorporated the front panel and subpanel actions into the Workspace Menu as another way to access these actions.

However, we can create customized Workspace Menu for all three systems using the left mouse key to access it. We'll be doing this task next.

A customized Workspace menu can contain frequently used commands and applications. The customized menu is usually accessed by pressing the left mouse button, which pops up the menu for viewing and selection. A Workspace menu comes in handy for those who have a set number of things they do regularly. For instance, a programmer who uses **C++, vi**, and **isql** may have a menu with those items, or a system administrator may have a menu with **df, ps -ef**, and **netstat** on it. Or those on an expansive network may want a menu of system logins.

For one or two changes, you can modify the existing Workspace menu. For major changes, it's probably easier to insert an entirely new menu definition in **sys.dtwmrc**.

A menu definition has the following syntax:

```
Menu MenuName
{
 "Menu Name"          f.title
 "Frame"              f.exec /nfs/system1/usr/frame/bin/maker
 "Second Item"        action
 "Third Item"         action
}
```

The first line specifies the menu name, following the keyword **Menu**. The lines between the curly braces list the items that appear in the menu in their order of appearance; thus the first line is the title as designated by the function **f.title**. The second line is an example of a definition that would start **FrameMaker** from a remote application server in a distributed environment. Numerous other functions exist, approximately 45 in all. For a complete list, see the **dtwmrc** (4) man page.

For users to display the menu, you need to bind the menu definition to a mouse button and a screen location using the action **f.menu MenuName**. For example, if your users want to post the menu by pressing mouse button 3 when the pointer is on the workspace background, you would insert the following line in the Mouse Button Bindings Description section at the end of **sys.dtwmrc**:

```
<Btn3Down> root f.menu MenuName
```

(Actually, it would be easier to modify the line that's already there by exchanging **MenuName** for **DtRootMenu** on the second line.)

Now we'll create a simple menu. Our menu is going to include running FrameMaker, running the **vi** editor, and logging into a remote system.

1. First we need a copy of the **/usr/dt/config/C/sys.dtwmrc** file to **/etc/dt/config/C**:

   cp /usr/dt/config/C/sys.dtwmrc /etc/dt/config/C/sys.dtwmrc

   If we were going to make this a local change for one user only, it would be copied to **/$HOME/.dt** and renamed **dtwmrc**. For the admin1 user, that would be **/home/admin1/.dt/dtwmrc** or on Solaris in **/home/admin1/.dt/C/dtwmrc**.

2. Go into our favorite editor and **modify the file**. We're going to add our new menu just after the DtRootMenu entry, which we see on our system as the left mouse button's "Workspace Menu." Add the following:

   ```
   Menu AdminMenu
   {
    "Admin1's Menu"          f.title
    "Frame Maker"            f.exec "/nfs/system1/usr/frame/bin/maker"
    "VI Editor"              f.exec "xterm -e /usr/bin/vi"
    "Login systemA"       f.exec "xterm -geometry 80x50+830+0 -sl 200
-bg DarkOrchid4 -fg white -n SYSTEMA -T SYSTEMA -e remsh systemA
&"
     }
   ```

   The **f.title** function shows that this is the menu title. **f.exec** means to execute the following string. Notice that FrameMaker does not need a terminal window because it uses its own, whereas **vi** and the login both need a terminal window in which to run. Also, I embellished on the **xterm** for the login, making the terminal window very large, with lots of terminal memory and using specific colors.

Now, let's restart the Workspace Manager and try out our new menu:

3. Position the mouse over a blank area on the workspace and press the left mouse button. Select **Restart Workspace Manager**.

Creating pull-down menus is easy and can easily be expanded by adding submenus to the menu. Just use the function **f.menu** followed by the menu name. Add the following just after "Admin1's Menu":

```
"Work Menu"         f.menu WorkMenu''
```

And then after the } from the Menu AdminMenu section, add:

```
Menu WorkMenu
{
.
}
```

Performing these advanced functions really isn't so hard. The hardest part is remembering which directories to put the files in: **/etc/dt** for global changes or **$HOME/.dt** for individual changes.

Creating Control Buttons, Actions, and File Types

An action starts a process such as a shell script or an application. An action can be connected to a front-panel button to start when the button is clicked. An action can be connected to a front panel drop zone to be performed when a data file is dropped on the drop zone. An action can be associated with an icon in a file manager window so that the action can be started by double clicking the icon. An action can be associated with a particular data type so that double-clicking the data file icon starts the action and opens the data file.

In addition to setting up a front panel and default session to meet your user's needs, the single most important thing you can do to make computing life easier for the people who depend on you is to create actions and data types.

CDE actions and data types are defined in files that end in **.dt**. Similar to most other CDE configuration files, ***.dt** files have a system-wide version that can be copied into a user's personal directory and customized for personal use. Most system-wide ***.dt** files are found in **/usr/dt/appconfig/types/C**; personal ***.dt** files are created by copying **user-prefs.dt** from **/usr/dt/appconfig/types/C** to **$HOME/.dt/types**.

The default search path that CDE uses to look for actions and file types includes the following main directories in the order listed:

- $HOME/.dt/types
- /etc/dt/appconfig/types
- /usr/dt/appconfig/types

You can add more directories to the search path using the **DTDATABASESEARCHPATH** environment variable. Insert this environment variable and the new search path into **/etc/dt/config/Xsession** for a system-wide influence. Insert the environment variable and search path into **$HOME/.dtprofile** for individual users.

The basic control definition has six parts:

- **CONTROL name** - The definition name. This is the only part of the definition outside the curly braces.
- **TYPE** - The type of control. Several types exist. The most useful for customizing the front panel are probably blank and icon. A blank is useful as a space holder. An icon can start an action or application or be a drop zone.
- **ICON** - The bitmap to display on the front panel. Front-panel bitmaps are located in the **/usr/dt/appconfig/icons** directory.

- **CONTAINER_NAME** - The name of the container that holds the control. This must correspond to the name of an actual container listed in **dtwm.fp**.

- **CONTAINER_TYPE** - The type of container that holds the control. This can be **BOX, SWITCH**, or **SUBPANEL**, but it must agree with the type of the container name.

- **PUSH_ACTION** - This is what happens when the control button is pushed. **PUSH_ACTION** is just one of several possible actions. For more information, see the **dtwm** man page.

To remove a control button from the front panel, type a pound sign (#) in the left-most column of the **CONTROL** definition line. The (#) turns the control specification into a comment line.

Add a control button by editing the **dtwm.fp** file:

1. Copy **dtwm.fp** from **/usr/dt/appconfig/types/C** to **/etc/dt/appconfig/types/C**.
2. Add the new control definition using the following format:

```
CONTROL NewControl
{
TYPE           icon
CONTAINER_NAME Top
CONTAINER_TYPE BOX
ICON           NewControlBitmap
PUSH_ACTION    NewControlExecutable
}
```

Action and File Types have their own peculiarities in that each has a couple of parts, and each must live in its own directory location. These peculiarities will become clear as we create an action on the subpanel. The following are the recommended locations in which to create an action or file type definition:

- Create a completely new file in the **/etc/dt/appconfig/types** directory. This file has a system-wide influence. Remember, the file must end with the **.dt** extension.

- Copy **user-prefs.dt** from **/usr/dt/appconfig/types** to the **/etc/dt/appconfig/types** directory and insert the definition there for system-wide use.

- Copy **user-prefs.vf** to **$HOME/.dt/types** and insert the definition there for individual users.

A typical action has the following syntax:

```
ACTION      ActionName
{
 TYPE      type
 keyword   value
 keyword   value
}
```

For example, here's a FrameMaker action:

```
ACTION     FRAME
{
  TYPE     COMMAND
  WINDOW-TYPE NO-STDIO
  EXEC-STRING /nfs/hpcvxmk6/usr/frame/bin/maker
}
```

A typical data type has the following syntax:

```
DATA_ATTRIBUTES        AttributesName
{
  keyword       value
  keyword       value
  ACTIONS       action, action
}
DATA_CRITERIA
{
 DATA_ATTRIBUTES AttributesName
 keyword        value
```

```
 keyword        value
}
```

Note that all definitions have the following general syntax:

```
KEYWORD value
```

Notice that a data type definition is actually in two parts: an attribute part and a criteria part. The attribute portion of the data type definition specifies the look of the datatype; the criteria portion specifies the behavior of the data type.

For example, here's a file type for FrameMaker files that uses the FRAME action:

```
DATA_ATTRIBUTES    FRAME_Docs
{
  DESCRIPTION     This file type is for FrameMaker documents.
  ICON             makerIcon
  ACTIONS          FRAME
}
DATA_CRITERIA
{
DATA_ATTRIBUTES_NAME  FRAME_Docs
NAME_PATTERN  *.fm
MODE   f
}
```

You can create actions and file types from scratch using these formats. However, the easiest way to create an action is to use the **CreateAction** tool. **CreateAction** is located in the Desktop Applications folder of the Applications Manager and presents you with a fill-in-the-blank dialog box that guides you through creating an **action.dt** file containing the action definition. You can then move this file to the appropriate directory for the range of influence you want the action to have: **/etc/dt/appconfig/types** for a system-wide influence; **$HOME/.dt/types** for individual users.

Creating a New Icon and Action

Creating an icon is a challenging task. We could use the Icon Editor found in Desktop_Apps to create a new icon, or we could find a picture we like and use it. One thing to be careful of when pulling in a picture, in order for it to be seen correctly on the front panel, is that it has to be no larger than 32x32 pixels or only a portion of the icon will be displayed. Viewing the picture in the Icon Editor shows you the size of the picture. You may want to search through the application directories for useful icons or pull one down from the Internet.

For this entire example, I'm going to use the Instant Information software that comes with HP-UX. This is the manual set on the CD. Other application software will work just as well; just make sure that your paths correspond with the software you are using. Also, I'm going to use a fictitious user's home directory: **/home/admin1**.

1. **Bring up the Icon Editor** from the Desktop_Apps (HP-UX), Desktoptools (AIX), or from the Desktop_Tools, on Solaris all in the Application Manager.
2. Choose **File** -> **Open**.
3. Enter a path or folder name: **/opt/dynatext/data/bitmaps**.
4. Enter the file name: **logoicon.bm**
5. Choose **Open**.

   We'll see the icon and that it is indeed 32x32. Now we need to save the icon to our own **.dt** directory.
6. Choose **File** -> **Save As**.
7. Enter path or folder name: **/$HOME/.dt/icons**. For the admin1 user, that would be in **/home/admin1/.dt/icons**.

   If we were going to do this task globally, we'd put this in **/etc/dt/appconfig/types/C**.
8. Leave the file name as is.
9. **Save**.

Now that we have the icon, we need to create an action file and a description file to go with it.

1. **Using your favorite Text Editor**, enter the following:

   ```
   ACTION instinfo
   {
   LABEL          instinfo
   TYPE           COMMAND
   WINDOW_TYPE    NO_STDIO
   EXEC_STRING    /opt/dynatext/bin/dynatext
   DESCRIPTION    This action starts Instant Information
   }
   ```

   The LABEL is the name of the action, the TYPE is a command, the WINDOW_TYPE is none (no standard I/O or NO_STDIO) because the application has its own window, EXEC_STRING is the command to be executed, and the DESCRIPTION is just a description of what this action does. Make sure that the NO_STDIO has an underscore and not a dash, and don't forget the last }. I've done both of these and then had fun trying to figure out why the action either didn't appear to exist or, if it did appear, why it wouldn't work.

2. **Save this file as /$HOME/.dt/types/instinfo.dt**. For the admin1 user, that would be **/home/admin1/.dt/types/instinfo.dt**. If this were a global configuration, we'd save the file as **/etc/dt/appconfig/types/C/instinfo.dt**.

   Now that we have an action, we need to create a description file. Note that this is a new file with just these two lines in it. The contents of this file are irrelevant, but the permissions *must* include executable.

3. Again, **using your favorite editor**, enter:

   ```
   ACTION instinfo
   DESCRIPTION This action starts Instant Information
   ```

4. **Save this file as /$HOME/.dt/appmanager/instinfo**. For the admin1 user, that would be **/home/admin1/.dt/appmanager/instinfo**. If this were a global configuration, we'd save the file as **/etc/dt/appconfig/appmanager/instinfo**.

5. Change the permissions to include execute as follows:

   ```
   chmod 555 /$HOME/.dt/appmanager/instinfo
   ```

Now it's time to modify the front panel to include the icon and action we just created. The front panel file, **dtwm.fp**, is located in the /**usr/dt/appconfig/types/C** directory. Because we don't want to overwrite the system file, we need to copy it locally and then modify it for our use.

1. Copy this to **/$HOME/.dt/types/dtwm.fp** for AIX and HP-UX and to **/$HOME/.dt/types/fp-dynamic/dtwm.fp** for Solaris. For the admin1 user, that would be:

   ```
   cp    /usr/dt/appconfig/types/C/dtwm.fp          /home/admin1/.dt/types/dtwm.fp
   ```

   or for Solaris:

   ```
   cp   /usr/dt/appconfig/types/C/dtwm.fp    /home/admin1/.dt/types/fp-dynamic/dtwm.fp
   ```

2. **Using your favorite editor**, modify the local **dtwm.fp** file.

   As we look at the file, we notice that it is in the same order as the front panel is displayed. The clock is the first CONTROL in the file and the first item on the front panel. Also notice that the POSITION_HINTS is 1. Date is next and so is POSITION_HINTS 2. What we want to do is put our new icon and action after POSITION_HINTS 4, the TextEditor CONTROL.

3. Go down just past the } ending CONTROL TextEditor and before CONTROL Mail. At this point, **insert the following exactly as shown below**. Make sure that the uppercase letters are capitalized and the lowercase letters aren't.

   ```
   CONTROL Info
   {
    TYPE            icon
    CONTAINER_NAME   Top
   ```

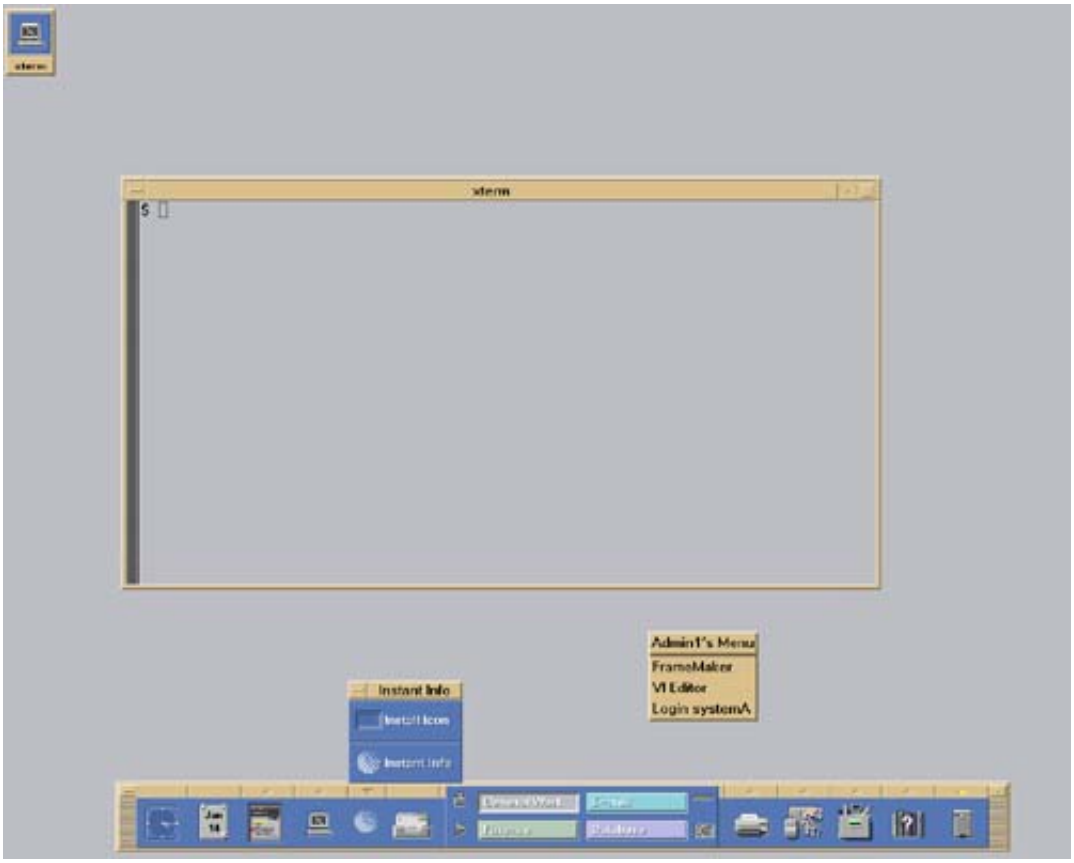
```
 CONTAINER_TYPE    BOX
 POSITION_HINTS    5
 ICON              logoicon.bm
 LABEL             Instant Info
 PUSH_ACTION       instinfo
}
```

4. Now be careful; this part is tricky. The next items have to have their POSITION_HINTS renumbered. But we are going to **renumber only the next eight items, beginning with Mail and ending with Trash**. Instead of 5 through 12, these are going to become 6 through 13.

5. **Save** the file.

   Now, let's restart the Workspace Manager. If we did everything right, we'll have a new icon, which, when clicked, will bring up Instant Information, shown in Figure 14-6.

Figure 14-6. Workspace Menu and New Subpanel and Action



6. Position the mouse over a blank area on the workspace and press the right mouse button. Select **Restart Workspace Manager**.

A couple of things to remember when creating actions: first and foremost, make sure that the PUSH_ACTION and the file names are the same. That similarity is how they find each other. Make sure the that action file ends with **.dt** and make sure that the description file is executable. If any of these are wrong, the action either won't work or won't appear.

To avoid a lot of typing, sometimes the easiest approach is just to copy an existing definition and insert it where you want your new control to be and then modify it. As you move down the list of control definitions, you're moving from left to right across the front panel (notice that the POSITION_HINTS value increases in each definition). So if you want your new control to be to the right of the date on the front panel, you insert the control on the line below "date" and add a POSITION_HINTS 3 line to your definition; if you wanted your new control to be to the left of "date," insert the control on the line above "date" with a POSITION_HINTS of 1.

The new control definition can be located anywhere in the list of control definitions. The POSITION_HINTS line keeps it from getting inadvertently bumped to a new position. It's still a good idea to copy an existing definition and avoid extra typing; it reduces the chance of typing mistakes. And don't forget to include the curly braces.

Using Different Fonts

Although CDE fonts have been carefully selected for readability, you may have valid reasons to prefer other fonts. To make your fonts available system-wide throughout the CDE environment, put them in **/etc/dt/app-defaults/Dtstyle** so that they will appear in the style manager's font dialog box. To make fonts available only for a particular X client application, specify the font in the **app-defaults** file for the application. Just remember, this overrides the fonts in the style manager.

The font dialog box can contain a maximum of seven font sizes. You can adjust this number downward by resetting the value of **Dtstyle*NumFonts** in **/etc/dt/app-defaults/Dtstyle**; however, you can't increase the number higher than seven.

The Font Dialog section of the **Dtstyle** configuration file has seven **SystemFont** resources and seven **UserFont** resources. Again, you can have fewer than seven system and seven user fonts, but you can't have more.

To specify fonts for a particular application, use the ***FontList** resource in the **app-defaults** file for the application.

To modify font resources on an individual user basis, you can use the **EditResources** action as described in the earlier section "Changing the Front Panel in Other Ways."

Changing the Login Messages

One of the nice things about CDE is the ability to modify so many parts. You can customize individual login accounts or the entire system. By customizing the login screen, you can show those about to log in the name of the system they are accessing, the company logo, and a personalized greeting. These modifications take place in the **Xresources** file:

1. As we've done already with the **dtwm.fp** and **dtwmrc** files, we will need to copy the system file from **/usr/dt/config/C** to **/etc/dt/config/C** as follows:

   ```
   cp /usr/dt/config/C/Xresources                    /etc/dt/config/C/Xresources
   ```

2. **Go into your favorite editor** and bring up the **Xresources** file so that it can be edited.
3. **Go to the GREETING area**. Here we'll find the following lines:

   ```
   !!Dtlogin*greeting.labelString:                   Welcome to %LocalHost%
   !!Dtlogin*greetingpersLabelString:                Welcome %s
   ```

The first line is the message on the initial login screen. Let's change that so that it welcomes us to our company, ABC, Inc.:

1. **Remove the comment notations**. Unlike shell scripts that most of us are used to, the Xresources file uses two exclamation points as comment notation. Remove the !!.
2. Next, **modify "Welcome to %LocalHost%"**. The %LocalHost% variable is replaced with our system name in the login screen. The line should look like this:

   ```
   Dtlogin*greeting.labelString:          ABC, Inc, Welcomes You to %LocalHost%
   ```

3. Next let's **change the second line** to include the department that this system is dedicated to: finance. This second line shows what is displayed when we are prompted for our password. The %s variable is our user name. The line should now look like this:

   ```
   Dtlogin*greetingpersLabelString:          The Finance Department Welcomes %s
   ```

4. **Save the file**.
5. **Now log out and back in**. We should see the changes in the login screen. We didn't need to "reload" the file, because the act of logging out and back in does that action.

Changing the Login Picture

Adding a new picture to the login screen is easy if you know one thing. The file has to be a bitmap (.bm) or pixmap (.pm) file. A bitmap file is black and white, and the pixmap file is color. I've tried using other kinds of pictures (.gif and .jpg formats), but they just don't display. The good news is that these can be imported from other systems or the Internet for our use. To make things simple, we're going to use one already on the system. A bitmap showing a birthday cake was found in **/usr/lib/X11/bitmaps** on an HP-UX workstation. However, I have also successfully pulled down pictures of flowers, the Grand Canyon, and country music singers (note: Donna wrote this chapter, not Marty) from the Internet and put them on my system login screen.

1. Once more, let's go into our favorite editor and modify **/etc/dt/config/C/Xresources**.
2. **Go to the MISC area**. ere we'll find the following lines:

   ```
   !!Dtlogin*logo*bitmapFile:                    < bitmap or pixmap file >
   ```

3. **Delete the leading !!**, which are the comment designators.
4. **Replace < bitmap or pixmap file >** with the name of the bitmap file using the entire path location. The line should look as follows:

   ```
   Dtlogin*logo*bitmapFile:                    /usr/lib/X11/bitmap/cake.bm
   ```

5. **Save the file**.
6. **Now log out and back in**. We should see the birthday cake in the login screen. Again, we didn't need to "reload" the file, because the act of logging out and back in does that task.

Now that we've seen how easily we can make some simple customizations in CDE for our end-users, we should be able to take this knowledge and really make their CDE environments a productive and friendly place to work.

CDE - Advanced Topics

The Relationship among X, Motif, and CDE X, OSF/Motif, and CDE are enabling framework technologies. Taken together, X, Motif, and CDE make up the three graphical layers on top of the operating system and hardware platform.

The GUI layers provide increasingly richer ease-of-use functions in a progressive series of layers that buffer the end user from the "user-hostile," character-based interface of the operating system layer.

The X Window System

The X Window System consists of the following:

- Xlib - Low-level library for programming window manipulation; graphics capabilities such as line drawing and text placement; controlling display output, mouse, and keyboard input; and application network transparency.
- Xt Intrinsics - Higher-level library for programming widgets and gadgets (graphical controls components like menus, scrollbars, and push buttons).
- Display servers - Hardware-specific programs, one per display, that manage the graphical input and output.
- Interclient communication conventions (ICCC) - A manual specifying standards for how X client programs should communicate with each other.
- Configuration files - One configuration file that specifies the default session to start (**sys.x11start**) and another specifying values for resources used to shape the X environment (**sys.Xdefaults**).

Through these mechanisms, X provides the standard upon which the graphical part of the network-oriented, client/server, distributed computing paradigm is based. A knowledge of **Xlib** and the **Xt** Intrinsics is important for programming in X and for programming at the Motif level. For system administrators, however, as long as the display servers work and X client applications are ICCC-compliant, you shouldn't need to delve into the X layer. CDE enables you to view X pretty much as part of "all that underlying technological infrastructure stuff" and focus on developing appropriate configurations of CDE to meet your users' work contexts.

Motif

Motif consists of the following:

- mwm window manager - Executable program that provides Motif-based window frames, window management, and a workspace menu in the X environment.
- Motif widget toolkit - Higher-level library of widgets and gadgets, the graphical components used to control the user environment.
- Motif style guide - A manual defining the Motif appearance and behavior for programmers.
- Configuration files - The **system.mwmrc** file containing configuration information for the workspace menu and key and button bindings. Resources for the window manager are in **mwm** in the **/usr/lib/X11/app-defaults** directory.

Motif provides the window manager for the end-user, the widget toolkit for application developers, and the

style guide to help developers design and build proper Motif-conformant applications. As with X, system administrators can view Motif mostly as "programmer's stuff," part of the underlying infrastructure, and focus on developing appropriate CDE configuration files.

CDE

As we have already seen, CDE consists of the following:

- Workspace Manager - Executable program that provides Motif-based window frames, window management, a workspace menu, and the front panel.
- File Manager - Program that iconically manages files and directories through direct manipulation.
- Style Manager - Container of dialog boxes that control elements of the CDE environment, such as workspace color and fonts.
- Help Manager - This program provides context-sensitive help text on CDE components.
- Login Manager - Daemon-like application that handles login and password verification.
- Session Manager - Manager that handles saving and restoring user sessions.
- Application Manager - Manager that registers and keeps track of applications in the CDE environment.
- Configuration Files - A big bunch, most of which you can avoid dealing with (see the next section).

CDE also provides a number of basic, end-user productivity-enhancing applications. In general, CDE provides a graphical environment into which users, or you, their system administrator, can incorporate the software tools needed to do their work.

[Team LiB]

X, Motif, and CDE Configuration Files

X, Motif, and CDE all use configuration files to shape their appearance and behavior. Elements of appearance and behavior such as foreground color, keyboard focus policy, and client decoration are resources that can be controlled by values in the appropriate configuration file. In X, Motif, and CDE, the word "resource" has a special meaning. It doesn't refer to vague natural resources or generic system resources, but to rather specific elements of appearance and behavior. Some examples are the **foreground** resource, the **keyboardFocusPolicy** resource, and the **clientDecoration** resource. For example, the foreground color could be black, keyboard focus policy could be explicit, and client decoration could be plus-title (title bar only). These would appear in some appropriate configuration file such as the following:

```
*foreground: black

*keyboardFocusPolicy: explicit

*clientDecoration: +title
```

Which configuration file these resources appear in depends on the scope of the effect desired (system-wide or individual user) and the graphical interface level being used (X, Motif, or CDE).

X Configuration Files

The X Window System has the following configuration files:

```
sys.x11start

sys.Xdefaults

system.mwmrc

X*screens

X*devices

X*pointerkey
```

By convention, these files are located in the **/usr/lib/X11** directory; however, I have noticed that many systems have eliminated this directory and moved many of the X-related files elsewhere in the system. In addition, each X client application has its own app-defaults configuration file located, also by convention, in the **/usr/lib/X11/app-defaults** directory. Although six files are listed above, unless you're configuring a workstation for multiple-display screens (X*screens), multiple-input devices (X*devices), or keyboard-only pointer navigation (X*pointerkey), you'll typically need to work with only **sys.x11start, sys.Xdefaults**, and **system.mwmrc**.

The **sys.x11start** file was a script used to start X and X clients before the advent of CDE. System administrators or knowledgeable users modified **sys.x11start** so that the appropriate mix of X clients started "automatically." The **sys.Xdefaults** file was read as X started to obtain values for various appearance and behavior resources. Modifications to **sys.Xdefaults** ensured that the X environment and clients had the proper appearance and behavior. **system.mwmrc** contained the configuration of the Workspace menu and button and key bindings. **system.mwmrc** has been replaced by the Motif version also, **system.mwmrc**.

Motif Configuration Files

Motif added only one new configuration file to the X list: **system.mwmrc**.

By convention, this file is kept with the X configuration files in **/usr/lib/X11**. Actually, this file isn't new; it is the Motif version of **system.mwmrc,** which simply replaced **system.mwmrc** in Motif environments.

Whereas X brought network and interclient communication standards to the graphical user interface, Motif brought a standard for appearance and behavior, the standard originally defined in IBM's System Application Architecture Common User Access (SAACUA), which forms the basis of most PC-based GUIs. Thus, push buttons and scroll bars have a defined look and a defined behavior, and double-clicking always causes the default action to happen.

From a programmer's point of view, the Motif Widget Toolkit represents quite an advance over programming in "raw" X. From a user's or system administrator's point of view, the Motif user environment is about the same as the X environment, except that the **mwm** Window Manager is replaced with the Motif window manager.

CDE Configuration Files

It is possible to point to over 80 files that, in one way or another, contribute to configuring some aspect of CDE. By convention, these files reside in the **/usr/dt** directory. However, if you remove from this list such files as those that:

- Configure CDE applications as opposed to the environment itself
- Establish default actions and datatype definitions that, although you create your own definitions in separate files, you never modify
- Are CDE working files and should not be customized
- Are more appropriately associated with configuring the UNIX, X, and Motif environments underlying CDE, including the various shell environments, then CDE has approximately 19 configuration files, as shown in Table 14-1

Table 14-1. CDE Configuration Files

* .Xauthority	* sys.font	* Xresources
* .Xdefaults	* sys.resources	* Xservers
* .dtprofile	* sys.sessions	* Xsession
* dtwm.fp * dt.wmrc	* Xaccess * Xconfig	* Xsetup * Xstartup
* sys.dtprofile	* Xfailsafe	
* sys.dtwmrc	* Xreset	

Although 19 configuration files are still a lot, don't be alarmed by the number. You won't need to modify many of them, and can ignore a couple that you modify once and then forget. You need to understand in depth for periodic modification only one or two, perhaps a system-wide ***.dt** file for custom actions and datatypes or maybe **dtwm.fp**, if you are required to modify the front panel on a regular basis for some reason.

Still, configuring CDE is not something you want to start hacking at without a little preparation and a good idea of what you want to accomplish. All CDE configuration files are pretty well commented, so a good first step is to print the ones you want to modify.

Table 14-2 organizes CDE configuration files according to their content and the breadth of their influence:

Table 14-2. CDE Configuration File Influence

Table 14-2. CDE Configuration File Influence

Nature of Configuration File	System-Wide Influence	User's Personal Influence
Environment Variables	sys.dtprofile Xconfig Xsession	.dtprofile
Appearance & Behavior Resources	sys.resources Xconfig Xresources sys.fonts	.Xdefaults
File Types & Action Definitions	misc *.dt files	user-prefs.dt
Client Startup at Login	sys.sessions Xstartup Xsession Xreset Xfailsafe	.xsession sessionetc
Workspace Manager & Front Panel	sys.dtwmrc dtwm.fp	dtwmrc user-prefs.fp
Clients/Servers & Access	Xaccess Xservers	.Xauthority

The file **sys.dtwmrc** controls the configuration of the Workspace Manager at the system level. This includes all of the following:

Workspace Menu	A menu that displays when mouse button 3 is pressed while the mouse pointer is over the workspace backdrop.
Button Bindings	Definitions of what action happens when a particular mouse button is pressed or released while the mouse pointer is over a particular area (frame, icon, window, or root).
Key Bindings	Definitions of what action happens when a particular key or key sequence is pressed while the mouse pointer is over a particular area (frame, icon, window, or root).

Unlike configuration files for X or Motif, **sys.dtwmrc** does not control the following configuration elements:

Front Panel	The box, usually at the bottom of the workspace, that contains commonly referenced indicators and frequently used graphical controls, including a six-button workspace switch.
Slideup Subpanels	Menus that slide up from the front panel at various locations to provide more functionality without consuming more screen space.

Instead, to avoid a massively large and overly complex configuration file, these elements were separated into their own configuration file in CDE, **dtwm.fp**.

Some front panel configuration elements, like the number of workspaces and their arrangement in the workspace switch, are controlled through resources in a **sys.resources, dt.resources**, or **.Xdefaults** file. Like other Workspace Manager configuration files, **sys.dtwmrc** can be copied to a user's home directory, actually to **$HOME/.dt/** as **dtwmrc**, and modified to personalize the user's environment beyond the system-wide configuration of **sys.dtwmrc**.

The **sys.resources** file is one of those files you might modify once, and then never again. The

dt.resources file is one of those files you won't ever need to modify and can ignore. The **.Xdefaults** file is one that you or your users may modify on occasion.

The **sys.resources** file is where you put any non-default resources that you want in effect when a brand new user logs into CDE for the very first time. For example, as system administrator, you may want your users to have a CDE front panel with prenamed workspaces, special colors, particular fonts, or application windows in certain locations. After the first-time login, **sys.resources** is ignored in favor of **dt.resources**. This file, **dt.resources**, resides in **$HOME/.dt/sessions/current** (or **$HOME/.dt/sessions/home** when the home session is restored) and is created automatically by CDE. You can consider it a CDE working file and forget about it. The **.Xdefaults** file is where you or an end-user would list X resources specific to the user's personal CDE environment. **sys.resources, dt.resources**, and **.Xdefaults** contain a list of resources and their values.

The **sys.sessions** file controls which clients start the very first time a new user logs into CDE. The **dt.sessions** file is to **sys.sessions** as **dt.resources** is to **sys.resources**.

It may be efficient to configure CDE to start particular applications for your users. You would specify these applications in **sys.sessions**. When a new user logs in for the first time, the CDE environment includes the specified clients. At the end of this first session by logging out, the remaining clients would be recorded in **$HOME/.dt/sessions/current** for CDE (**$HOME/.dt/sessions/home** when the home session is restored).

The **sys.dtprofile** file is a template that is automatically copied at first login into each new user's home directory as **.dtprofile. sys.dtprofile** replaces **.profile** or **.login** in the CDE environment (although either **.profile** or **.login** can be sourced in **.dtprofile** by removing the **#** comment symbol in front of **DTSOURCEPROFILE=true**). The **.dtprofile** file holds the personal environment variables that would, in a character-based environment, be found in **.profile** or **.login**. Use **.dtprofile** to avoid the interference that terminal I/O commands cause to CDE's graphical environment.

The CDE login manager, **dtlogin**, presets the following environment variables to default values:

DISPLAY	The name of the local display
EDITOR	The default text editor
HOME	The user's home directory as specified in **/etc/passwd**
KBD_LANG	The current language of the keyboard
LANG	The current NLS language
LC_ALL	The value of LANG
LC_MESSAGES	The value of LANG
LOGNAME	The user's login name as specified in **/etc/passwd**
MAIL	The default file for mail (usually, **/var/mail/$USER**)
PATH	The default directories to search for files and applications
USER	The user name
SHELL	The default shell as specified in **/etc/passwd**
TERM	The default terminal emulation
TZ	The time zone in effect

Variations to these default values belong in each user's **.dtprofile**. Additional environment variables can be added as needed to shape the user's environment to the needs of the work context. Just beware of using commands that cause any terminal I/O.

Like **.dtprofile**, **Xsession** is a shell script that sets user environment variables. The environment variables in **Xsession** apply system-wide. The environment variables in **.dtprofile** apply only to a user's personal environment. Furthermore, Because the login manager runs **Xsession** after the X server has started, the variables in **Xsession** are not available to the X server. Variables typically set in **Xsession** include the following:

EDITOR	The default text editor.
KBD_LANG	The language of the keyboard (usually, set to the value of $LANG).
TERM	The default terminal emulation.
MAIL	The default file for mail, which is usually **/var/mail/$USER**.
DTHELPSEARCHPATH	The locations to search for CDE help files.
DTAPPSEARCHPATH	The locations to search for applications registered with the CDE application manager.
DTDATABASESEARCHPATH	The locations to search for additional action and datatype definitions.
XMICONSEARCHPATH	The locations to search for additional icons.
XMICONBMSEARCHPATH	Same as above.

As an example, suppose that you are the system administrator for several mixed workstation and X terminal clusters located at a single site. As usually happens, most users have grown accustomed to certain text editors. Some like **vi**, others prefer **emacs**, and a couple wouldn't be caught dead without **dmx**. An easy way to provide each user with his or her favored text editor would be to reset their EDITOR variable to the appropriate value in the individual **.dtprofile** files.

Xconfig contains resources that control the behavior of **dtlogin** and it also provides a place to specify the locations for any other **dtlogin** configuration files you create. The **Xconfig** file works on a system-wide basis, so it's one of those files that you modify only once and then forget about. When, during login, **Xconfig** is run, several CDE configuration files get referenced: **Xaccess**, **Xservers**, **Xresources**, **Xstartup**, **Xsession**, **Xreset**, and **Xfailsafe**. Like **Xconfig** itself, most of these files are the type that you modify once when installing CDE and then, unless the network topology changes, you never deal with again.

Xaccess, as the name implies, is a remote display access control file. **Xaccess** contains a list of the host names allowed or denied XDMCP connection access to the local computer. For example, when an X terminal requests login service, **dtlogin** consults the **Xaccess** file to determine whether service should be granted.

The primary use of the **Xservers** file is to list the display screens on the local system that **dtlogin** is responsible for managing. **dtlogin** reads the **Xservers** file and starts an X server for each display listed there. It then starts a child **dtlogin** process to manage the server and display the login screen. Note that **dtlogin** works only locally; **dtlogin** can't start an X server on a remote system or X terminal. For remote display servers, some other mechanism must be used to start the server, which then uses the X Display Management Control Protocol (XDMCP) to request a login screen from **dtlogin**.

The **Xservers** file is another of those files that you may spend some time with initially and then, unless the topology of your network changes, never deal with again. When do you use **Xservers**? When a display doesn't match the default configuration. The default configuration assumes that each system has a single bitmap display and is the system console. X terminals, multiple displays (heads), multiple screens, and Starbase applications all require configuration lines in the **Xservers** file.

The **Xresources** file contains the list of resources that control the appearance and behavior of the login screen. After you substitute your company's logo for the CDE logo and change the fonts and colors, you'll probably never have to deal with **Xresources** again (unless your company changes its logo).

Xstartup is a system-wide configuration file executed by the login manager, from which it receives several environment variables:

DISPLAY	The name of the local display.
USER	The login name of the user.
HOME	The user's home directory.
PATH	The value of the **systemPath** resource in **Xconfig**.
SHELL	The value of the **systemShell** resource in **Xconfig**.
XAUTHORITY	The file to access for authority permissions.
TZ	The local time zone.

Because it can execute scripts and start clients on a system-wide basis, **Xstartup** is similar to **sys.sessions**. The difference is that **Xstartup** runs as root. Thus, modifications to **Xstartup** should be reserved for such actions as mounting file systems.

Xreset is a system-wide companion script to **Xstartup**. It runs as root and essentially undoes what **Xstartup** put in motion.

The **Xfailsafe** file contains customizations to the standard failsafe session. The failsafe session provides a way to correct improper CDE sessions caused by errors in the login and session configuration files. As such, **Xfailsafe** is something that your users are not ever going to use, but you can make your life a little easier with a few judicious customizations.

The **sessionetc** file resides in a user's **.dt/sessions** directory and personalizes that user's CDE session. **sessionetc** handles the starting of additional X clients like **sys**.**session**, but on a per-user basis, as opposed to system-wide. Although **dt.session** also starts clients on a per-user basis, the clients are those of the default or current session. **dt.session** resides in **.dt/session/current**. **sessionetc**, which resides in **.dt/session**, and should contain only those clients that are not automatically restored. Typically, these are clients that do not set the **WM_COMMAND** properly, so the session manager can't save or restore them; thus, they need to be restarted in **sessionetc**.

The **sys.font** file contains the system-wide, default session font configuration. These default fonts were based on usability studies, so **sys.font** is a file you may never change. However, should you encounter a situation that requires a different mix of fonts on a system-wide basis, this is where you'd change them. Note that the font resources and values mentioned in **sys.font** must match exactly the default font resources specified in the **/usr/dt/app-defaults/C/Dtstyle** file.

CDE has a bunch of files that specify CDE action and data type definitions. All these files end with the file extension ***.dt**. A ***.dt** ("dt" for "desk top") contains both data type and action definitions. The default ***.dt** files are in **/usr/dt/appconfig/types/C** and act on a system-wide basis. Similarly, **user-prefs.dt**, the master copy of which is also located in **/usr/dt/appconfig/types/C**, is used at the personal user level.

The **.Xauthority** file is a user-specific configuration file containing authorization information needed by clients that require an authorization mechanism to connect to the server.

CDE Configuration File Locations

Where CDE looks for particular configuration files depends on the nature of the configuration files, principally what the files configure and how wide their influence is. Table 14-3 shows the location of system and user configuration files based on the nature of the file content.

For each of the default system-wide file locations listed in Table 13-3, a corresponding location exists for custom system-wide configuration files. These custom files should be located in the appropriate subdirectory under **/etc/dt**. The basic procedure is to copy the file you need to customize from **/usr/dt/something** to **/etc/dt/something** and then do your modifications there. For example, to change the default logo in **Xresources**, copy **/usr/dt/config/C/Xresources** to **/etc/dt/config/C/Xresources**, open **/etc/dt/config/C/Xresources**, and make your changes.

Table 14-3. CDE System and User Configuration Files

Nature of Configuration File	System-Wide Influence	User's Personal Influence
Environment Variables	/usr/dt/config/	$HOME/
Appearance & Behavior Resources	/usr/dt/config/C	$HOME/.dt/
	/usr/dt/app-defaults/C	$HOME/.dt/sessions/current/
		$HOME/.dt/sessions/home/
File Types & Action Definitions	/usr/dt/appconfig/types/C	$HOME/.dt/types
Client Startup at Login	/usr/dt/config/	$HOME/.dt/session/
	/usr/dt/config/C	$HOME/.dt/session/current/

		$HOME/.dt/session/home/
Workspace Manager	/usr/dt/config	$HOME/.dt/

This is an important point. Files located under **/usr/dt** are considered CDE system files and will be overwritten during updates. Thus, any customizations you do there will be lost. Make all modifications to system-wide configuration files in **/etc/dt** and its subdirectories.

[Team LiB]

How Configuration Files Play Together

From the material covered so far, you've probably concluded correctly that CDE configuration files aren't something to go hacking at without a plan - a well-thought-out plan. You've probably figured out that the element you want to configure and the breadth of influence you want it to have determine which configuration file you modify.

For instance, if you wanted to set an environment variable, you have a choice of four configuration files: **sys.dtprofile, Xconfig, Xsession**, and **.dtprofile**. But if you want to set environment variables that affect only a particular user, your choice immediately narrows to a single file, **.dtprofile**.

Now the only remaining piece of the puzzle is to understand the order in which CDE reads its configuration files. When a configuration element (an environment variable, resource, action, or data type) is specified twice but with different values, you obviously want the correct value used and the incorrect value ignored.

The following rules apply:

- For environment variables, the last specified value is used.
- For resources, the last specified value is used. However, this is influenced by specificity. Thus, **emacs*foreground** takes precedence over just ***foreground** for *emacs* clients, regardless of the order in which the resources were encountered.
- For actions, the first specified is used.
- For data types, the first specified is used.

Table 14-4 illustrates which specification is used when CDE reads multiple specifications of configuration elements in its configuration files:

Table 14-4. What CDE Uses for Configuration

Configuration Element	Element Used
resource	last encountered or most specific
environment	last encountered
action	first encountered
file type	first encountered

Put in terms of scope, a user configuration file overrides a system-wide configuration file. Looking at the order of precedence of just system-wide configuration files, the files in **/etc/dt** have precedence over those in **/usr/dt**, so global custom configurations have precedence over the CDE default configuration. And **$HOME/.dt** files take precedence over those in **/etc/dt.**

For resources, the elements used to specify a GUI's appearance and behavior, CDE sets values according to the following priorities:

1. **Command line** - When you start a client from the command line, options listed on the command line have top priority.
2. **Xresources, .Xdefaults, dt.resources, sys.resources,** - When CDE starts, it reads these resource configuration files to determine the value of X resources to use for the session.
3. **RESOURCE MANAGER** - Resources already in the property **RESOURCE_MANAGER** may affect an application that is just starting.

4. **app-defaults** - Specifies "default" resource values that differ from built-in resource values.
5. **built-in defaults** - Default resources that are "hard-coded" have the lowest priority.

Specific resource specifications take precedence over general resource specifications. For example, suppose that you want a certain font in your text entry areas. You could correctly specify a ***FontList** resource in your personal **.Xdefaults** file, only to have it overwritten by an ***XmText*FontList** in an **app-defaults** file. Although **app-defaults** is of lower priority than **.Xdefaults**, the resource specification set there is more specific, so it takes precedence.

For environment variables, CDE sets values according to the following priorities:

1. **$HOME/.dtprofile** - User-specific variables have top priority.
2. **/etc/dt/config/C/Xsession** - Custom system-wide variables not read by X server.
3. **/etc/dt/config/C/Xconfig** - Custom system-wide variables read by X server.
4. **/usr/dt/config/C/Xsession** - Default system-wide variables not read by X server.
5. **/usr/dt/config/C/Xconfig** - Default system-wide variables read by X server.
6. **/usr/dt/bin/dtlogin** - Built-in default variables have the lowest priority.

For data type and action definitions, CDE looks for **.dt** files according to the following priority:

1. $HOME/.dt/types
2. /etc/dt/appconfig/types/C
3. /usr/dt/appconfig/types/C

Remember that for data types or actions, the first value that it finds is the one it uses. So if you just can't get a file type or action to work, check for a duplicate entry earlier in the file or for an entry in a file with higher priority. Note also that the environment variable DTDATABASESEARCHPATH can be set either in **/etc/dt/config/Xsession** or **$HOME/.dtprofile**, to add directories where CDE can search for file type and action definition information.

[Team LiB]

Specifying Appearance and Behavior

You need to know only two tricks to specifying appearance and behavior resources in configuration files. The first is to specify the resource and its value correctly. The second is to specify the resource and value in the correct configuration file.

Two caveats involve colors and fonts. The CDE style manager provides a graphical interface for modifying colors and fonts. However, if you specify an application's color or font directly, this specification will override the ability of the style manager to manage that resource for the application.

Typical ways to specify a color or font directly include the following:

- Type the specification on the command line as a startup option.
- Include the specification in the application's **app-defaults** file.
- Use the **xrdb** utility to add resources for the application to the resource database.



The Sequence of Events When CDE Starts

The following section is a blow-by-blow account of what happens when a user logs into CDE. In this particular account, assume a distributed topology like a diskless cluster. The account begins with the boot of the hub system and nodes in step 1. By step 4, X servers are running on each node and login screens are being displayed. By step 6, the user is logged in. By step 11, the session manager is busy re-creating the user's session.

1. The **dtlogin** executable is started as part of the **init** process that occurs during the system boot sequence on the hub machine and each cluster node.
2. **dtlogin** reads **/usr/dt/config/Xconfig** to get a list of resources with which to configure the login process. This is where **dtlogin** first learns about files such as **Xaccess, Xservers, Xresources, Xstartup, Xsession**, and **Xreset** and gets the values of a number of appearance and behavior resources.
3. **dtlogin** reads two files in **/usr/dt/config**:
    - **Xservers** or the file identified by the **Dtlogin*servers** resource setting in **Xconfig**.
    - **Xresources** or the file identified by the **Dtlogin*resources** resource setting in **Xconfig**.
4. **dtlogin** starts an X server and a child **dtlogin** for each local display.
5. Each child **dtlogin** invokes **dtgreet**, the login screen.
6. When a login and password are validated, a child **dtlogin** sets certain environment variables to default values.
7. The child **dtlogin** runs **/usr/dt/config/Xstartup**.
8. The child **dtlogin** runs **/usr/dt/config/Xsession**.
9. **Xsession** runs **dthello**, the copyright screen.
10. **Xsession** reads **$HOME/.dtprofile**, setting any additional environment variables or overwriting those set previously by **dtlogin**.
11. The child **dtlogin** invokes the session manager, **dtsession**.
12. **dtsession** restores the appropriate session. For example, to restore the current session, **dtsession** reads **dt.resources** and **dt.session** in **$HOME/.dt/sessions/current**.

At logout, the reverse happens. The session is saved and **dtlogin** runs **/usr/dt/config/Xreset**. After **Xreset** completes, **dtlogin** again displays the login screen, as in step 4.



CDE and Performance

CDE isn't a monolithic application; it's a set of components layered on top of the operating system, the X Window System, and Motif. Each underlying layer takes its share of RAM before CDE or any other client even starts. Because of the low-level nature of these layers, the RAM they use is hardly ever regained through swapping to disk.

In some cases, operating system overhead and user application requirements restrict the amount of RAM available for a graphical user interface to little more than enough to run a window manager such as Motif. Because the CDE workspace manager and the Motif window manager take roughly the same amount of RAM, users can enjoy an enriched graphical environment with the added value of CDE's multiple workspaces at essentially no extra RAM cost over running the Motif window manager.

Tactics for Better Performance

Unless all your users have RAM-loaded powerhouses for systems, you will need to spend some time developing a performance strategy. If you conceive of performance as a bell-shaped curve, satisfaction lies on the leading edge. Your performance strategy should do everything it can to keep your users on the leading edge.

Probably the most logical approach is to start small and grow. In other words, start out with minimal user environments on all the systems on your network. Gradually add software components until you or your users begin to notice performance degradation. Then back off a little. Such an approach might take several weeks or more to evaluate, as you add components and as your users spend several days actually working in the environment to determine the effect of your changes on system performance and their frustration levels.

The most RAM-expensive pieces of CDE are the workspace manager, the session manager, and the file manager. The workspace manager is expensive because portions of it are always in RAM (assuming that you are moving windows around and switching workspaces). The CDE workspace manager is no more expensive than the Motif window manager; if you want a GUI, it's just a price you have to pay. The session manager is expensive only during logout and login, as it saves and restores sessions. The rest of the time, the session manager is dormant and gets swapped out of RAM. Saving your current work session is nice at the end of the day, but it's something to consider giving up if you want to improve your login and logout performance. The file manager is expensive because it wakes up periodically and jumps into RAM to check the status of the file system and update its file manager windows. When it jumps into RAM, it pushes something else out, for example, perhaps the desktop publishing program you're using.

Here are some other ideas that you may find useful:

Terminal Emulators	**xterms** are a little less RAM-expensive than **dtterms**. Unless you need the block mode functionality of a **dtterm, xterm** might be a better choice for terminal emulation.
Automatic Saves	Some applications automatically save data at periodic intervals. Although this feature can be beneficial, you need to evaluate its effect in light of performance. If the application is central to your users' work, fine, but if not, you might want to disable the automatic save feature.
Scroll Buffers	Large scroll buffers in terminal emulators can be a real convenience, but they can also take up a lot of RAM. Even modestly sized scroll buffers, when multiplied by three or four terminal emulators, consume a lot of RAM.
Background Bitmaps	Avoid large bitmaps; they increase the X server size. Especially avoid switching large bitmaps frequently within a session. If you are hunting for a new background, be sure to restart the X server after you've found the one you want and have included it in the proper **sessionetc** file. The most efficient bitmaps are small ones that can be repeated to tile the background.
Front Panel	Reconfigure the front panel to minimize the number of buttons. Keep just enough to meet user needs. This tactic decreases the workspace manager size in RAM and speeds login and logout.
Pathnames	Whenever possible, use absolute pathnames for bitmap specifications. Although this

approach decreases the flexibility of the system, it speeds access time.



Font Paths

You can set-up a font server on your HP-UX system to provide an Xserver with the CDE fonts required to properly display CDE on the system. Please follow the steps below on every HP-UX system you want to act as a font server. When connecting a PC using an Xserver program you can make all of the systems on which you want to run CDE font servers using the following procedure:

1. Append the following line to the end of **/usr/dt/config/xfonts/C/**

   ```
   RUN_X_FONT_SERVER=1
   ```

2. Stop any font server processes that may be running with the following command:

   ```
   /sbin/init.d/xfs stop
   ```

3. Start the font server with the following command:

   ```
   /sbin/init.d/xfs start
   ```

4. Add each font server to the font path using the following steps:

   - Copy the **Xsetup** file from the **/usr/dt/config** directory to the /**etc/dt/config/Xsetup** file.

   - Add the following line to **/etc/dt/config/Xsetup** file:

   ```
   $XDIR/xset fp+ tcp/IPADDRESS:7000 1>/dev/null
   ```

where *IPADDRESS* is the IP address of the server where the font server is running.

NOTE: If you have multiple systems running font servers on your network, you can have multiple entries (each with a separate *IPADDRESS*) in your **/etc/dt/config/Xsetup** file, as shown below:

```
$XDIR/xset fp+ tcp/11.11.11.111:7000 1>/dev/null
$XDIR/xset fp+ tcp/22.22.22.222:7000 1>/dev/null
```




CDE and Virtual Partitions

Let's start out talking about Common Desktop Environment (CDE) and Virtual Partitions. Running multiple instances of HP-UX on the same computer is just like having multiple systems. This means that your CDE configuration for users in different vPars is done just as it would be on different systems. There are configuration files that affect CDE on a system-wide basis. You'll need to set up these system-wide configuration files on each vPar to meet the specific needs of the people working in the respective vPars.

In the upcoming sections we'll modify CDE in a Virtual Partition to include a custom CDE Greeting at the login screen and customize the menu to include some Virtual Partition-specific commands.

Custom Greeting for Different Virtual Partitions

The first encounter users have with CDE on the vPars system is the login screen. Let's customize the login screen so that it shows the hostname, which is probably already configured on your CDE login screen, as well as the vPar name as part of the greeting. If your hostname and vPar name are the same, you don't need to supply the vPar name on the login screen. This is an interesting customization to CDE so I'll provide the login screen changes as an example customization.

The default information for the login screen is contained in **/usr/dt/config/C/Xresources**. We'll copy this file to **/etc/dt/config/C/Xresrouces** and then add the vPar name to the file. The portion of the file that includes the *GREETING* is shown below:

```
!!#############################################################
!!
!!  GREETING
!!


!! Dtlogin*greeting.foreground:         black
!! Dtlogin*greeting.background:         #a8a8a8
!! Dtlogin*greeting.labelString:        Welcome to %LocalHost%
!! Dtlogin*greeting.persLabelString:    Welcome %s
!! Dtlogin*greeting.alignment:          ALIGNMENT_CENTER

!!#############################################################
!!
```

The *!!* in this file indicates a comment (unlike the # that we are accustomed to using in shell programs). We could uncomment the line that shows *%LocalHost%* and add to it the Virtual Partition name as shown below:

From:

```
!! Dtlogin*greeting.labelString: Welcome to %LocalHost%
```

To:

```
Dtlogin*greeting.labelString: Welcome to %LocalHost% \n\
            This is Virtual Partition vPar Name
```

You would substitute the name of your Virtual Partition for *vPar Name* if indeed the vPar name and hostname are different. Figure 14-7 shows the modified greeting we created:

Figure 14-7. Login Screen with Update Greeting



This screen shows our updated greeting at the login screen with the hostname of *actappd1* as well as the line on which we would place the vPar name.

Menu Picks for Virtual Partitions

In this section we'll add some Virtual Partition-related commands to our CDE menu. Since most Virtual Partition work is performed by the system administrator, we'll update the menu picks for *root* only in the upcoming example. The default information for the login screen is contained in **/usr/dt/config/C/Xresources**. We'll copy this file to the home directory and **.dt** directory for root, which, on this vPar, is **/.root/.dt**. We'll rename the file **dtwmrc** after copying it. The menu portion of **/.root/.dt/dtwmrc** looks like the following before it is updated:

```
Menu DtRootMenu
{
    "Workspace Menu"                    f.title
    "Shuffle Up"                        f.circle_up
    "Shuffle Down"                      f.circle_down
    "Refresh"                           f.refresh
    "Minimize/Restore Front Panel"      f.toggle_frontpanel
     no-label                           f.separator
    "Restart Workspace Manager..."      f.restart
     no-label                           f.separator
    "Log out..."                        f.action ExitSession

}
```

We're going to add our new menu pick just after the *DtRootMenu* entry, which we see on our system as the left mouse button's "Workspace Menu." After adding our menu picks, the file now looks like the following:

```
Menu DtRootMenu
{
    "Workspace Menu"                    f.title
    "Shuffle Up"                        f.circle_up
    "Shuffle Down"                      f.circle_down
    "Refresh"                           f.refresh
    "Minimize/Restore Front Panel"      f.toggle_frontpanel
     no-label                           f.separator
    "Restart Workspace Manager..."      f.restart
     no-label                           f.separator
    "Log out..."                        f.action ExitSession


Menu vParMenu
    "vPar Menu"               f.title
    "vparstatus"              f.exec "xterm -T vparstatus -e
                                 /usr/bin/sh -c 'vparstatus; read xxx'"
    "vparstatus -v"           f.exec "xterm -geometry 80x50+830+0
                                 -sl 200 -bg white -T vparstatus_verbose
                                 -e /usr/bin/sh -c 'vparstatus -v|more; read xxx'"
    "Login vPar2"              f.exec "xterm -geometry 80x50+830+0 -sl 200 -bg white
                                 -T vPar2 -e remsh ecvpard2 &"

}
```

The **f.title** function shows that this is the menu title. **f.exec** means to execute the following string. In the first menu pick, *vparstatus*, we simply open an *xterm* window, run the **vparstatus** command, and wait for user input before we close the window with **read**. The next command uses **vparstatus -v** which has a more extensive output. For this command I open a much bigger *xterm* window and run **vparstatus -v|more** so that the user can hit the spacebar if the command results in more than one window of output. (There is a single quote around the **vparstatus** and **read** commands.) I included yet more options on the **xterm** for the remote login to the second vPar, making the terminal window very large, with lots of terminal memory and using specific colors. Our new menu now looks like that shown in Figure 14-8:

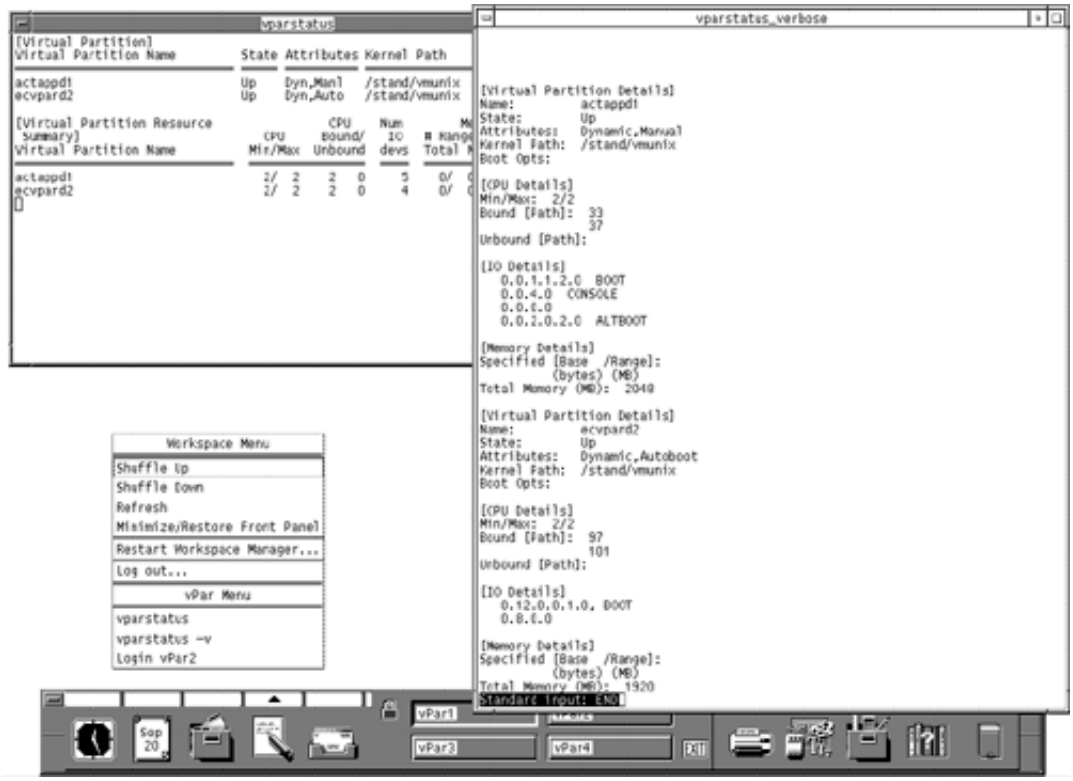
Figure 14-8. vPars Menu Picks



These menu picks give us vPar status, status with verbose output, and a remote login to our second vPar. Issuing these commands open up windows and executes the commands shown in our menu picks.

Figure 14-9 shows a screen shot with the two **vparstatus** windows open from having selected the two **vparstatus** commands from the menu:

Figure 14-9. Screen Shot Showing Two vparstatus Windows



The upper left-hand window contains the output of **vparstatus** and the right-hand window contains the output of **vparstatus -v**. The updated menu is shown in the lower left of the screen.

We can perform additional CDE customization, such as create workspaces for our vPars so that when we select a workspace we have windows open to the vPar for which the workspace is named.

We performed only basic customization to CDE for our vPars environment. You can perform extensive customization in CDE. The background provided in the remainder of this chapter provides information that can help you with additional customization.

GNOME

GNOME (GNU Network Object Model Environment) is a full desktop environment and application framework. GNOME is part of the GNU project which means it is open source software. You can find out more about GNOME at www.gnome.org and more about the GNU project at www.gnu.org. GNOME is not a window manager itself. Instead, it uses a variety of window managers from which you can select, *Sawfish* being the default window manager. GNOME provides development libraries and session management. Applications that use these features are tightly integrated with GNOME. Since GNOME is based on X Windows, X-based applications work great with GNOME.

The GNOME that you can download and run on HP-UX systems is *ximian*. You can download GNOME, system requirements, installation instructions, a FAQ, and a variety of other information about ximian GNOME from the URL http://www.hp.com/workstations/support/software/hpux/gnome/index.html. At the time of this writing this is a pilot project meaning that you wouldn't want to use ximian GNOME on HP-UX in production. I like GNOME so much and it worked so well that I left it as the default on my lab system.

From the URL above you can download GNOME, run **swinstall**, change **/etc/rc.config.d/desktop** to *DESKTOP=GNOME,* and run. The requirements state that for the pilot your system should be a B132 system. The installation instructions were explicit and downloading and configuring GNOME went smoothly. Please download the installation and FAQ before you work with GNOME. Having said that, I'll provide a syopsis of the steps I performed so you have an idea of what is requred to get GNOME running:

- Download the GNOME from the URL.
- Install the software: **swinstall -x mount_all_filesystems=false -s /var/tmp/B6852AA+B6848AB.depot**
- **/sbin/init.d/dtlogin.rc stop**
- Change **/etc/rc.config.d/desktop** from *DESKTOP=CDE* to *DESKTOP=GNOME*
- **/sbin/init.d/gdm.rc start**

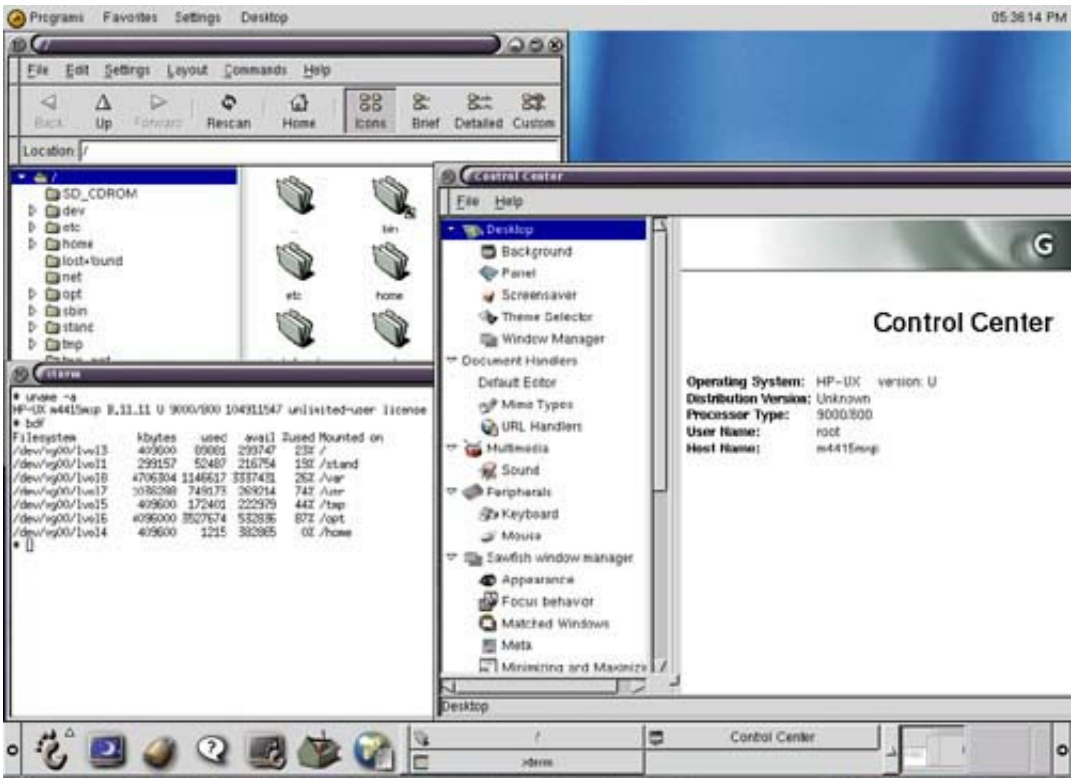
You should run GNOME on only the systems supported; however, I decided to run it on an unsupported system. It is a server and I wanted to use an X server program remotely, so I followed the procedure in the FAQ for using the GNOME login manager remotely. I performed the following:

- Run **gdmconfig** and select *Expert, XDMCP*, and *Enable XDMCP*. This adds *Enable=true* to the *[xdmcp]* section of **/etc/opt/gnome/gdm/gdm.conf**
- Comment out, using *#*, the line *0=/usr/bin/X11/X* from **/etc/opt/gnome/gdm.conf**.

The second step was not in the FAQ. This procedure worked great and I was able to run GNOME.

After logging in through the GNOME login screen I had a fully functional GNOME environment, as shown in Figure 14-10:

Figure 14-10. Full GNOME Screen Shot on HP-UX



There are several windows open in the figure. Along the bottom left are the GNOME start menu (the footprint), screen saver, screen locking tool, help system, terminal emulation, configuration tool, and Internet button. Along the top left you also have a *Programs* menu along with *Favorites, Settings,* and *Desktop*.

In the middle on the bottom are the programs currently running. In this case there is a file manager window at **/**, **Control Center**, and **xterm**. On the far right along the bottom are the *Workspace names*. In this case we have only one workspace setup, but we'll add a few more shortly. In the middle of the screen are the aforementioned applications.

The *Programs* at the top of the screen is a pull-down menu from which you can select the program you wish to run, as shown in Figure 14-11:

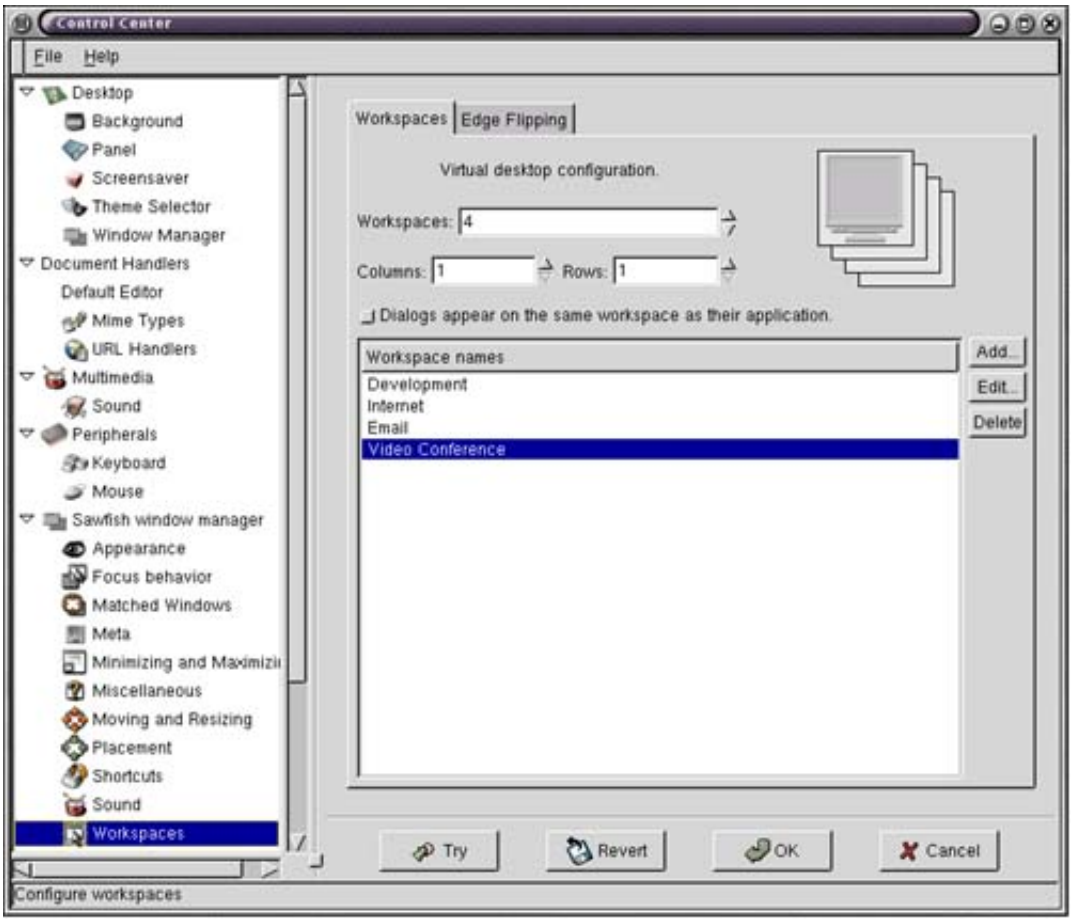
Figure 14-11. GNOME Menu Pick



The default organization of programs are shown in the figure. We have selected *System* which brings up various system programs we can run. You can also select the GNOME start menu at the bottom left of Figure 14-10 to select programs you wish to run.

You can have multiple workspaces in GNOME. Figure 14-12 shows using *Control Center* to specify the names of four workspaces:

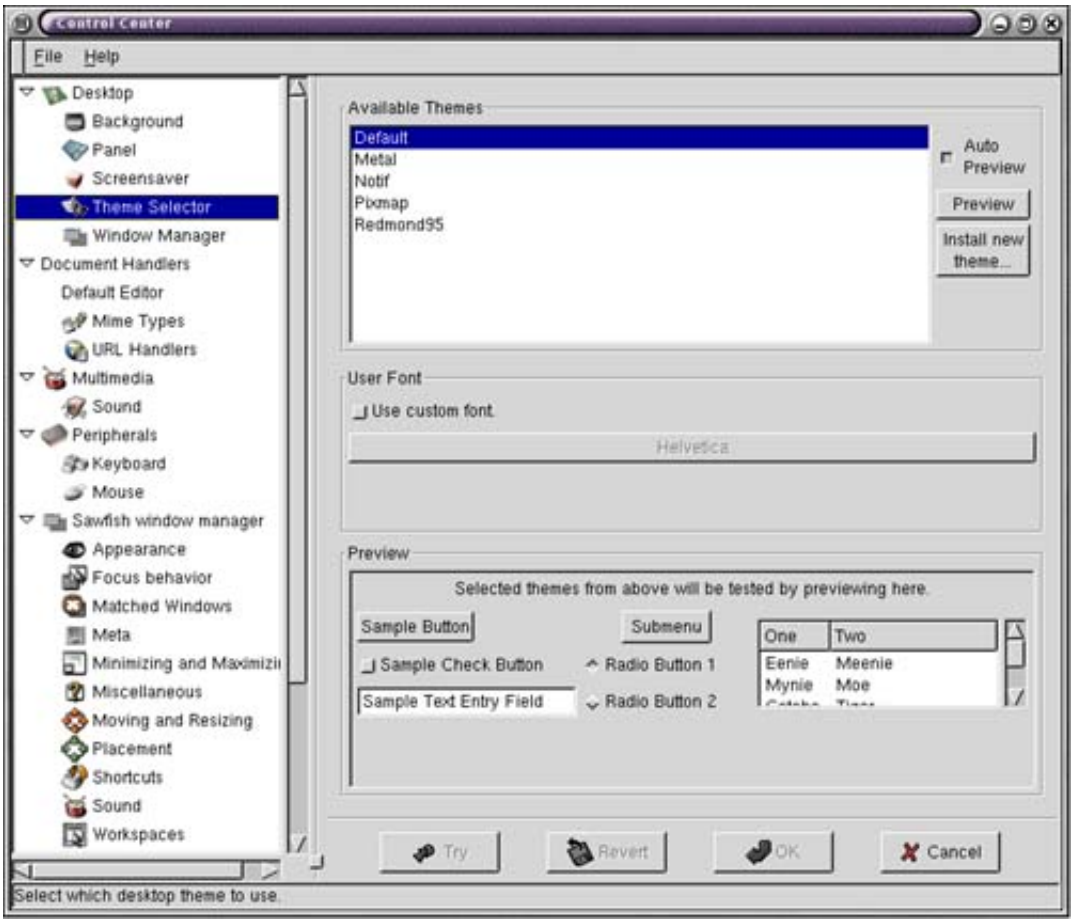
Figure 14-12. GNOME *Control Center* Showing *Workspaces*



Along the left side of the *Control Center* window are many menus that allow you to configure any aspect of your GNOME environment. We have selected *Sawfish window manager* and *workspaces* to name four workspaces that we want on our system. You can see in Figure 14-12 that there are many selections you can make for customizing your GNOME environment. We'll have four workspaces with the names shown when we press *OK*.

Next we'll select *Themes* in *Control Center*. Themes control your desktop in ways such as windows, borders, fonts, and so on. There are a variety of places from which you can download additional themes. Figure 14-3 shows some of the themes that are available:

Figure 14-13. GNOME *Control Center* Showing *Theme Selector*



There is unlimited customization you can apply to your GNOME environment. We've only scratched the surface of what you can do in GNOME in this section.

If you're interested in GNOME on your HP-UX system please monitor the web site listed earlier so you can download and work with the latest version of ximian GNOME on HP-UX. Be sure to read the requirements for your system and other related information when you visit the web site.



Chapter 15. CIFS/9000 and Samba

CIFS/9000 Overview

Samba Overview

Additional Samba Topics



CIFS/9000 Overview

CIFS/9000 Server is an HP product based on Samba that provides file and print services to CIFS clients. The server is an HP 9000 and the clients are usually Windows-based systems; however, the client could also be an HP 9000 or any system running CIFS client software. CIFS/9000 Client is used to access file and print services on a CIFS server. To an HP-UX user running CIFS/9000 Client software, the shares on CIFS/9000 Server look like UNIX filesystems whether they are on a UNIX server or a Windows server. The server could be an HP 9000 running CIFS/9000 Server, a Windows-based system, or any system running CIFS as a server. In this chapter, I'll cover only CIFS/9000 Server being accessed by a Windows-based system since that is the use for which most HP 9000 users will deploy CIFS/9000.

CIFS stands for Common Internet File System, which is a Windows specification for remote file access. CIFS is a remote access protocol based largely on Server Message Block (SMB), which has long been used as the native file-sharing protocol on Windows-based systems. CIFS is a remote file access protocol that sits on top of the host file system(s). As mentioned earlier there is both a CIFS Server, for which I'll use an HP 9000 system in this chapter, as well a CIFS Client, for which I'll use a Windows-based system in this chapter. We'll set up CIFS/9000 server such that an HP-UX file system can be mounted on a Windows-based system.

CIFS/9000 Server is based on Samba open source software. For that reason, the second half of this chapter is from one of my other books covering Samba running on a Linux system. The HP 9000 setup of CIFS/9000 Server covered in the first half of this chapter is very similar to the Samba setup on a Linux system covered in the second half of this chapter.

My goal in the first half of this chapter is to give an overview of the setup of CIFS/9000 and demonstrate a subset of its functionality. We'll focus on the file server functionality only. We'll set up shares on the CIFS/9000 server system that will appear as a drive letters and icons on a Windows system.

CIFS/9000 Server provides this file-sharing functionality using Server Message Block (SMB) protocol as described earlier. SMB runs on top of TCP/IP. In our example in this chapter, both the Windows system and CIFS/9000 Server system are running TCP/IP and SMB. These provide all of the technology that is required to establish file sharing between the two systems.

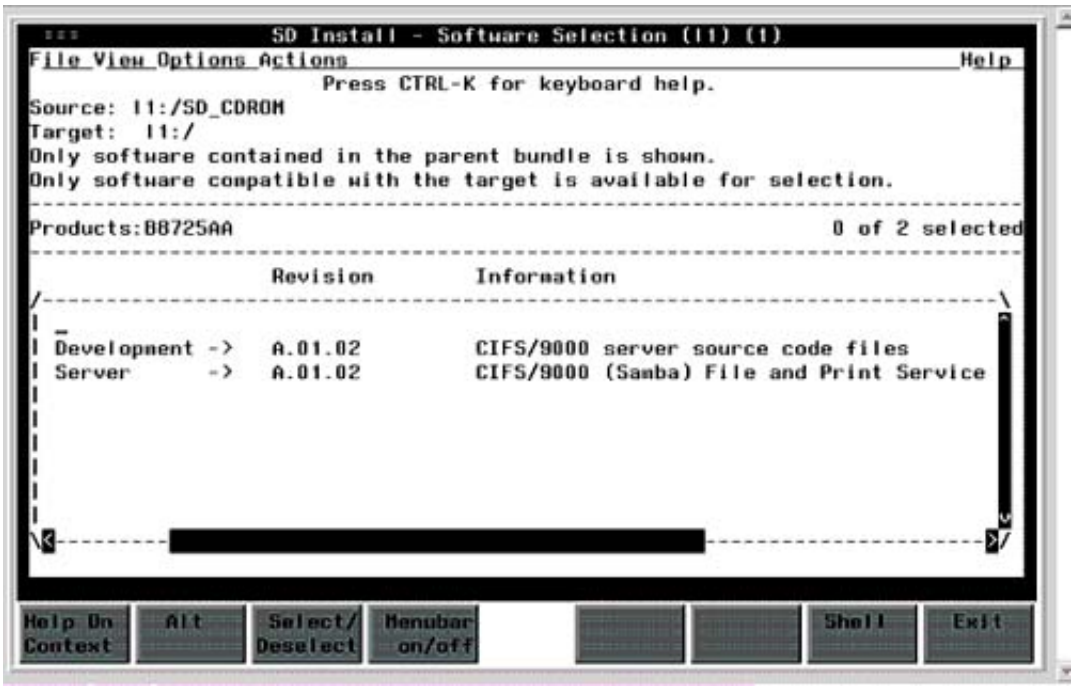
Chapter 13 covers the Network File System (NFS) running on a Windows system and accessing files on a UNIX system. This functionality is similar to that which we'll cover with CIFS/9000 in this chapter. In addition to the file-sharing capability, CIFS/9000 also provides printer sharing and additional user access control capability. We won't focus on these capabilities in this chapter; however, CIFS/9000 does indeed provide some advanced functionality in these areas.

At the time of this writing, CIFS/9000 contains the functionality just mentioned, including: file-sharing; printer sharing; and advanced user access control of files. There are many advancements taking place with Samba that will be incorporated into CIFS/9000 and other software provided under the GNU Public License (GPL) as free software. Because the software is free, many people have access to it and spend time enhancing the software. For this reason, you may find that additional functionality is included in CIFS/9000 in the future.

Installing CIFS/9000

CIFS/9000 is loaded from CD-ROM like any other HP application. Figure 15-1 shows the **swinstall** screen with the two CIFS/9000 components that are part of the parent bundle. As shown in the figure, you receive both the source code and the compiled version of CIFS/9000 Server as one bundled product. The same would be true of the CIFS/9000 Client software. The source code is included because CIFS/9000 Server is based on Samba Open Source software for which source code is freely available.

Figure 15-1. Installing CIFS/9000 Server with swinstall



After having installed both components of the parent bundle shown in Figure 15-1, we have all of the software present on our system to configure CIFS/9000 Server. Although I don't plan on modifying the source code, I may view it at some point, so I loaded it as well.

Let's next change directory to **/opt** (for optional software) and see what directories have been created in the following listing:

```
# cd /opt
# ls -l
total 10
dr-xr-xr-x   6 bin        bin            96 Jul 11 19:27 audio
dr-xr-xr-x  10 bin        bin          1024 Jul 11 19:27 dce
dr-xr-xr-x   4 bin        bin          1024 Jul 11 19:44 dcelocal
dr-xr-xr-x   3 bin        bin            96 Jul 11 19:25 fc
dr-xr-xr-x   3 bin        bin            96 Jul 11 19:18 fcms
dr-xr-xr-x   4 bin        bin            96 Jul 11 19:44 graphics
drwxr-xr-x   5 root       sys            96 Jul 11 19:29 hparray
dr-xr-xr-x   4 bin        bin            96 Jul 11 19:27 ifor
dr-xr-xr-x  16 bin        bin          1024 Jul 12 13:10 ignite
dr-xr-xr-x   3 bin        bin            96 Jul 11 19:26 image
drwxr-xr-x   2 root       root           96 Jul 11 19:14 lost+found
dr-xr-xr-x   7 bin        bin            96 Jul 11 19:25 nettladm
drwxr-xr-x   2 root       sys          1024 Jul 11 22:57 networkdocs
dr-xr-xr-x   7 bin        bin            96 Jul 11 19:27 pd
drwxr-xr-x   8 root       users        1024 Jul 18 11:53 samba
drwxr-xr-x   3 root       users          96 Jul 18 11:53 samba_src
dr-xr-xr-x   5 bin        bin            96 Jul 11 19:25 upgrade
dr-xr-xr-x   4 bin        bin            96 Jul 11 19:27 video
drwxr-xr-x   3 root       sys            96 Jul 12 13:10 webadmin
#
```

This listing shows that there are both the **samba** and **samba_src** directories that we loaded in **/opt**. The CIFS/9000 product with which we'll be working is loaded in the **samba** directory, so let's change to it and then see what files and directories are present under **/opt/samba** and **/opt/samba/bin**:

```
# ls -l /opt/samba
total 44
-rwxr-xr-x   1 root       users      17982 Jul  7  1999 COPYING
drwxr-xr-x   4 root       users         96 Jul 18 11:53 HA
-rwxr-xr-x   1 root       users        772 Jan 10  2000 README
drwxr-xr-x   2 root       users       1024 Jul 18 11:53 bin
drwxr-xr-x   7 root       users       1024 Jul 18 11:53 docs
drwxr-xr-x   6 root       users         96 Jul 18 11:53 man
drwxr-xr-x   2 root       users       1024 Jul 18 11:53 script
```

```
drwxr-xr-x   6 root       users        96 Jul 18 11:53 swat
# ls -l /opt/samba/bin
total 13588
-rwxr-xr-x   1 root       users      1708 Apr 20 17:38 addtosmbpass
-rwxr-xr-x   1 root       users       446 Apr 20 17:38 convert_smbpasswd
-rwxr-xr-x   1 root       users    292124 Apr 20 17:38 make_printerdef
-rwxr-xr-x   1 root       users    292115 Apr 20 17:38 make_smbcodepage
-rwxr-xr-x   1 root       users    579513 Apr 20 17:38 nmbd
-rwxr-xr-x   1 root       users    402933 Apr 20 17:38 nmblookup
-rwxr-xr-x   1 root       users    752117 Apr 20 17:38 rpcclient
-rwxr-xr-x   1 root       users     12988 Apr 20 17:38 samba_setup
-rwxr-xr-x   1 root       users    476863 Apr 20 17:38 smbclient
-rwxr-xr-x   1 root       users   1302921 Apr 20 17:38 smbd
-rwxr-xr-x   1 root       users    707074 Apr 20 17:38 smbpasswd
-rwxr-xr-x   1 root       users    402929 Apr 20 17:38 smbspool
-rwxr-xr-x   1 root       users    324986 Apr 20 17:38 smbstatus
-rwxr-xr-x   1 root       users      4862 Apr 20 17:38 smbtar
-rwxr-xr-x   1 root       users      1860 Apr 20 17:38 startsmb
-rwxr-xr-x   1 root       users      2340 Apr 20 17:38 stopsmb
-rwxr-xr-x   1 root       users    789251 Apr 20 17:38 swat
-rwxr-xr-x   1 root       users    288012 Apr 20 17:38 testparm
-rwxr-xr-x   1 root       users    312629 Apr 20 17:38 testprns
# export PATH=$PATH:/opt/samba/bin
#
```

There are some files in **/opt/samba/bin** of interest. Let's take a look at one of these used to configure CIFS/9000 in the upcoming section.

Configuring CIFS/9000

Among the executables that appear in the **/opt/samba/bin** directory is **samba_setup**. When you run this program, it performs some of the initial CIFS/9000 setup, including the name of the domain or workgroup and selecting the authentication to be used. Authentication gets a little tricky because you have a few options. The Samba part of this chapter covers some of the authentication issues. Let's now run **samba_setup**:

```
# samba_setup


This is a utility to help you configure your HP-UX system as
a SAMBA server.  This should only be run once after the initial
installation of SAMBA.
Do you wish to continue with the configuration?
(Y or N)?  : y

Proceeding with samba_setup...


You now must choose what level of authentication you
would like this SAMBA server to use:
1) domain
2) server
3) user
4) share
5) CANCEL
#? 3

You have chosen user level authentication
Is this correct?
(Y or N)?  : y

Enter the name you wish the SAMBA server to be known as (the
netbios name), or just hit enter if you want to keep the default
server name (l1):



Enter the name of the domain or workgroup that
you want this SAMBA server to be a part of: users


Configuring SAMBA server with user level authentication...
```

```
Your SAMBA server should now be ready to start.


You may wish to put /opt/samba/bin in the PATH for root
so that you don't have to type the fully qualified path
name for the SAMBA utilities and commands.
#
```

The minimal configuration we completed, along with information related to the Samba configuration, is in **/etc/opt/samba/smb.conf**, which is shown in the following listing:

```
# cat /etc/opt/samba/smb.conf
# This is the main Samba configuration file. You should read the
# smb.conf(5) manual page in order to understand the options listed
# here. Samba has a huge number of configurable options (perhaps too
# many!) most of which are not shown in this example
#
# Any line which starts with a ; (semi-colon) or a # (hash)
# is a comment and is ignored. In this example we will use a #
# for commentry and a ; for parts of the config file that you
# may wish to enable
#
# NOTE: Whenever you modify this file you should run the command "testparm"
# to check that you have not many any basic syntactic errors.
#
#======================
Global Settings
======================
[global]
   netbios name = l1

# workgroup = NT-Domain-Name or Workgroup-Name, eg: REDHAT4
   workgroup = users

# server string is the equivalent of the NT Description field
   server string = Samba Server

# this tells Samba to use a separate log file for each machine
# that connects
   log file = /var/opt/samba/log.%m

# Put a capping on the size of the log files (in Kb).
   max log size = 1000

# Security mode. Most people will want user level security. See
# security_level.txt for details.
   security = user
# Use password server option only with security = server or domain
   password server = *

# You may wish to use password encryption. Please read
# ENCRYPTION.txt, Win95.txt and WinNT.txt in the Samba documentation.
# Do not enable this option unless you have read those documents
  encrypt passwords = no

# Most people will find that this option gives better performance.
# See speed.txt and the manual pages for details
   socket options = TCP_NODELAY

# Browser Control Options:
# set local master to no if you don't want Samba to become a master
# browser on your network. Otherwise the normal election rules apply
   local master = no

   read only = no
   preserve case = yes
   short preserve case = no
   dos filetime resolution = yes
   syslog = 0

#============================          Share           Definitions
==============================
[homes]
```

```
   comment = Home Directories
   browseable = no

# This one is useful for people to share files
[tmp]
   path = /tmp

#
```

You can modify this file directory to make changes for your environment.

In order to run CIFS/9000, we need to start the Samba server. This is done manually with the **startsmb** program as shown in the following listing:

```
# /opt/samba/bin/startsmb
Samba started successfully; process ids: smbd: 25363, nmbd: 25361
# ps -ef | grep samba
    root 25363     1  0 12:10:45 ?         0:00 /opt/samba/bin/smbd -D
    root 25361     1  0 12:10:45 ?         0:00 /opt/samba/bin/nmbd -D
#
```

We have successfully started the Samba server with **startsmb**, as confirmed by the Samba daemons shown in the **ps** output. We want to start Samba every time our system boots, so we can modify the **/etc/rc.config.d/samba** file and change the value of *RUN_SAMBA* to *1,* as shown in the following listing:

```
# pwd
/etc/rc.config.d
# cat samba
#
# (c) Copyright Hewlett-Packard Company 1999
#
# This program is free software; you can redistribute it and/or modify
# it under the terms of the GNU General Public License as published by
# the Free Software Foundation; either version 2 of the License, or (at
# your option) any later version.
#
# This program is distributed in the hope that it will be useful, but
# WITHOUT ANY WARRANTY; without even the implied warranty of
# MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See GNU General
# Public License for more details.
#
# You should have received a copy of the GNU General Public License
# along with this program; if not, write to the Free Software
# Foundation, Inc., 675 Mass Ave, Cambridge, MA 02139, USA.
#
# samba configuration: set RUN_SAMBA to a non-zero value to
# initiate the SAMBA server at run level 2.
#
# Installed at /etc/rc.config.d/samba
#

RUN_SAMBA=1
#
```

Up to this point, we have loaded CIFS/9000 using **swinstall**, performed some minimal configuration, and started Samba. The entire process has taken only minutes and we're ready to use CIFS/9000.

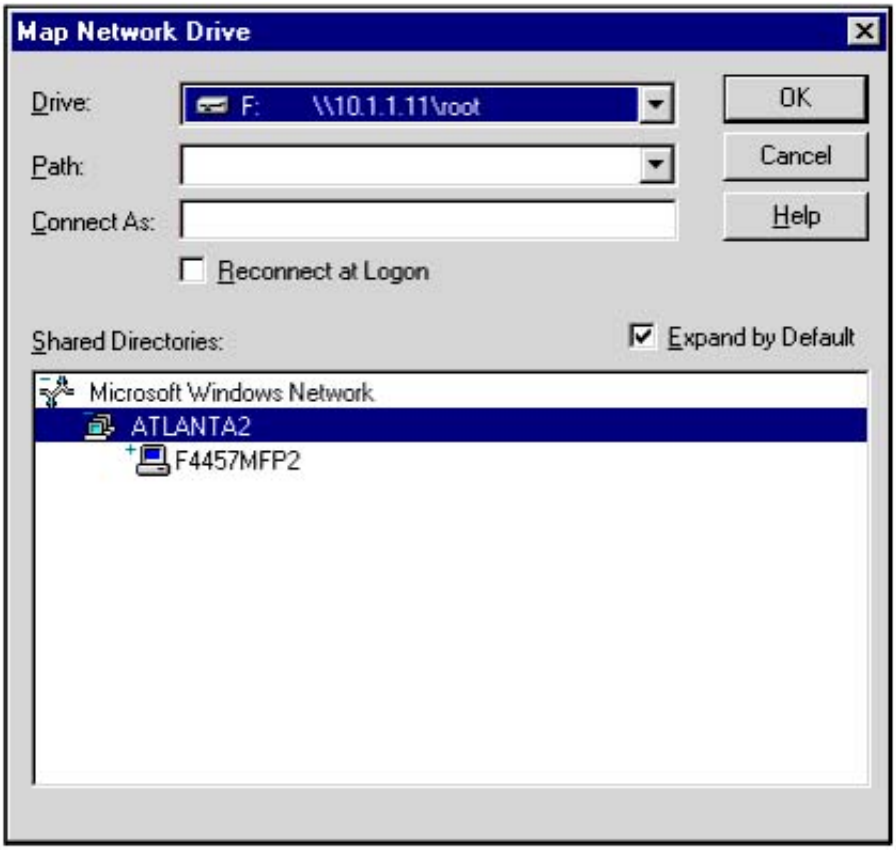
Now that we have loaded Samba and performed the initial configuration, we'll move to our Windows system and access an HP-UX file system on the Windows system as if it were local.

Map a Network Drive

Although we haven't done much work, we're now ready to map a network drive on our Windows system to an HP-UX file system.

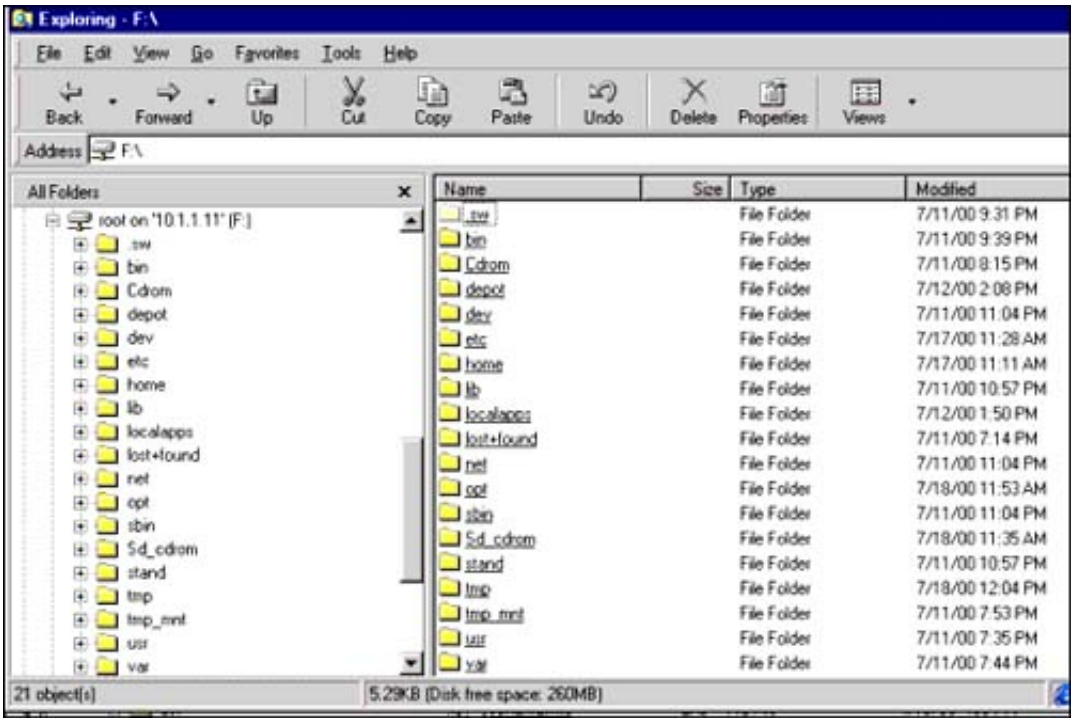
Let's map the root file system on our CIFS/9000 server to **F:** on our Windows system, as shown in Figure 15-2:

Figure 15-2. Map Root FileSystem on CIFS/9000 Server to F:



We now have access to the root filesystem on our CIFS/9000 server (10.1.1.11), as though it were local to our Windows system. Figure 15-3 shows accessing **F:**, which is root on our CIFS/9000 Server, to view the files and directories:

Figure 15-3. Viewing HP-UX Root Filesystem as F: on Windows



We can also use the **net view** command on our Windows system to see both the root filesystem on our CIFS/9000 server, which we have mapped to **F:** as well as **tmp**, which was in **smb.conf** but was not mapped to a drive letter, as shown in the following listing:

```
C:\> net view \\10.1.1.11

Shared resources at \\10.1.1.11

Samba Server

Share name     Type     Used as    Comment
-------------------------------------------
root           Disk     F:         Home Directories
tmp            Disk

The command completed successfully.

C:\>
```

We have only scratched the surface of CIFS/9000 functionality in this section. The subjects of user authentication, printing, ACLs, and many other topics were not covered. The next section covering Samba on Linux includes some additional sections of interest, such as the Web-based configuration tool for Samba called SWAT, that you may want to review. There are some documents on docs.hp.com that contain more detailed information on CIFS/9000 that you'll want to download and print if you plan to use CIFS/9000 in your environment.

Samba Overview

Samba is an application that allows a Linux or UNIX host to act as a file server for Windows systems and Windows systems can access UNIX filesystems and printers using their native Windows networking. We'll demonstrate a Linux system mounting a Windows drive and a Windows system mapping a Linux drive in this section.

My goal in this section is to give an overview of the setup of Samba and demonstrate a subset of its functionality. In this section we'll use Red Hat Linux software used throughout this book; however, Samba is available for almost all Linux and UNIX variants.

Samba provides its file-sharing functionality using Server Message Block (SMB) protocol. SMB runs on top of TCP/IP. In our example in this section, both the Windows system and UNIX system are running TCP/IP and SMB. These provide all of the technology that is required to establish file sharing between the two systems.

At the time of this writing Samba contains the functionality just mentioned: file-sharing, printer sharing, and advanced user access control of files. There are many advancements taking place with Samba and other software provided under GNU Public License (GPL) as free software. Because the software is free, many people have access to it and spend time enhancing the software. For this reason, you may find that additional functionality is included in Samba and other such software. There are many enhancements planned for Samba, including an administration tool and other advancements that were not available at the time this chapter was written. There is more information about obtaining Samba and other free software at the end of this section.

smb.conf Setup

Because Samba is supplied on the Red Hat Linux CD-ROM, we'll walk through a simple Samba setup using Red Hat Linux. When installing Red Hat Linux, you can select the software packages you wish to load, as you can on most all UNIX variants. If you did not load Samba at the time you originally loaded the operating system, you can use a graphical *RPM* tool or **rpm** from the command line to load Samba or any other software. These tools were briefly discussed in the System Administration Chapter 10.

The configuration file for Samba is **/etc/samba/smb.conf**. The **smb.conf** file has all Samba-related setup information in it. For our simple Samba setup in this chapter we need to have only a few lines in the file setup. We need to have a few *Global Settings* and *Share Definitions* in order to achieve our file system sharing between the Windows and Linux systems. The following are three select sections from **smb.conf** that were modified for our example in this section:

```
# workgroup = NT-Domain-Name or Workgroup-Name
   workgroup = DEVENV


  encrypt passwords = yes
   smb passwd file = /etc/samba/smbpasswd

[homes]
   comment = Home Directories
   browseable = yes
   writable = yes
   valid users = %S
   create mode = 0664
   directory mode = 0775
```

1. We have a *workgroup* of *DEVENV* specified.
2. We want our passwords to be *encrypted* and we'll use the file **/etc/samba/smbpasswd** for these encrypted passwords. We'll have a user on both the Windows and Linux systems with the same name.
3. We have some information related to our *Share Definitions*, such as making the *Home Directories browseable*.

We made only these three modifications to the **smb.conf** file; however, next we'll run a Samba utility called

testparm. This utility will check our **/etc/smb.conf** file for errors. This utility produces a very long output which I won't include here, but you'll want to run this and check for any warnings or errors it produces.

```
# testparm smb.conf

Load smb config files from smb.conf
Processing section "[homes]"
Processing section "[printers]"
Loaded services file OK.
Press enter to see a dump of your service definitions
# Global parameters
[global]
              coding system =
              client code page = 850
              code page directory = /usr/share/samba/codepages
              workgroup = DEVENV
              netbios name =
              netbios aliases =
              netbios scope =
              server string = Samba Server
              interfaces =
              bind interfaces only = No
              security = USER
              encrypt passwords = Yes
              update encrypted = No
              allow trusted domains = Yes
              hosts equiv =
              min passwd length = 5
              map to guest = Never
              null passwords = No
              obey pam restrictions = Yes
              password server =
              smb passwd file = /etc/samba/smbpasswd
                 .
                 .
                 .
```

There were not warnings or errors produced from having run **testparm** so we'll proceed to the next step.

User Setup

We now need to create a user on both the Linux and Windows systems that can be used for our Samba-related work. It may already be that you have suitable users on your system. For the purposes of this demonstration we'll create a user on both systems.

On the Linux system we can use the **useradd** program from the command line or *User Manager* graphical program (*Programs-System-User Manager*) to add the user. The user will appear in the **/etc/passwd** file after they have been added. We'll then issue the following command to add the encrypted user passwd to the **/etc/samba/smbpasswd** file:

```
# smbpasswd -a linuxconnect
```

linuxconnect is the name of the user we created for this example. You can view the **smbpasswd** file to see the entry for your user.

This same user was created on the Windows system. The menu pick *Start-Programs-Administrative Tools-Computer Management (Local Users and Groups-Users)* was used to create *linuxconnect*.

Samba Startup

You can start Samba daemons every time the system boots or start them at the command line. The following commands show starting the daemons at the command line and then viewing them:

```
# /sbin/service smb start

Starting SMB services:          [  OK  ]
Starting NMB services:          [  OK  ]
```

```
# ps -ef  | grep mbd
root     10828     1  0 16:59 ?        00:00:00 smbd -D
root     10833     1  0 16:59 ?        00:00:00 nmbd -D
#
```

We started the daemons with **/sbin/service smb start** and check to see our two daemons running with **ps**. Should you make a change to your Samba setup and wish to restart the daemons, you could use **/sbin/service smb restart**.

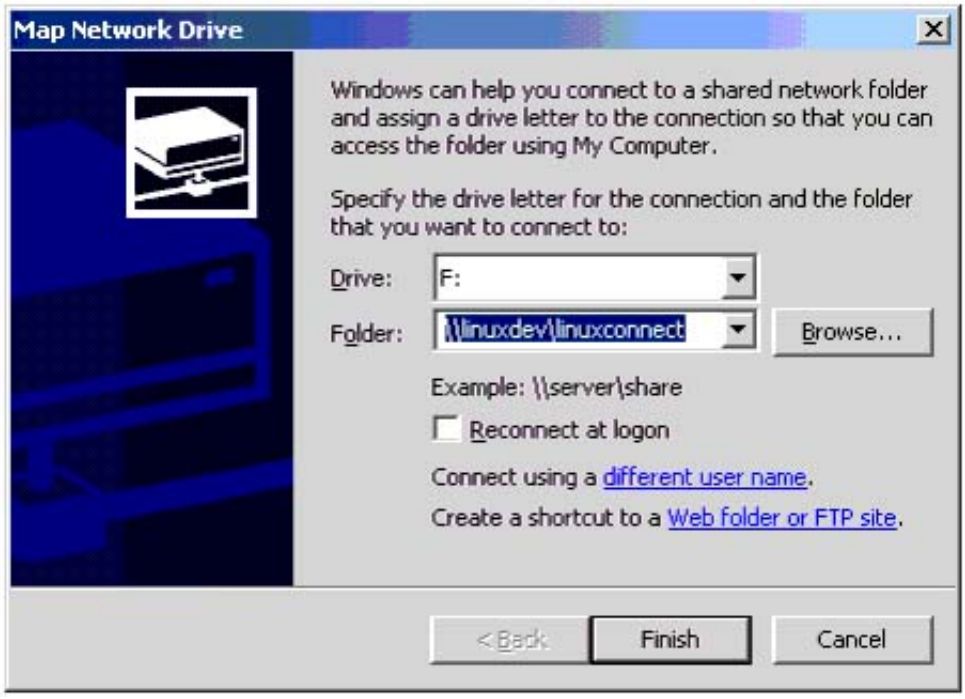
The *smbd* server daemon provides the file and print services to SMB clients, such as Windows systems. SMB stands for "Server Message Block" and is defined as a network protocol for sharing files, printers, serial ports, and communications abstractions such as named pipes and mail slots between computers.

The *nmbd* server daemon allows for NetBIOS over IP name service requests over a network, like those produced by SMB/CIFS clients such as Windows systems. We have now performed all of the basic setup required to proceed with mounting disks.

Mapping a Windows Drive to a Linux Directory

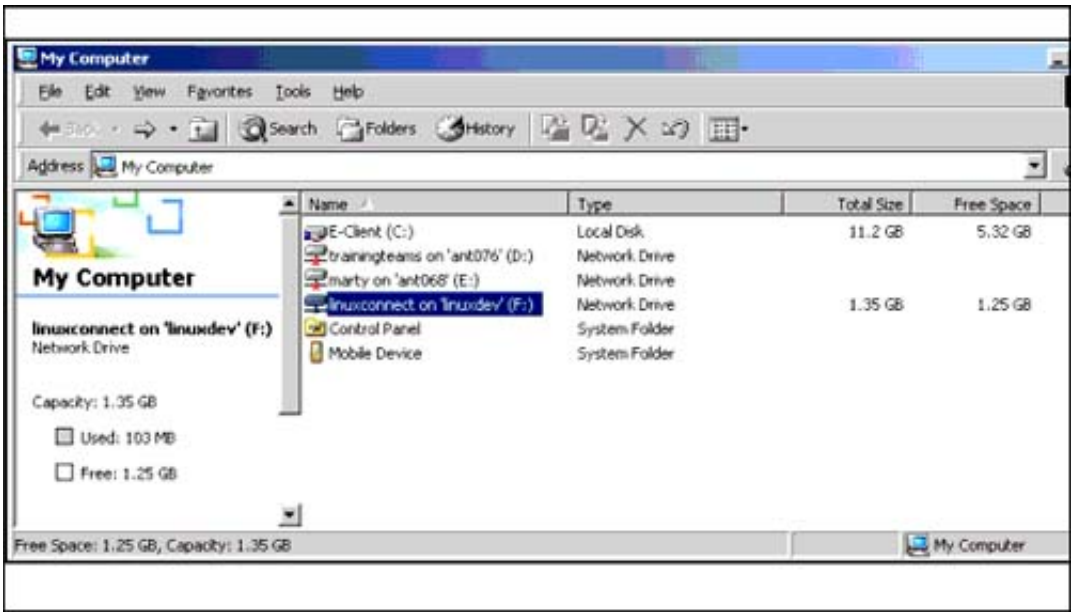
Now we can both browse the Linux system from Windows as well as mount a specific drive directory of the Linux system on our Windows system. Figure 15-4 shows the window in which we specify the Linux system and directory we will mount as *F:* on the Windows system:

Figure 15-4. Mapping Linux Drive to F: on Windows



We specified the path of *linuxdev**linuxconnect* to mount on the Windows system drive **F:**. This is the hostname (*linuxdev*) and the username (*linuxconnect*). The system knows automatically to go to the home directory for *linuxconnect* and mounts that directory as **F:** for us on the Windows system, as shown in the Figure 15-5:

Figure 15-5. Viewing HP-UX Root Filesystem as F: on Windows



The **/home/linuxconnect** directory on our Linux system is now fully accessible on the Windows system. We can browse, edit, copy, and perform other tasks on these shared files.

We can use **net view** on our Windows system to see the shared resource of our Linux directory as shown in the following listing:

```
c: net view \\linuxdev

Shared resources at \\linuxdev

Samba Server

Share name   Type         Used as  Comment

-------------------------------------------------------
homes        Disk         Home     Directories
linuxconnect Disk         F:       Home Directories
The command completed successfully.
```

This listing shows that **F:** has our Linux directory mapped to it.

Let's now get the overall status of the Samba setup with the **smbclient** utility on our Linux system, as shown in the following listing:

```
# su - linuxconnect
$ smbclient -L linuxdev

added interface ip=192.168.1.102 bcast=192.168.1.255 nmask=255.255.255.0

Domain=[DEVENV] OS=[Unix] Server=[Samba 2.2.3a]

                  Sharename      Type      Comment
                  ---------      ----      -------
                  homes          Disk      Home Directories
                  IPC$           IPC       IPC Service (Samba Server)
                  ADMIN$         Disk      IPC Service (Samba Server)
                  linuxconnect   Disk      Home Directories

                  Server               Comment
                  ---------            -------
                  LINUXDEV             Samba Server

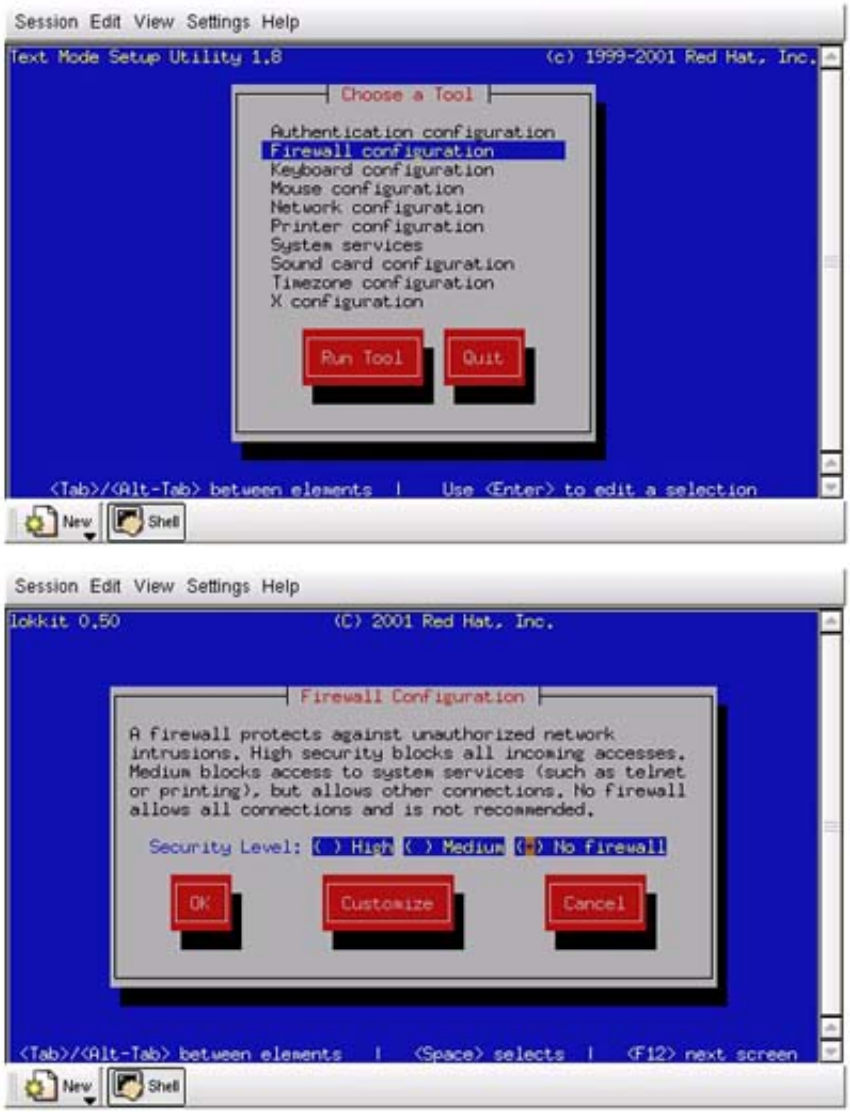
                  Workgroup            Master
                  ---------            -------
                  ATLANTA2             F4457MXP
                  DEVENV               LINUXDEV
```

Before issuing the **smbclient** command we changed user to *linuxconnect* and then issued the command.

This utility produces a useful summary of the Samba setup, including the share *linuxconnect* we set up, the Samba server for our example, and other useful information.

Keep in mind that Linux, including the Red Hat distribution used in this example, has a high level of security by default upon installation. The firewall setting on the system in our example was set too high for this drive to be mounted in our example. We first had to lower the security level of the firewall setting. Using the **/usr/sbin/setup** program and selecting *Firewall Configuration,* we lowered the *Security Level* to *No Firewall,* as shown in Figure 15-6:

Figure 15-6. *setup* Program Used to Eliminate Firewall



We used *setup* to eliminate the firewall in this example. This greatly reduces the overall security of the Linux system, so you have to consider this before making the change. As a result of this selection, files on the system were updated to permit more open access to the system, which gave us the ability to map a network drive to our Linux system, but it also reduced the overall level of security on the system.

Mounting a Windows Drive on a Linux System

The previous example demonstrated mapping a Linux file system to a Windows drive. This is commonly

done since Linux file systems often act as file servers in a Windows environment. We can also get access to a Windows drive on our Linux system for file sharing purposes or to back up a Windows drive to a Linux system. The following commands show mounting the **C:** drive on our Windows system as the directory **/home/linuxconnect/pcdev** on our Linux system:

```
# smbmount //F4457MXP/c /home/linuxconnect/pcdev -o username=linuxconnect
Password:
#
#
# cd /home/linuxconnect/pcdev
# ls
6.1_for_2002_Setup.exe  HP                  phone
~adpttmp                I386                PMig01.Log
aim95.exe               INFONET             PMig02.Log
AUTOEXEC.BAT            IO.SYS              PMig03.Log
BDE                     kmd.exe             PMig0.Log
BOOT.BAK                lj527en.exe         PMig.Log
boot.ini                Morpheus-b3d-v2.exe  Program Files
BOOTLOG.TXT             MSDOS.SYS           Recycled
cmdcons                 My Music            RECYCLER
cmldr                   My Photos           RunDelay
COMMAND.COM             NETPRINTER          sdc_install.log
comreads.dbg            netzero.exe         sdc.ok
comused.dbg             NI                  setupmpe.exe
CONFIG.SYS              notworm             Support
data                    NTDETECT.COM        System Volume Information
DMI                     ntldr               temp
Documents and Settings  OMNIBOOK            temp.ps
drivers                 OMNILIB             TOUR
ENV                     openmail.log        unzipped
FDWINW                  pagefile.sys        VERSION.INF
hiberfil.sys            Perl                WINNT
#
```

In the **smbmount** command we specified the name of the Windows system and the path (**/F4457MXP/c**) we wanted to mount on the Linux system. We could have specified any drive or a directory of any drive but we specified the full **C:** directory. We then specified the mount point on the Linux system of **/home/linuxconnect/pcdev** and the username of *linuxconnect.* We were then prompted for the password for this user and given access to the drive. The listing shows that the entire **C:** drive has been mounted.

Now that the entire **C:** drive is accessible on our Linux system, we can perform any tasks on these files, such as performing a backup of the entire drive or selected files to our Linux system.

Figure 15-7 shows a Linux file manager window (*Konqueror*) open showing **C:** from the Windows system.

Figure 15-7. *Konqueror* Viewing C: on Linux System

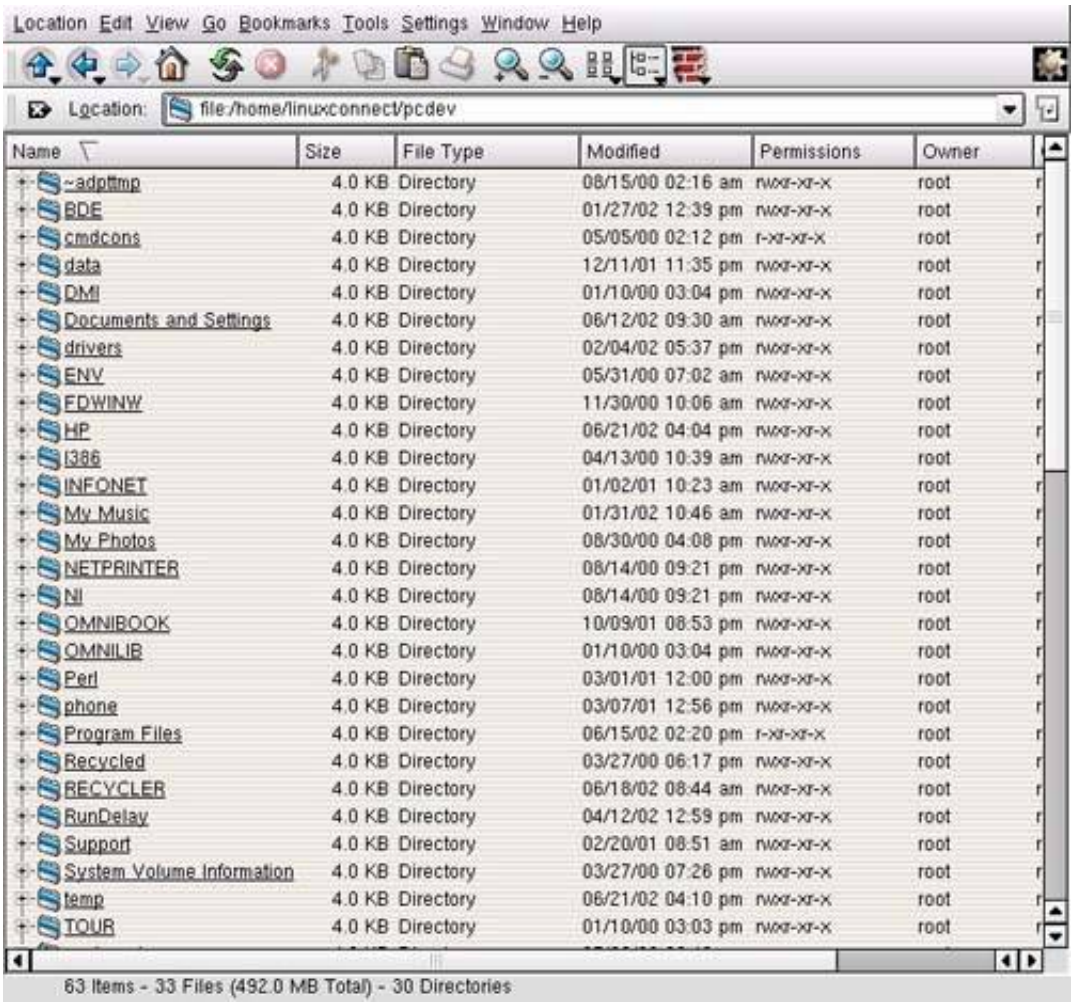


Figure 15-7 shows folders on the **C:** drive of our Windows system displayed in one of the file manager windows on our Linux system. We can traverse this hierarchy freely to view, copy, and work with these files in other ways.

We have demonstrated mounts working in both directions. **C:** on our Windows system was mounted as **/home/linuxconnect/pcdev** on our Linux system. **/home/linuxconnect** was mapped to **F:** on our Windows system. Additional Samba functionality exists, such as print serving, that were not shown in this example.

Log Files

Like most UNIX applications, Samba provides extensive logging. The **smb.conf** file contains a section that allows you to specify the level of Samba logging you wish to take place. The short section below shows that you can have separate log files for each Windows machine that connects, and you can specify the maximum size of the log file:

```
# this tells Samba to use a separate log file for each machine
# that connects
    log file = /var/log/samba/log.%m

# Put a capping on the size of the log files (in Kb).
    max log size = 50
```

The directory **/var/log/samba** contains a variety of Samba log files, including the log file for the Windows system used in our examples in this chapter, called *f4457mfp2,* as shown in the following listing:

```
# ls -l /var/log/samba
```

```
total 27
-rw-r--r--    1 root     root            0 Jun 15 11:15 f4412bfg.log
-rw-r--r--    1 root     root            0 Jun 23 04:03 f4457mxp.log
-rw-r--r--    1 root     root          664 Jun 23 04:03 f4457mxp.log.1
-rw-r--r--    1 root     root            0 Jun 21 12:22 linuxdev.log
-rw-r--r--    1 root     root         8993 Jun 29 05:01 log.nmbd
-rw-r--r--    1 root     root         1918 Jun 29 04:56 log.smbd
-rw-r--r--    1 root     root          405 Jun 29 04:56 nmbd.log
-rw-r--r--    1 root     root           65 Jun 24 04:03 smbd.log
-rw-r--r--    1 root     root           65 Jun 24 04:03 smbd.log.1
-rw-r--r--    1 root     root         3271 Jun 29 05:00 smbmount.log
-rw-r--r--    1 root     root         7431 Jun 23 04:03 smbmount.log.1
```

[Team LiB]

PREVIOUS NEXT

Additional Samba Topics

Samba Web Configuration Tool (SWAT)

SWAT is a Web-based administration tool for Samba. It is easy to configure and provides a simple interface for most Samba configuration tasks. On our Red Hat Linux system, the following steps had to be performed to get SWAT running. If you have a different UNIX variant, then your steps will be different.

Edit the file **/etc/xinetd.d/swat** and change *disable* to *no* as shown in the following listing:

```
# default: off
# description: SWAT is the Samba Web Admin Tool. Use swat \
#                       to configure your Samba server. To use SWAT, \
#                       connect to port 901 with your favorite web browser.
service swat
{
                  port= 901
                  socket_type= stream
                  wait = no
                  only_from = 127.0.0.1
                  user= root
                  server= /usr/sbin/swat
                  log_on_failure+= USERID
                  disable= no
}
```

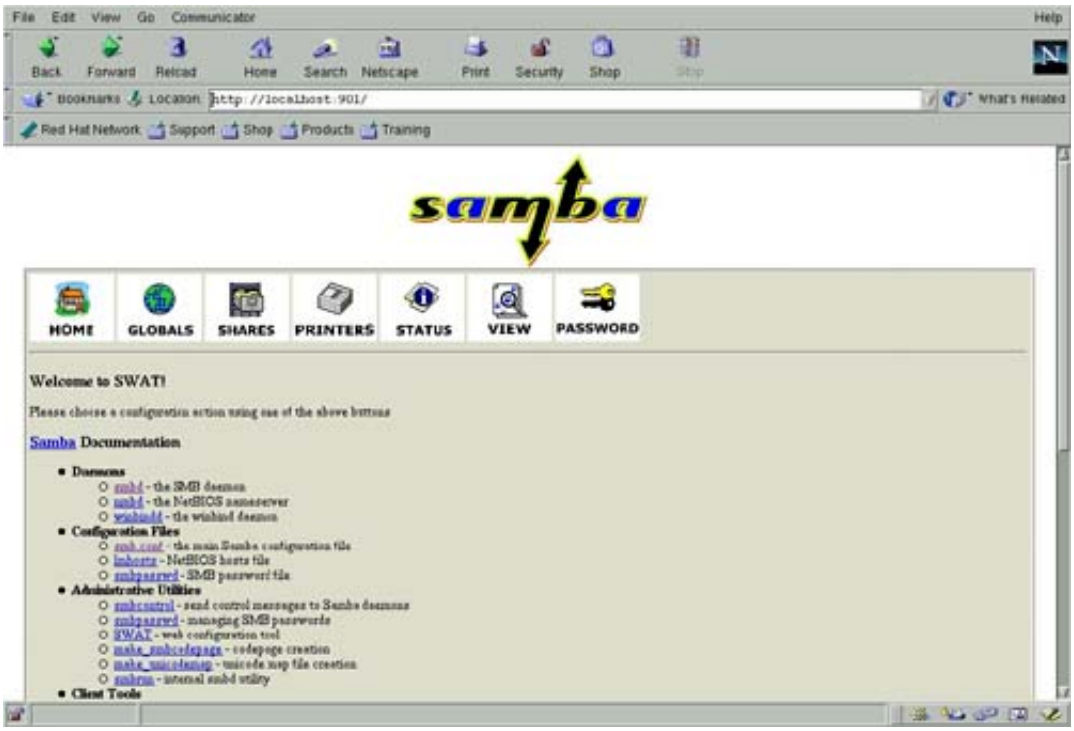
Restart **xinetd** with the following command:

```
# /sbin/service xinetd restart
```

From your browser specify a *location* of *localhost:901* and enter the root user and password when prompted for them.

Figure 15-8 shows the SWAT interface. It includes links for *HOME, GLOBALS, SHARES, PRINTERS, STATUS, VIEW,* and *PASSWORD* and documentation on several important Samba topics.

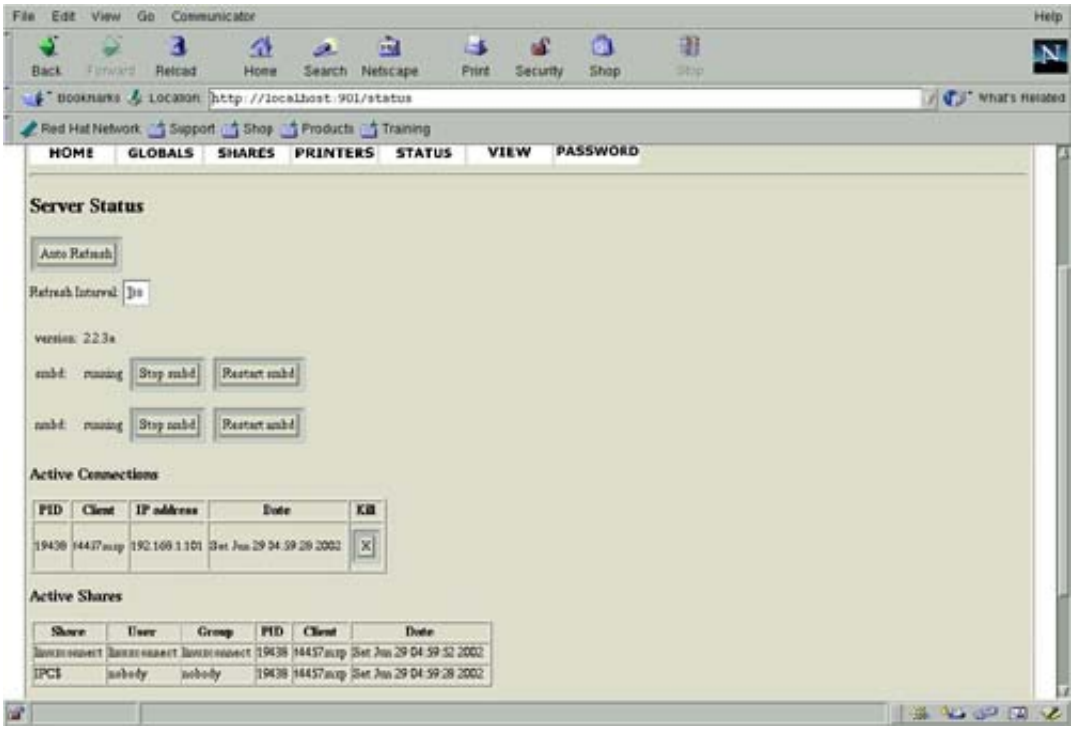
Figure 15-8. SWAT Showing Initial Window



From this point we can view our current configuration and make configuration changes to Samba.

When we select *STATUS,* we are shown the information in Figure 15-9:

Figure 15-9. SWAT *STATUS*



We get such useful information as the state of the *smbd* and *nmbd* daemons, the *Active Connections*, and *Active Shares*.

There is also extensive online documentation for all Samba-related software at www.samba.org.

Samba Utilities and Programs

We have used several Samba utilities and programs in this chapter. The following list gives a description of the most often used Samba-related commands. There are manual pages for all these, which are part of most Samba installations.

- **smbd** - This is the daemon that provides file and print services to SMB clients, such as the Windows system used in our examples throughout this chapter.
- **nmbd** - This is the daemon that provides NetBIOS name server capability and browsing.
- **smbclient** - A program that gives the server access to remotely mounted SMB shares on other servers.
- **testparm** - A test program for **/etc/smb.conf**.
- **smbstatus** - Program that displays status information about current Samba connections.
- **smbpasswd** - Program used to change a user's SMB password on the local machine.
- **smbrun** - Program that runs shell commands for **smbd**.
- **smbtar** - Program to back up SMB shares directly to a UNIX tape drive.
- **smbmount** - Used to mount an SMB file system.
- **smbumount** - Used to unmount an SMB file system.

The online manual pages for these and other Samba-related commands provide more detail. Even in a simple setup such as the one performed in this chapter, you will want to run some of these programs.

Obtaining Samba

In the examples used throughout this chapter, we set up Samba on a Linux system that had Samba installed on it as part of the Red Hat Linux CD-ROM. If Samba does not come on the CD-ROM provided with your UNIX variant, or if you wish to be sure that you're loading the very latest Samba, then you can obtain Samba from the Web.

www.samba.org is the place to start. From this Web site, you can select a "download site" in your country. You can also select "Web sites" on www.samba.org, which provides a wealth of information on Samba, including the GNU General Public License mentioned earlier in the chapter.

There is extensive documentation on Samba-related Web sites, including detailed descriptions of the programs that I listed earlier and used in this chapter.

If you decide to download Samba, you'll probably be given an option of loading a precompiled Samba on your system or building and compiling Samba yourself. The choice you make depends on a lot of factors. If you have a good, reliable Samba distribution, as we did in this chapter when working with Red Hat Linux, then working with a precompiled Samba may be best. If you're interested in learning more about how Samba works and is configured and want the very latest and greatest version, then download the source and compile it yourself.

Even if you have a great prepackaged Samba, as we did in this chapter, it is still worth visiting the Samba-related Web sites to view the extensive documentation available.

[Team LiB]

Section 2: HP-UX Partitions

Chapter 16. Virtual Partitions (vPars)

Chapter 17. nPartitions Overview

Chapter 18. HP Process Resource Manager (PRM)

Chapter 19. MC/ServiceGuard

Chapter 16. Virtual Partitions (vPars)

Introduction

About Virtual Partitions

Virtual Partitions Background

Uses of Virtual Partitions

Preparing to Create Virtual Partitions

Loading the Software Required for Virtual Partitions

Virtual Partitions Command Summary

Steps to Create Virtual Partitions

Virtual Partition Dump Files

Summary

vparmgr GUI

Manual Pages for Commands Used in Chapter 16

Introduction

This chapter is based on the text *HP Virtual Partitions*, published by Prentice Hall PTR ISBN # 0-13-065419-1. All of the topics in this chapter are covered in much more detail in the book.

I refer to chapters in the vPars book many times in this chapter. There is a lot of good vPars background and many useful examples in this chapter; however, vPars are a complex topic on which I have based a separate book, and there is no way I could cover all aspects of vPars in one chapter. I refer to the chapters in the vPars book as a source of more detail on a given topic.

Please see Appendix B for vPars online manual pages.

There is a short section on **vparmgr**, which is a GUI for vPars just introduced at the time of this writing.

About Virtual Partitions

With Virtual Partitions (vPars) you can take almost any HP 9000 server and turn it into many "virtual" computers. These virtual computers can each be running their own instance of HP-UX and associated applications. The virtual computers are isolated from one another at the software level. Software running on one Virtual Partition will not affect software running in any other Virtual Partition. In the Virtual Partitions you can run different revisions of HP-UX, different patch levels of HP-UX, different applications, or any software you want and not affect other partitions.

There are some base requirements that must be met in order to run vPars on your system. At the time of this writing, the following minimum requirements must be met for each vPar on your system:

- Minimum of one CPU.
- Sufficient memory to run HP-UX and any other software that will be present in the vPar.
- A boot disk off which HP-UX can be booted. At the time of this writing it is not possible to share bus adapters between vPars. Therefore, a separate bus adapter is required for each of the vPars. This requirement may have been removed by the time you read this book.
- A console for managing the system. The console can be either physical or virtual. We'll cover the console in detail in the book.
- An HP 9000 system supported by HP-UX 11i. At the time of this writing only HP-UX 11i is supported in vPars. With systems based on the Itanium processor, there are plans to support numerous operating systems in vPars in the future.

The system we'll use in most of the examples throughout this chapter is an rp 5400 (formerly know as L-Class) system that meets all of the requirements in the previous list. You may also want to have additional disks and a separate LAN card in your vPars. I strongly recommend the LAN card so that you can establish TELNET, or other, sessions to your vPars rather than connecting to them only from the console. The LAN card is also required to perform backup and Ignite-UX-related work.

If you have Instant Capacity on Demand (iCOD) employed on your server, all CPUs must be activitated in order for vPars to work. When employing Processor Sets (psets) in a vPar, use only bound CPUs.

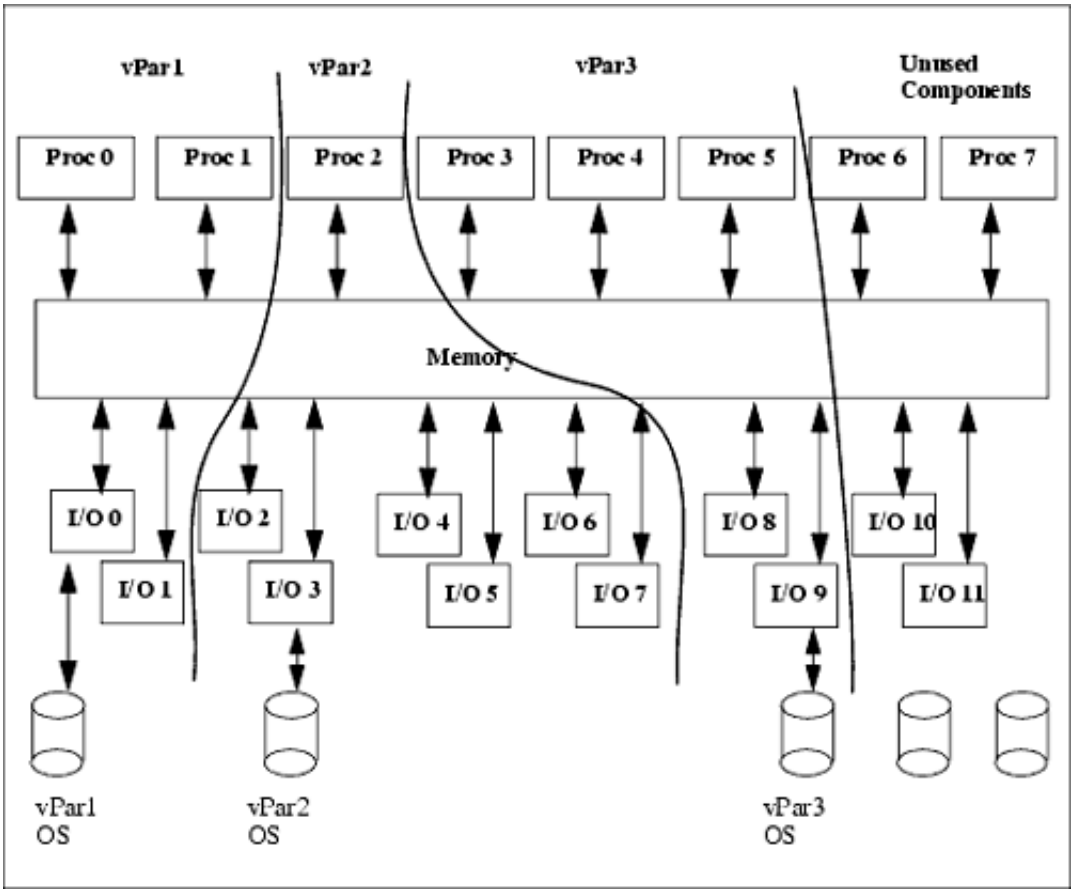
There is a vPars product bundled with HP-UX 11i as well as a full, or add-on product. There are very few limitations with the add-on product. The bundled product has a limitation of a maximum of two vPars, and one of the vPars can have only one CPU.

This chapter was written with Virtual Partitions software that had recently been released for the first time. There have been many enhancements to Virtual Partitions since the writing of this chapter. There is a Graphical User Interface being considered for vPars that I haven't covered in this chapter. There is something to be said for working with a product when it is new. You really get a good understanding of the functionality of a product by using the command line only and performing a lot of manual procedures. In addition, Superdome vPars software is covered in an Appendix in the vPars book but not in this chapter.

Virtual Partitions Background

HP-UX Virtual Partitions (vPars) allow you to run multiple instances of HP-UX on the same HP 9000 server. From a hardware perspective a vPar consists of CPU, memory, and I/O that is a subset of the overall hardware on the computer. From a software perspective a vPar consists of the HP-UX 11i Operating Environment and all application-related software to successfully run your workload. Figure 16-1 shows a conceptual diagram of the way in whcih HP 9000 computer system resources can be allocated to support multiple vPars.

Figure 16-1. Example of HP System Resource Allocation with vPars



The components of which your HP 9000 is comprised can be allocated in a variety of ways. You can see that the eight-way system shown has a different number of processors, different amount of memory, and different number of I/O cards allocated to each vPar. The unused components can be added to any of the vPars or be the basis for yet another vPar. In addition, components can be moved from one vPar to another (with some restrictions described later in the chapter).

Uses of Virtual Partitions

I have worked on many vPars installations that have a variety of uses for vPars. The following are a sampling of the reasons to use vPars:

Increased System Utilization -	Many servers are underutilized. With vPars you can devote a subset of system resources to each vPar. With each vPar running its own instance of HP-UX 11i and associated applications, you'll get higher overall system utilization.
Flexibility -	Many applications have resource needs that change. With vPars you can devote fewer system components when application needs are low and additional resources when an application needs them. An increased end-of-the-month workload, for instance, can be given more system resources to complete faster.
Server Consolidation -	Running multiple instances of HP-UX 11i and their assoicated applications on one HP server reduces the overall number of servers required. Web servers that had run on different servers can now be run in different vPars on the same computer.
Application Isolation -	HP vPars are fully software-isolated from one another. A software failure in one vPar does not affect other vPars.
Mixed Production, Test, and Development -	Production and testing can take place on the same server with vPars. When testing is complete, the test vPar can become the production vPar. Similarly, development usually takes place on a separate system. With the software isolation of vPars, however; development can take place on the same system with other applications.

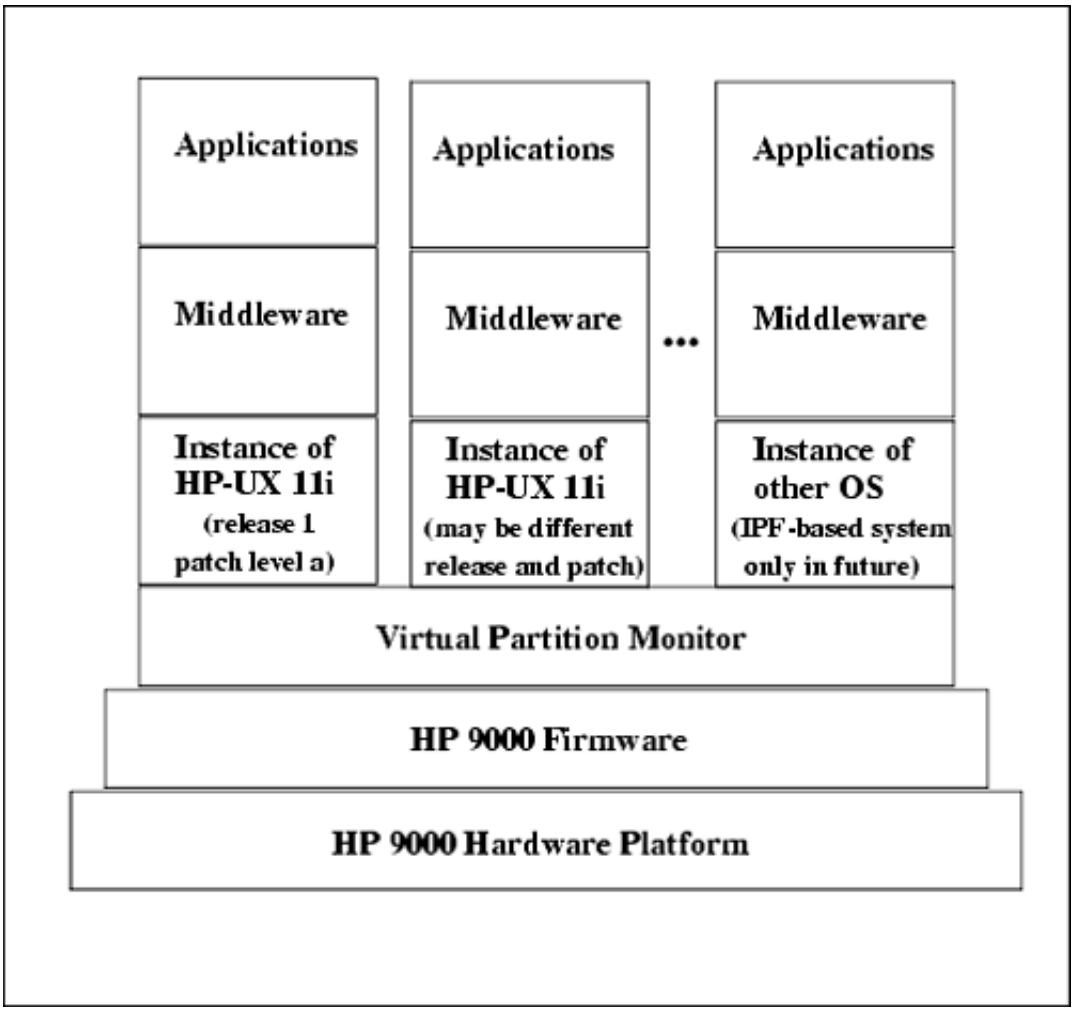
These are just a sampling of the uses I've seen for vPars. Many others will emerge as vPars become widely used and systems experts implement them in more computing environments.

Preparing to Create Virtual Partitions

After having loaded vPars software, we can create partitions. Chapter 1 of the vPars book covers loading HP-UX 11i and vPars software in detail. At the time of this writing, vPars software comes on a separate CD-ROM. Now we're ready to create Virtual Partitions. After we create Virtual Partitions in this chapter, we have to boot the partitions we've created. Although we will boot the partitions in this chapter, the details of booting Virtual Partitions, and HP 9000 booting in general, are covered in detail in Chapter 3 of the vPars book. Chapter 1 of this book covers booting HP-UX in general.

With both HP-UX 11i and the Virtual Partitions software on our disk, we can begin the process of creating partitions. Our goal is to have a system that looks like that in Figure 16-2:

Figure 16-2. Virtual Partitions Software Stack



There are many components in Figure 16-2. We already have many of the components in this diagram on our system. Starting from the bottom, we have the hardware, firmware, Virtual Partition Monitor (covered as part of the boot process in Chapter 3 of the vPars book), and HP-UX 11i installed on two different disks. There are two HP-UX 11i instances shown in the leftmost two stacks of Figure 16-2. These are the operating systems that we have already loaded.

The two HP-UX 11i instances can't run simultaneously on our rp5400 (formerly know as L-Class) system because we have not yet created our Virtual Partitions. Without Virtual Partitions created, we can boot HP-

UX off of one *or* the other of these disks, but we can't run both. Let's now create our Virtual Partitions so that we can indeed have two instances of HP-UX 11i running simultaneously. After Virtual Partitions have been created, you can proceed to load the middleware and applications shown on top of HP-UX 11i in Figure 16-2.

Although we're covering running multiple instances of HP-UX 11i in the Virtual Partitions we create, we won't be restricted to only HP-UX 11i in the future. On systems based on the Itanium processors, we'll be able to run additional operating systems. The rightmost Virtual Partition In Figure 16-2 depicts this future capability. This capability does not exist at the time of this writing, so it is not covered. Look for an updated revision of the vPars book in the future to cover this capability once it exists and keep an eye on www.hp.com for both IPF and Virtual Partition enhancements.

Although the examples in this chapter take place on four-way (four-processor) rp5400 (formerly know as L-Class) systems, vPars run on most HP servers. The more components of which your HP server is comprised, the more vPars you can have on the server and the more options you have for crafting vPars. In the examples in this chapter we'll end up with a four-way rp5400 (formerly know as L-Class) with two fully configured Virtual Partitions. Next we'll cover loading software.

[Team LiB]

PREVIOUS NEXT

Loading the Software Required for Virtual Partitions

We cover installing Virtual Partitions software in this section. I assume that you already have HP-UX 11i installed on your system or know how to do so. If you have not yet installed HP-UX 11i on your system, please see Chapter 2, which covers installing HP-UX 11i.

Keep in mind that HP-UX must be loaded for each Virtual Partition you wish to run. If, for instance, you want to run two Virtual Partitions, as we do in our examples in this book, HP-UX 11i will need to be loaded for both Virtual Partitions. The procedure covered for loading HP-UX 11i needs to be performed for every Virtual Partition you want to run. HP-UX 11i can be loaded from media, such as your HP-UX 11i distribution on a CD-ROM or from an Ignite/UX server. You can use any method to load HP-UX 11i and the Virtual Partitions software for every Virtual Partition you want to run.

At the time of this writing there are two vPars products: a product with the full functionality covered throughout the book and a free product that has a subset of the full product. The free product has a limitation of a maximum of two vPars ,and one of the vPars can have only one CPU.

When you buy the full product, product number T1355AC, a CD-ROM is provided that has on it the following components:

- vPars software
- patches
- vPars administration guide
- booklet
- WINSTALL files used for booting

At some point the full product will be on standard distribution rather than a separate CD-ROM. The free product, called VPARSBASE at the time of this writing, can be downloaded from www.software.hp.com.

Figure 16-3 shows an example of the software components that appear below the full product T1355AC.

Figure 16-3. Example of Loading vPars Software

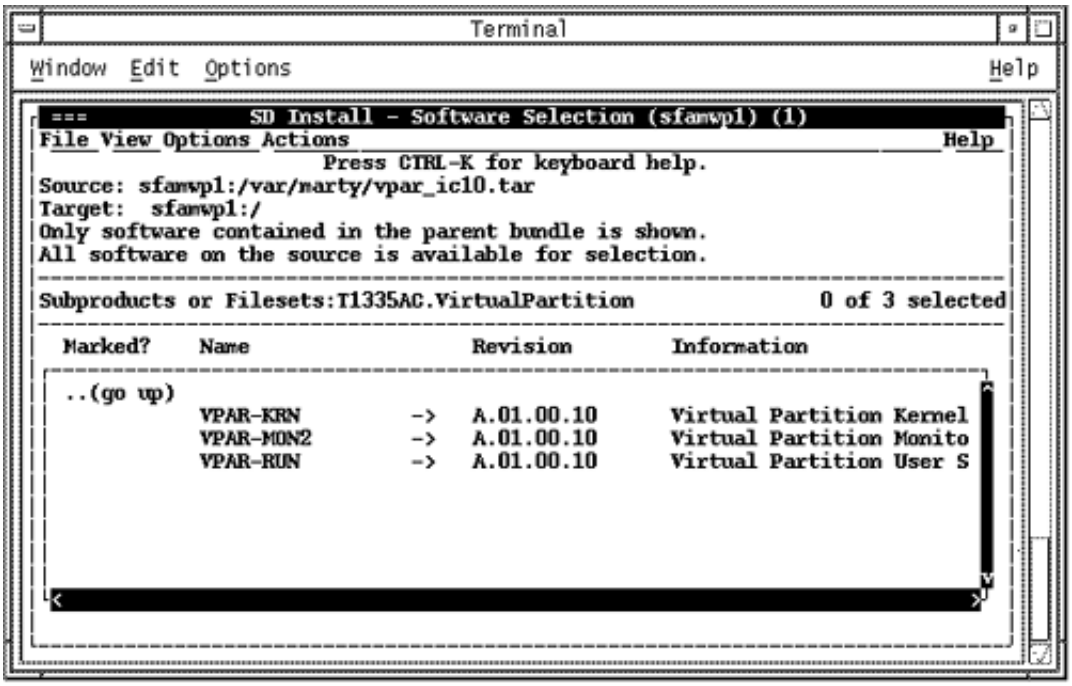


Figure 16-3 shows the three components of which T1355AC is comprised: the kernel, monitor, and run environment. The additional components listed earlier, patches and WINSTALL, will also have to be loaded for vPars to be fully operational.

After loading this software, a reboot takes place to build the kernel. This is done for you automatically, however; vPars kernel-related background is covered in both Chapters 2 and 4 of the vPars book.

All of the vPars software must be loaded on every HP-UX 11i volume that will be used on your vPars server. The loading of this software will take place for every HP-UX 11i instance that you wish to run simultaneously on your vPars server. There are two ways to load the HP-UX 11i operating system and vPars software on all of the volumes used for vPars. The first, which is the method used throughout this book, is to load HP-UX 11i and vPars software on all vPars volumes prior to creating Virtual Partitions. The second is to load only the volume of the first vPar with all software, create as many vPars as you want, and then use **vparboot -p** *vp_name* **-I** *ignite_kernel* to boot and load HP-UX 11i on the other disks. In this chapter I first load HP-UX 11i and vPars software on all disks before creating vPars.

A lot of software has been loaded as a result of loading the vPars software. The **/sbin** directory has in it the *vpar* commands we'll use in upcoming sections. The following is a long listing of the *vpar* commands in **/sbin**.

```
# ll /sbin/vpar*
-r-xr-xr-x   1 bin        bin         128760 Oct 18 22:08 /sbin/vparboot
-r-xr-xr-x   1 bin        bin         161216 Oct 18 22:07 /sbin/vparcreate
-r-xr-xr-x   1 bin        bin         101040 Oct 18 22:04 /sbin/vpard
-r-xr-xr-x   1 bin        bin          59592 Oct 18 22:04 /sbin/vpardump
-r-xr-xr-x   1 bin        bin          30520 Oct 18 22:04 /sbin/vparextract
-r-xr-xr-x   1 bin        bin         140072 Oct 18 22:08 /sbin/vparmodify
-r-xr-xr-x   1 bin        bin          47232 Oct 18 22:04 /sbin/vparreloc
-r-xr-xr-x   1 bin        bin         127808 Oct 18 22:08 /sbin/vparremove
-r-xr-xr-x   1 bin        bin         132008 Oct 18 22:08 /sbin/vparreset
-r-xr-xr-x   1 bin        bin         152040 Oct 18 22:08 /sbin/vparstatus
-r-xr-xr-x   1 bin        bin          26152 Oct 18 22:04 /sbin/vparutil
#
```

These are the commands that you'll use to create, view, modify, and work with vPars in general. We'll use some of these commands in this chapter. Chapter 2 of the vPars book is devoted to describing these commands and giving examples of using most of them in their various forms. In addition, the tear-out card included with the book summarizes many of these commands.

There are several files in **/stand** related to the vPars kernel. The following listing shows some of these:

```
# ll /stand/vp*
-rw-------   1 root       root          8232 Nov 16 07:09 /stand/vpdb
-r-xr-xr-x   1 bin        bin         849992 Oct 18 22:02 /stand/vpmon
```

```
#
```

vpmon is loaded at the time of system startup and is the basis for running vPars. Chapter 3 of the vPars book covers booting in detail, including bringing up **vpmon**. **vpdb** is the vPars database that contains all information related to all of the vPars running on your system. This file is automatically synchronized by the vPars application to ensure that all vPars have the same information about all vPars on your system.

There are several startup-related files, including those shown below, which are covered in more detail in Chapter 8 of the vPars book covering startup.

```
/etc/rc.config.d/vpard
/etc/rc.config.d/vparhb
/etc/rc.config.d/vparinit

/sbin/init.d/vpard
/sbin/init.d/vparhb
/sbin/init.d/vparinit
```

Of particular interest is **vparhp**, which is the *heartbeat* daemon related to keeping **vpdb** synchronized on all of your vPars.

Very important to your work related to vPars are the online man pages. The following listing shows the man pages loaded on my system at the time of this writing.

```
# man -k vpar
vparboot(1M)       - boot a virtual partition
vparcreate(1M)     - create a virtual partition
vpardump(1M)       - manage monitor dump files
vparextract(1M)    - extract memory images from a running virtual
                     partition system
vparmodify(1M)     - modify the attributes of a virtual partition
vparreloc(1M)      - relocate the load address of a vmunix file,
                     determine if a vmunix file is relocatable,
                     or promote the scope of symbols in a
                     relocatable vmunix file
vparremove(1M)     - remove a virtual partition
vparreset(1M)      - reset a virtual partition
vparresources(5)   - description of virtual partition resources
                     and their requirements
vparstatus(1M)     - display information about one or more
                     virtual partitions
vpartition(1)      - display information about the Virtual
                     Partition Command Line Interface
vparutil(1M)       - get and set SCSI parameters for SCSI
                     controllers from a virtual partition
#
```

Many of these man pages appear in Appendix B, so you can refer to them as you read the upcoming sections.

At this point we have HP-UX 11i and the Virtual Partitions software loaded on the system.

Virtual Partitions Command Summary

There are several commands used to create and work with Virtual Partitions. A table in the vPars book as well as a tear-out card provide an overview of many commonly used Virtual Partitions related commands. Table 16-1 is an abbreviated version of the command summary:

Table 16-1. Virtual Partition Commands

Command	Description

ISL>
 	
Initial System Load prompt.
	Virtual Partitions Monitor is loaded from *ISL>* with:
	`ISL> hpux /stand/vpmon`
	`MON>`
	To load Virtual Partitions directly from *ISL>,* use:
	`ISL> hpux /stand/vpmon vparload -p vPar_name`

MON>
 	
Virtual Partitions Monitor prompt. (Also see **vparload** command.)
	This is loaded from *ISL* with:
	`ISL> hpux /stand/vpmon`
	`MON>`
	To load an alternate database from *ISL,* use:
	`ISL> hpux /stand/vpmon -D db_file`
	To load one vPar from *MON*, use:
	`MON> vparload vPar_name`
	Many other commands can be issued from *MON*. Type **help** or **?** to list. (Commands include: **scan, vparinfo, ls, log, getauto, lifls, cbuf, cat**.)

vparload Load Virtual Partitions from *MON>* prompt only.	To boot a Virtual Partition from *MON>*: `MON> vparload -p vPar_name`
vparboot Boot a Virtual Partition from the command line only.	To boot a Virtual Partition from the command line: `# vparboot -p vPar_name`
vparcreate Create a Virtual Partition.	To create a Virtual Partition with three processors (*num*) total, two bound (*min*), 2048MB RAM, all components on 0/0, boot disk at 0/0/1/1.2.0, with a kernel of **/stand/vmunix**, autoboot on, and console at 0/0/4/0: `# vparcreate -p vPar_name -a cpu::3 -a cpu:::2:4 -a mem::2048 -a io:0/0 -a io:0/0/1/1.2.0:boot -b /stand/vmunix -B auto`
vparmodify	To add processor at path *109* (adds this proc to those already assigned): `# vparmodify -p vPar_name -a cpu:109`

Modify the attributes of a Virtual Partition.	
vparremove	To delete a Virtual Partition in the currently running database: `# vparremove -p vPar_name`
Delete a Virtual Partition.	
vparreset	To reset a Virtual Partition without TOC (t), hard (h), bypassing display of PIM data (q), or forcing (f): `# vparreset -p vPar_name`
Reset a Virtual Partition.	
vparresources(5) man page	This is a manual page that describes Virtual Partition resources in general and how resources are specified in other commands, such as **vparmodify**.
Provides description of Virtual Partitions and their resources.	
vparstatus	To display the status of a Virtual Partition in verbose mode: `# vparstatus -v -p vPar_name`
Display the status of Virtual Partitions.	
vpartition man page	Provides the following brief description of Virtual Partitions commands:
Display information about the Virtual Partition Command Line Interface.	**vparboot** Boot (start) a virtual partition. **vparcreate** Create a new virtual partition. **vparmodify** Modify an existing virtual partition. **vparremove** Remove (delete) an existing virtual partition. **vparreset** Simulate a TOC or hard reset to a virtual partition. **vparstatus** Display virtual partition and available resources information.
Specify CPU Resources by:	Number of bound and unbound CPUs: *cpu::num* CPU hardware path(s): *cpu:path* Minimum and maximum number: *cpu:::[min][:[max]]*
Specify Memory by:	Size *mem::size* Base and range: *mem:::base:range* combination of both above.
Specify I/O:	Use path: *io:path[:attr1[,attr2[...]]]* (see man page **vparresources** for details).
To add resources use: (This adds component relative to what already exists if running **vparmodify**.)	*-a cpu:path* *-a cpu::num* (can be done with vPar running) *[-a cpu::num] [-a cpu:::[min]:[max]] -* *[a cpu:path]* (*:::* is **vparcreate** only) *-a io:path[:attr1[,attr2[...]]]* *-a mem::size* *-a mem:::base:range*
To delete resources use (This deletes component relative to what already exists if running	*-d cpu:path* *-d cpu::num* (can be done with vPar running) *-d io:path[:attr1[,attr2[...]]*

vparmodify.)	*-d mem::size* *-d mem:::base:range*
To modify resources use: (This modifies to absolute number rather than relative.)	*-m cpu::num* (can be done with vPar running) *-m cpu:::[min][:max]]* *-m io:path[:attr1[,attr2[...]]* *-m mem::size*
vPars **setboot** Options:	
-a	Changes the alternate boot path of the Virtual Partition.
-b	Sets the autoboot attribute of the Virtual Partition.
-p	Changes the primary boot path of the Virtual Partition.
-s	No effect.
no options	Displays information about boot attributes. To set Autoboot *on*: `# setboot -b on`
vPars States:	
load	The kernel image of a Virtual Partition is being loaded into memory. This is done by the Virtual Partition monitor.
boot	The Virtual Partition is in the process of booting. The kernel image has been successfully loaded by the Virtual Partition monitor.
up	The Virtual Partition has been successfully booted and is running.
shut	The Virtual Partition is in the process of shutting down.
down	The Virtual Partition is not running and is down.
crash	The Virtual Partition has experienced a panic and is crashing.
hung	The Virtual Partition is not responding and is hung.

We'll use some of the commands shown in Table 16-1 in the upcoming section on creating virtual partitions. There is more detail on the Virtual Partition commands in Appendix B, which contains the online manual pages for the commands.

Let's now move on to creating our virtual partitions.

[Team LiB]

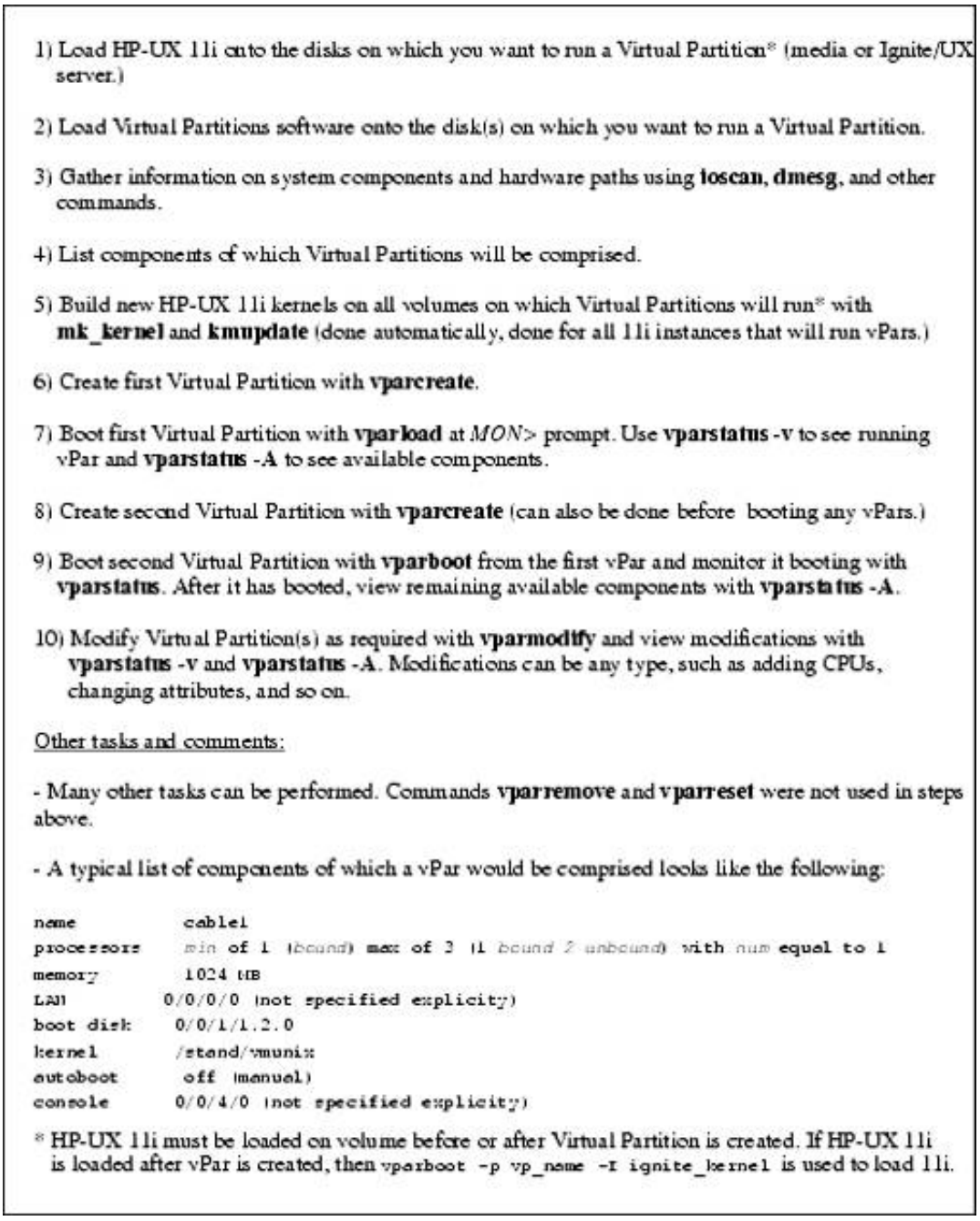
PREVIOUS NEXT

Steps to Create Virtual Partitions

In this section we'll cover the steps to create Virtual Partitions. These are steps I performed while working with vPars with some of the very first installations. This list should serve as a framework for working with vPars. You may chose not to perform some of the steps and to add others. It is only a framework for getting vPars working on your system.

In our upcoming examples to create our Virtual Partitions, we'll execute the steps shown in Figure 16-4.

Figure 16-4. Steps to Create Virtual Partitions



1) Load HP-UX 11i onto the disks on which you want to run a Virtual Partition* (media or Ignite/UX server.)

2) Load Virtual Partitions software onto the disk(s) on which you want to run a Virtual Partition.

3) Gather information on system components and hardware paths using **ioscan**, **dmesg**, and other commands.

4) List components of which Virtual Partitions will be comprised.

5) Build new HP-UX 11i kernels on all volumes on which Virtual Partitions will run* with **mk_kernel** and **kmupdate** (done automatically, done for all 11i instances that will run vPars.)

6) Create first Virtual Partition with **vparcreate**.

7) Boot first Virtual Partition with **vparload** at *MON>* prompt. Use **vparstatus -v** to see running vPar and **vparstatus -A** to see available components.

8) Create second Virtual Partition with **vparcreate** (can also be done before booting any vPars.)

9) Boot second Virtual Partition with **vparboot** from the first vPar and monitor it booting with **vparstatus**. After it has booted, view remaining available components with **vparstatus -A**.

10) Modify Virtual Partition(s) as required with **vparmodify** and view modifications with **vparstatus -v** and **vparstatus -A**. Modifications can be any type, such as adding CPUs, changing attributes, and so on.

Other tasks and comments:

- Many other tasks can be performed. Commands **vparremove** and **vparreset** were not used in steps above.

- A typical list of components of which a vPar would be comprised looks like the following:

```
name          cable1
processors    min of 1 (bound) max of 2 (1 bound 2 unbound) with num equal to 1
memory        1024 MB
LAN1          0/0/0/0 (not specified explicity)
boot disk     0/0/1/1.2.0
kernel        /stand/vmunix
autoboot      off (manual)
console       0/0/4/0 (not specified explicity)
```

* HP-UX 11i must be loaded on volume before or after Virtual Partition is created. If HP-UX 11i is loaded after vPar is created, then `vparboot -p vp_name -I ignite_kernel` is used to load 11i.

1) Load HP-UX 11i

HP-UX 11i must be loaded on the volumes that will be used to host all vPars. The method you use to install 11i, whether media, Ignite-UX, or some other technique, are all acceptable provided that HP-UX 11i is present on all of the disks. HP-UX 11i must be present on the first disk before you begin the vPar creation. You can create vPars on other disks before HP-UX 11i is loaded on them and then use **vparboot -p** *vp_name* **-I** *ignite_kernel* to boot and load HP-UX 11i on the other disks. In this chapter I first load HP-UX 11i on all disks before creating vPars. Chapter 1 covers loading HP-UX 11i and vPars software. In our upcoming example the first Virtual Partition will be created on the internal disk on an rp5400 (formerly know as L-Class) system. The second Virtual Partition will be created on an external disk. After loading HP-UX 11i on one of the root volumes, I issued **uname**, which resulted in the following output:

```
# uname -a
HP-UX cvhdcon3 B.11.11 U 9000/800 136414696 unlimited-user license
```

```
#
```

The hostname of *cvhdcon3* and HP-UX revision *11.11*, which is the latest HP-UX 11i available at the time of this writing, are shown.

An interesting nuance to working with vPars is the naming of hosts and vPars. In a nutshell, you supply hostnames when installing 11i and Virtual Partition names when creating vPars. It reduces confusion if both the hostname and vPar name are the same for an instance of HP-UX 11i. In some cases, however, organizations require hostnames to conform to conventions that result in names that are difficult to remember. In this case some system administrators pick easy-to-remember vPar names. In the upcoming example we used different names for the vPars and hostnames.

Our upcoming examples have hostnames of *cvhdcon3* and *cvhdcon4*. The respective vPar names used are *cable1* and *cable2*.

2) Load the Virtual Partitions Application Software

The Virtual Partitions software must also be loaded on the volumes that will be used to host all vPars. At the time of this writing there are *base* and *full* versions of the software. The restrictions of the *base* software are a maximum of two vPars and a maximum of one CPU in one of the vPars. After loading the vPars software on one of the root volumes, I ran **swlist**, which resulted in the following output:

```
# swlist
# Initializing...
# Contacting target "cvhdcon3"...
#
# Target:  cvhdcon3:/
#

#
# Bundle(s):
#

  BUNDLE11i             B.11.11.0102.2 Required Patch Bundle for HP-UX 11i, Febr
uary 2001
  CDE-English           B.11.11        English CDE Environment
  FDDI-00               B.11.11.01     PCI FDDI;Supptd HW=A3739A/A3739B;SW=J3626
AA
  FibrChanl-00          B.11.11.06     PCI/HSC FibreChannel;Supptd HW=A6684A,A66
85A,A5158A
  GigEther-00           B.11.11.14     PCI/HSC GigEther;Supptd HW=A4926A/A4929A/
A4924A/A4925A;SW=J1642AA
  HPUX11i-OE-MC         B.11.11.0106   HP-UX Mission Critical Operating Environm
ent Component
  HPUXBase64            B.11.11        HP-UX 64-bit Base OS
  HPUXBaseAux           B.11.11.0106   HP-UX Base OS Auxiliary
  HWEnable11i           B.11.11.0106.8 Hardware Enablement Patches for HP-UX 11i
, June 2001
  OnlineDiag            B.11.11.03.08  HPUX 11.11 Support Tools Bundle, Jun 2001

  RAID-00               B.11.11.01     PCI RAID; Supptd HW=A5856A
  VPARSBASE             A.01.00.03     HP-UX Virtual Partitions
#
```

The *HP-UX Virtual Partitions* software is the last entry shown in this listing.

3) Gather the System Component and Hardware Paths

You get to know your hardware at an intimate level when working with vPars. You not only need to know the components of which your system is comprised, you also need to know the paths of much of the hardware. Some system components, such as *System Bus Adapters* and the *memory controller*, are shared among vPars, so you don't specify those components as part of individual Virtual Partitions. Most other components in your system, such as processors, I/O cards, disks, and others, are fixed to specific vPars.

In order to see the components of which our example system is comprised,we'll run **ioscan -f** and **dmesg** in the following listing:

```
# ioscan -f
```

```
Class       I  H/W Path       Driver    S/W State   H/W Type     Description
=============================================================================
root        0                 root      CLAIMED     BUS_NEXUS
ioa         0  0              sba       CLAIMED     BUS_NEXUS    System Bus
                                                                  Adapter (803)
ba          0  0/0            lba       CLAIMED     BUS_NEXUS    Local PCI Bus
                                                                  Adapter (782)
lan         0  0/0/0/0        btlan     CLAIMED     INTERFACE    HP PCI 10/100
                                                                  Base-TX Core
ext_bus     0  0/0/1/0        c720      CLAIMED     INTERFACE    SCSI C896
                                                                  Ultra Wide Single-Ended
target      0  0/0/1/0.1      tgt       CLAIMED     DEVICE
disk        0  0/0/1/0.1.0    sdisk     CLAIMED     DEVICE       HP DVD-ROM 304
target      1  0/0/1/0.3      tgt       CLAIMED     DEVICE
tape        0  0/0/1/0.3.0    stape     CLAIMED     DEVICE       HP      C1537A
target      2  0/0/1/0.7      tgt       CLAIMED     DEVICE
ctl         0  0/0/1/0.7.0    sctl      CLAIMED     DEVICE       Initiator
ext_bus     1  0/0/1/1        c720      CLAIMED     INTERFACE    SCSI C896
                                                                  Ultra Wide Single-Ended
target      3  0/0/1/1.0      tgt       CLAIMED     DEVICE
disk        1  0/0/1/1.0.0    sdisk     CLAIMED     DEVICE       SEAGATE ST17340 4LC
target      4  0/0/1/1.2      tgt       CLAIMED     DEVICE
disk        2  0/0/1/1.2.0    sdisk     CLAIMED     DEVICE       SEAGATE ST17340 4LC
target      5  0/0/1/1.7      tgt       CLAIMED     DEVICE
ctl         1  0/0/1/1.7.0    sctl      CLAIMED     DEVICE       Initiator
ext_bus     2  0/0/2/0        c720      CLAIMED     INTERFACE    SCSI C87x
                                                                  Ultra Wide Single-Ended
target      6  0/0/2/0.0      tgt       CLAIMED     DEVICE
disk        3  0/0/2/0.0.0    sdisk     CLAIMED     DEVICE       SEAGATE ST17340 4LC
target      7  0/0/2/0.2      tgt       CLAIMED     DEVICE
disk        4  0/0/2/0.2.0    sdisk     CLAIMED     DEVICE       SEAGATE ST17340 4lc
target      8  0/0/2/0.7      tgt       CLAIMED     DEVICE
ctl         2  0/0/2/0.7.0    sctl      CLAIMED     DEVICE       Initiator
ext_bus     3  0/0/2/1        c720      CLAIMED     INTERFACE    SCSI C87x
                                                                  Ultra Wide Single-Ended
target      9  0/0/2/1.7      tgt       CLAIMED     DEVICE
ctl         3  0/0/2/1.7.0    sctl      CLAIMED     DEVICE       Initiator
tty         0  0/0/4/0        asio0     CLAIMED     INTERFACE    PCI Serial
tty         1  0/0/5/0        asio0     CLAIMED     INTERFACE    PCI Serial
ba          1  0/1            lba       CLAIMED     BUS_NEXUS    Local PCI Bus
                                                                  Adapter (782)
ba          2  0/2            lba       CLAIMED     BUS_NEXUS    Local PCI Bus
                                                                  Adapter (782)
ba          3  0/3            lba       CLAIMED     BUS_NEXUS    Local PCI Bus
                                                                  Adapter (782)
ba          4  0/4            lba       CLAIMED     BUS_NEXUS    Local PCI Bus
                                                                  Adapter (782)
ba          5  0/5            lba       CLAIMED     BUS_NEXUS    Local PCI Bus
                                                                  Adapter (782)
ba          6  0/8            lba       CLAIMED     BUS_NEXUS    Local PCI Bus
                                                                  Adapter (782)
fc          0  0/8/0/0        td        CLAIMED     INTERFACE    HP Tachyon TL/TS
                                                          Fibre Channel Mass Storage Adapter
fcp         0  0/8/0/0.8      fcp       CLAIMED     INTERFACE    FCP Protocol
                                                                  Adapter
ext_bus     4  0/8/0/0.8.0.5.0        fcparray  CLAIMED     INTERFACE    FCP
                                                                  Array Interface
target     10  0/8/0/0.8.0.5.0.0      tgt       CLAIMED     DEVICE
disk        5  0/8/0/0.8.0.5.0.0.0    sdisk     CLAIMED     DEVICE  HP A5277A
disk        6  0/8/0/0.8.0.5.0.0.1    sdisk     CLAIMED     DEVICE  HP A5277A
disk        7  0/8/0/0.8.0.5.0.0.2    sdisk     CLAIMED     DEVICE  HP A5277A
disk        8  0/8/0/0.8.0.5.0.0.3    sdisk     CLAIMED     DEVICE  HP A5277A
target     11  0/8/0/0.8.0.5.0.1      tgt       CLAIMED     DEVICE
disk        9  0/8/0/0.8.0.5.0.1.0    sdisk     CLAIMED     DEVICE  HP A5277A
target     12  0/8/0/0.8.0.5.0.2      tgt       CLAIMED     DEVICE
disk       10  0/8/0/0.8.0.5.0.2.0    sdisk     CLAIMED     DEVICE  HP A5277A
target     13  0/8/0/0.8.0.5.0.3      tgt       CLAIMED     DEVICE
disk       11  0/8/0/0.8.0.5.0.3.0    sdisk     CLAIMED     DEVICE  HP A5277A
ext_bus     5  0/8/0/0.8.0.255.0      fcpdev    CLAIMED     INTERFACE    FCP
                                                                  Device Interface
target     14  0/8/0/0.8.0.255.0.5    tgt       CLAIMED     DEVICE
ctl         4  0/8/0/0.8.0.255.0.5.0  sctl      CLAIMED     DEVICE  HP A5277A
ba          7  0/9                    lba       CLAIMED     BUS_NEXUS Local PCI
                                                                  Bus Adapter (782)
fc          1  0/9/0/0                 td       CLAIMED     INTERFACE HP Tachyon
```

```
                                        TL/TS Fibre Channel Mass Storage Adapter
fcp          1  0/9/0/0.8                fcp      CLAIMED     INTERFACE FCP Protocol
                                                               Adapter
ext_bus      6  0/9/0/0.8.0.4.0          fcparray  CLAIMED     INTERFACE FCP Array
                                                               Interface
target      15  0/9/0/0.8.0.4.0.0        tgt       CLAIMED     DEVICE
disk        12  0/9/0/0.8.0.4.0.0.0      sdisk     CLAIMED     DEVICE HP A5277A
disk        13  0/9/0/0.8.0.4.0.0.1      sdisk     CLAIMED     DEVICE HP A5277A
disk        14  0/9/0/0.8.0.4.0.0.2      sdisk     CLAIMED     DEVICE HP A5277A
disk        15  0/9/0/0.8.0.4.0.0.3      sdisk     CLAIMED     DEVICE HP A5277A
target      16  0/9/0/0.8.0.4.0.1        tgt       CLAIMED     DEVICE
disk        16  0/9/0/0.8.0.4.0.1.0      sdisk     CLAIMED     DEVICE HP A5277A
target      17  0/9/0/0.8.0.4.0.2        tgt       CLAIMED     DEVICE
disk        17  0/9/0/0.8.0.4.0.2.0      sdisk     CLAIMED     DEVICE HP A5277A
target      18  0/9/0/0.8.0.4.0.3        tgt       CLAIMED     DEVICE
disk        18  0/9/0/0.8.0.4.0.3.0      sdisk     CLAIMED     DEVICE HP A5277A
ext_bus      7  0/9/0/0.8.0.255.0        fcpdev    CLAIMED     INTERFACE FCP
                                                               Device Interface
target      19  0/9/0/0.8.0.255.0.4      tgt       CLAIMED     DEVICE
ctl          5  0/9/0/0.8.0.255.0.4.0    sctl      CLAIMED     DEVICE HP A5277A
ba           8  0/10                     lba       CLAIMED     BUS_NEXUS Local PCI
                                                               Bus Adapter (782)
lan          1  0/10/0/0                btlan     CLAIMED     INTERFACE HP A5230A/
                                                    B5509BA PCI 10/100Base-TX Addon
ba           9  0/12                     lba       CLAIMED     BUS_NEXUS Local PCI
                                                               Bus Adapter (782)
lan          2  0/12/0/0                btlan     CLAIMED     INTERFACE    HP A5230A/
                                                    B5509BA PCI 10/100Base-TX Addon
pbc          0  32              pbc       CLAIMED     BUS_NEXUS    Bus Converter
processor    0  33              processor CLAIMED     PROCESSOR    Processor
pbc          1  36              pbc       CLAIMED     BUS_NEXUS    Bus Converter
processor    1  37              processor CLAIMED     PROCESSOR    Processor
pbc          2  96              pbc       CLAIMED     BUS_NEXUS    Bus Converter
processor    2  97              processor CLAIMED     PROCESSOR    Processor
pbc          3  100             pbc       CLAIMED     BUS_NEXUS    Bus Converter
processor    3  101             processor CLAIMED     PROCESSOR    Processor
memory       0  192             memory    CLAIMED     MEMORY       Memory
#
# dmesg

Jul 31 20:03
gate64: sysvec_vaddr = 0xc0002000 for 2 pages
NOTICE: nfs3_link(): File system was registered at index 3.
NOTICE: autofs_link(): File system was registered at index 6.
NOTICE: cachefs_link(): File system was registered at index 7.
0 sba
0/0 lba
0/0/0/0 btlan
0/0/1/0 c720
0/0/1/0.1 tgt
0/0/1/0.1.0 sdisk
0/0/1/0.3 tgt
0/0/1/0.3.0 stape
0/0/1/0.7 tgt
0/0/1/0.7.0 sctl
0/0/1/1 c720
0/0/1/1.0 tgt
0/0/1/1.0.0 sdisk
0/0/1/1.2 tgt
0/0/1/1.2.0 sdisk
0/0/1/1.7 tgt
0/0/1/1.7.0 sctl
0/0/2/0 c720
0/0/2/0.0 tgt
0/0/2/0.0.0 sdisk
0/0/2/0.2 tgt
0/0/2/0.2.0 sdisk
0/0/2/0.7 tgt
0/0/2/0.7.0 sctl
0/0/2/1 c720
0/0/2/1.7 tgt
0/0/2/1.7.0 sctl
0/0/4/0 asio0
0/0/5/0 asio0
0/1 lba
```

```
0/2 lba
0/3 lba
0/4 lba
0/5 lba
0/8 lba
0/8/0/0 td
td: claimed Tachyon TL/TS Fibre Channel Mass Storage card at 0/8/0/0
0/8/0/0.8 fcp
0/8/0/0.8.0.5.0 fcparray
0/8/0/0.8.0.5.0.0 tgt
0/8/0/0.8.0.5.0.0.0 sdisk
0/8/0/0.8.0.5.0.0.1 sdisk
0/8/0/0.8.0.5.0.0.2 sdisk
0/8/0/0.8.0.5.0.0.3 sdisk
0/8/0/0.8.0.5.0.1 tgt
0/8/0/0.8.0.5.0.1.0 sdisk
0/8/0/0.8.0.5.0.2 tgt
0/8/0/0.8.0.5.0.2.0 sdisk
0/8/0/0.8.0.5.0.3 tgt
0/8/0/0.8.0.5.0.3.0 sdisk
0/8/0/0.8.0.255.0 fcpdev
0/8/0/0.8.0.255.0.5 tgt
0/8/0/0.8.0.255.0.5.0 sctl
0/9 lba
0/9/0/0 td
td: claimed Tachyon TL/TS Fibre Channel Mass Storage card at 0/9/0/0
0/9/0/0.8 fcp
0/9/0/0.8.0.4.0 fcparray
0/9/0/0.8.0.4.0.0 tgt
0/9/0/0.8.0.4.0.0.0 sdisk
0/9/0/0.8.0.4.0.0.1 sdisk
0/9/0/0.8.0.4.0.0.2 sdisk
0/9/0/0.8.0.4.0.0.3 sdisk
0/9/0/0.8.0.4.0.1 tgt
0/9/0/0.8.0.4.0.1.0 sdisk
0/9/0/0.8.0.4.0.2 tgt
0/9/0/0.8.0.4.0.2.0 sdisk
0/9/0/0.8.0.4.0.3 tgt
0/9/0/0.8.0.4.0.3.0 sdisk
0/9/0/0.8.0.255.0 fcpdev
0/9/0/0.8.0.255.0.4 tgt
0/9/0/0.8.0.255.0.4.0 sctl
0/10 lba
0/10/0/0 btlan
0/12 lba
0/12/0/0 btlan
32 pbc
33 processor
36 pbc
37 processor
96 pbc
97 processor
100 pbc
101 processor
192 memory
btlan: Initializing 10/100BASE-TX card at 0/0/0/0....

    System Console is on the Built-In Serial Interface
btlan: Initializing 10/100BASE-TX card at 0/10/0/0....
btlan: Initializing 10/100BASE-TX card at 0/12/0/0....
Entering cifs_init...
Initialization finished successfully... slot is 9
Logical volume 64, 0x3 configured as ROOT
Logical volume 64, 0x2 configured as SWAP
Logical volume 64, 0x2 configured as DUMP
    Swap device table:  (start & size given in 512-byte blocks)
        entry 0 - major is 64, minor is 0x2; start = 0, size = 8388608
    Dump device table:  (start & size given in 1-Kbyte blocks)
        entry 0000000000000000 - major is 31, minor is 0x12000; start = 117600,
size = 4194304
Starting the STREAMS daemons-phase 1
Create STCP device files
Starting the STREAMS daemons-phase 2
        $Revision: vmunix:    vw: -proj    selectors: CUPI80_BL2000_1108 -c 'Vw
for CUPI80_BL2000_1108 build' -- cupi80_bl2000_1108 'CUPI80_BL2000_1108'  Wed
```

```
Nov  8 19:24:56 PST 2000 $
Memory Information:
  physical page size = 4096 bytes, logical page size = 4096 bytes
  Physical: 4194304 Kbytes, lockable: 3231756 Kbytes, available: 3711728 Kbytes

#
```

The output of **ioscan -f** and **dmesg** provide a lot of useful information about our system. We'll use the components and paths in **ioscan** output and the memory information in **dmesg** to create a list of components for the respective vPars in the upcoming step. We now know, for instance, that the paths of two of the LAN cards are at *0/0/0/0* and *0/10/0/0*. We know the paths of all four processors of *33, 37, 97,* and *101*. The console is located at *0/0/4/0*. From the **dmesg** output we know that we have a total of four GBytes of RAM that can be spread among the vPars.

From these two outputs we have the information we need to create the Virtual Partitions in the next step.

4) List the Components of the Virtual Partitions

From the **ioscan** and **dmesg** messages we can select the components of our first Virtual Partition. The following is a list of components we'll include in this partition:

First vPar *cable1*

```
name         cable1
processors   min of one (bound) max of three (two unbound)
             with num (bound + umnbound) equal to one
memory       1024 MB
LBA Core I/O 0/0 (all components on 0/0 are implied)
LAN          0/0/0/0 (not specified explicitly, on 0/0)
boot disk    0/0/1/1.2.0
kernel       /stand/vmunix (this is default)
autoboot     off (manual)
console      0/0/4/0 (not specified explicitly, on 0/0)
```

You may want to set *autoboot* to *auto* during installation and set to *manual* after installation. This makes booting easier during installation.

Some of the components require some explanation concerning the way in which they are implemented with vPars. The following is a more detailed discussion of some of these components, including CPU, memory, and LAN, bootdisk, setboot, kernel, and console.

CPU

The CPUs used in both this partition (*cable1*) and the one we will define shortly (*cable2*) are specified with *min, max*, and *num*. We will have *min bound* CPUs that have I/O interrupts assigned to them and are therefore ideal for I/O-intensive applications. The additional CPUs assigned to the vPars are *unbound*, and do not process I/O interrupts. Therefore, *unbound* CPUs are ideal for processor-intensive applications as opposed to I/O-intensive applications. *Unbound* CPUs can be freely moved from one vPar to another while vPars are running, so having *min* bound CPUs gives us the freedom to move around the *unbound* CPUs. *Bound* CPUs can also be added to and deleted from Virtual Partitions only when the partition is down.

On machines that employ Non-Uniform Memory Access (NUMA) you would use hardware paths (*path*) to specify CPUs. This is to ensure that you minimize the distance between CPUs and memory. On systems such as the rp7400 (formerly know as N-Class) and rp5400 (formerly know as L-Class) that do not employ NUMA, *min* is recommended to define *bound* CPUs.



For our work on *cable1* and *cable2* and for my work with vPars in general, the most common desire is to have a *min* number of *bound* CPUs in all vPars and then move around *unbound* CPUs as the applications in vPars need them. For instance, when **vparcreate** is run we would specify the following:

vparcreate -p cable1 -a cpu:::1:3 -a cpu::1

At the time of creation *cable1* will have one *bound* CPU only because we specified a *min* of one and a *num* of one. *num* is the total *bound* + *unbound* CPUs, and since



we specified one for *num*, we'll get one *bound* CPU (I have seen some vPars material use *num* and some use *total* so *num* and *total* are interchangable in this book.) Since *max* is three we have left the door open to add as many as two additional *unbound* CPUs. If we have two *unbound* CPUs on our system, we can move them among the vPars as required using **vparmodify**. To remove the two *unbound* CPUs from *cable2* and add them to *cable1*, we would issue the two following **vparmodify** commands:

vparmodify -p cable2 -m cpu::1 <-- reduces cable2 from 3 to 1

vparmodify -p cable1 -m cpu::3 <-- increases cable1 from 1 to 3

We first removed the two *unbound* CPUs from *cable2* and then added them to *cable1*. If the two *unbound* CPUs were not assigned to a vPar, we would not have to remove them from *cable2* prior to adding them to *cable1*.

There are many ways to work with CPUs, so by characterizing your applications and understanding the options for using *bound* and *unbound* CPUs, you can use the processor mix that best meets your needs.

Memory

We have identified one GByte of memory for *cable1*. Memory can be specified by *range* or *size*.

To add one GByte of memory to *cable1* using *size,* we would use the following **vparcreate** command:

```
vparcreate -p cable1 -a mem::1024
```



This **vparcreate** command specifies only the memory for use in *cable1*. The full **vparcreate** command for creating *cable1* will be shown in an upcoming section.

The memory is specified in MBytes (1024 MBytes = 1 GByte) in multiples of 64 MBytes. At the time of this writing, the Virtual Partition Monitor consumes roughly 128 MBytes of RAM, so this will not be available to allocate to a Virtual Partition. Modifying memory allocation requires that the Virtual Partition be down, at the time of this writing.

On machines that employ Non-Uniform Memory Access (NUMA) you would use the *range*. *Range* is a subset of *size.* On systems such as the rp7400 (formerly know as N-Class) and rp5400 (formerly know as L-Class) that do not employ NUMA, the *size* is recommended to define memory. The syntax for specifying memory by range is as follows:

```
mem::base:range
```

None of the examples in this book were prepared using NUMA systems so you won't see any examples using the *range* syntax; all examples use the *size* syntax.

LAN

The LAN interface used for this first Virtual Partition is on Local Bus Adaptper (LBA) zero. This means that any other components on LBA zero would have to be in this Virtual Partition as well. At the time of this writing, components on an LBA can't be shared between vPars.

Note that we have decided not to use the hostname as our Virtual Partition name. As mentioned earlier, it is desirable to use the same name for the hostname and vPar. Because our hostnames are a little hard to remember the system administrator decided to use simple vPar names. When we loaded HP-UX 11i on the system we selected the hostnames (you can also run **set_parms** after loading 11i to set the system name and other parameters) of *cvhdcon3* and *cvhdcon4*. We then chose the simple vPar names of *cable1* and *cable2*, respectively.

Boot Disk

The **ioscan** issued earlier in this chapter showed many disk devices. The boot device for our first Virtual Partition is the internal disk with the hardware path *0/0/1/1.2.0*.

At the time of this writing, components that are at or below the Local Bus Adapter (LBA) level are devoted to a single Virtual Partition. This means that although the output of our earlier **ioscan** command shows four

internal disks, all four of these disks must be in the same Virtual Partition because they are on the same LBA.

Kernel



We'll use the default HP-UX kernel of **/stand/vmunix** for the kernel in this Virtual Partition. Since we're using the default kernel, we don't have to specify this as part of the **vparcreate** command, however; we'll include it in the **vparcreate** command for completeness purposes.

setboot Command

In our example we have *autoboot* set to *off* for our Virtual Partition. The **setboot** command on a non-vPars system reads from and writes to stable storage. On a vPars system the **setboot** command interacts with the Virtual Partition database. In our upcoming example we'll set the *autoboot* to *off* when we create *cable1* with **vparcreate**. Running **setboot** on a vPars system has the effects shown in Table 16-2:

Table 16-2. setboot and Virtual Partitions

vPars setboot Option	Description
-a	Changes the alternate boot path of the Virtual Partition. To set the alternate boot path: `# setboot -a 0/8/0/0.8.0.5.0.0.0`
-b	Sets the autoboot attribute of the Virtual Partition. To set Autoboot *on*: `# setboot -b on`
-p	Changes the primary boot path of the Virtual Partition. To set the primary boot path: `# setboot -p 0/0/1/1.2.0`
-s	Has no effect.
no options	Displays information about boot attributes.

The **setboot** command is one of the aspects of working with vPars that is different from a non-vPars system.

Console

Chapter 5 of the vPars book contains detailed information on the way in which the console operates in a vPas enviornment. In our first partition we have specified LBA *0/0* as a component of vPar *cable1*. Since the physical console at *0/0/4/0* is on the Core I/O card at *0/0*, it is an implied component of *cable1* and we do not have to specify the physical console in this partition. The other Virtual Partitions on this system will use the virtual console functionality of vPars whereby issuing *Ctrl-A* cycles between virtual console displays.

Database

The Virtual Partition database that contains all vPar-related information is **/stand/vpdb**. This is managed, and synchronized for you, so you don't need to pay too much attention to it if you don't want to. You can, however; create an alternate database if you wish. You may want to do this in order to create a completely different Virtual Partition configuration for your system without affecting your currently running database.

> When creating Virtual Partitions with **vparcreate** you can use the *-D* option and specify an alternate database name that is a file in the **/stand** directory, such as **/stand/vpdb.app2**. When you boot vPars from this database (with `ISL> hpux`



`/stand/vpmon -D` *`db_file`*) it is the default, so all modifications made to vPars defined in this database are made to it rather than the default.

Second vPar *cable2*

We'll list the same categories of components for *cable2* as we did for *cable1* in the following list:

```
name          cable2
processors    min of one (bound) max of three (two unbound)
              with num (bound + umnbound) equal to one
memory        1024 MB
LAN           0/10/0/0
boot disk     0/8/0/0.8.0.5.0.0.0
kernel        /stand/vmunix (this is the default)
autoboot      off (manual)
console       virtual console to be created
```

We now have a list of components for two vPars. The result is that our rp5400 (formerly know as L-Class) system has been divided into two vPars that look like Figure 16-5:

Figure 16-5. rp5400 (formerly know as L-Class) with Two vPars (unused components not shown)

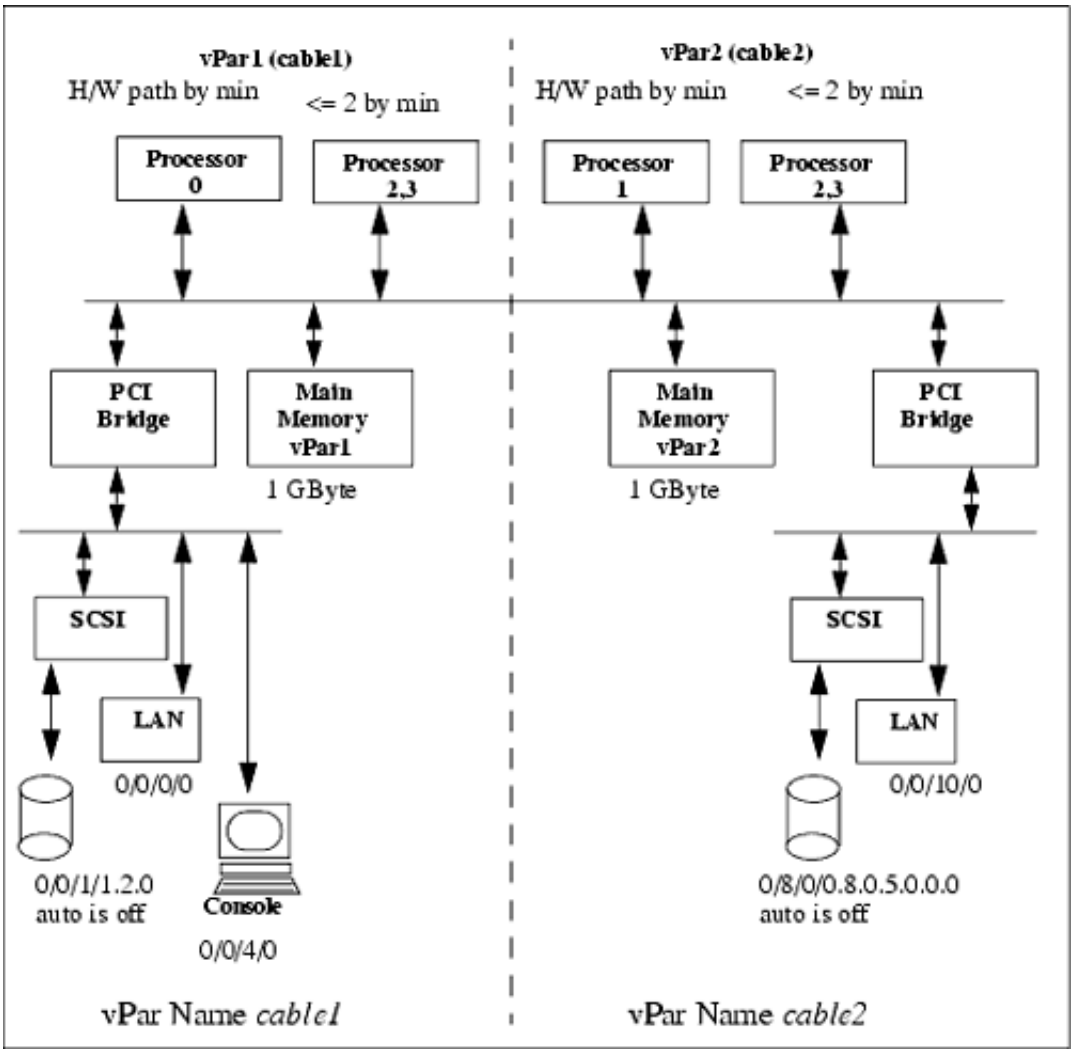


Figure 16-5 reflects what our system will look like when we perform the upcoming steps to create our two vPars. Note that two *unbound* processors, shown as *2,3* in Figure 16-5, can be assigned to *cable1* and *cable2* as required.

5) Virtual Partition Kernel-Related Work

Each Virtual Partition has its own instance of HP-UX 11i, which has its own HP-UX kernel. It is likely that you'll customize these kernels in a variety of ways to suit the applications you have running in the respective vPars. When you install the vPars software, it automatically reconfigures the kernel to include the vPar drivers and make the kernel relocatable. You do not have to perform the kernel-related steps in this section because they are performed for you when vPars software is loaded. It is still informative; however, to see the steps that were manually performed in this section to get better insight conconcerning the way that vPars operate. In this step we'll investigate the files that have been updated by the vPars application and build the new kernel. Keep in mind that the new kernel needs to be built on every volume that has HP-UX 11i on it and will run a vPar.

Because memory is shared among multiple vPars, the kernel must be relocatable in memory. At the time of this writing, there are patches that allow the kernel to be built as a relocatable kernel. We won't perform any checks related to patches.

The file **/sbin/vecheck** is a vPar file that is required on the system. The following listing is a portion of **/usr/conf/gen/config.sys** that checks to see if **/sbin/vecheck** has been loaded on the system:

```
# Determine whether the linker supports kernel relocation.  If it does,
# link the kernel using the relocation options.
LOADOPTS_ADDL=` \
```

```
        if [ -f /sbin/vecheck ]; then \
                ${WHAT} ${LD} | \
                ${AWK} '$$0 ~ /92453-07 linker/ { \
                        split($$7, vers, "."); \
                        if ( vers[1] == "B" && \
                        ( vers[2] == 11 && vers[3] >= 25 ) || vers[2] > 11 ) \
                                print "${LOADOPTS_RELOC}"; \
                        else print "${LOADOPTS_STATIC}"; }'; \
        else \
                echo "${LOADOPTS_STATIC}"; \
        fi; \
    `
```

The following is a long listing of **/sbin/vecheck** which was loaded with the vPar software:

```
# ll /sbin/vecheck
-r-xr-xr-x   1 bin        bin          20533 Mar  5 19:01 /sbin/vecheck
#
```

Next let's take a look at **/stand/system** to see the *vpar* driver that has been added to the file:

```
# cat /stand/system
*************************************
* Source: /ux/core/kern/filesets.info/CORE-KRN/generic
* @(#)B.11.11_LR
*
*************************************
* Additional drivers required in every machine-type to
* create a complete
* system file during cold install.  This list is every driver that the
* master.d/ files do not force on the system or is not identifiable by
* ioscan.
* Other CPU-type specific files can exist for their special cases.
* see create_sysfile (1m).
*************************************
*
* Drivers/Subsystems
sba
lba
c720
sctl
sdisk
asio0
cdfs
cxperf
olar_psm
olar_psm_if
dev_olar
diag0
diag1
diag2
dmem
dev_config
iomem
nfs_core
nfs_client
nfs_server
btlan
maclan
dlpi
token_arp
inet
uipc
tun
telm
tels
netdiag1
nms
hpstreams
clone
strlog
sad
echo
sc
timod
```

```
tirdwr
pipedev
pipemod
ffs
ldterm
ptem
pts
ptm
pckt
td
fddi4
gelan
GSCtoPCI
iop_drv
bs_osm
vxfs
vxportal
lvm
lv
nfsm
rpcmod
autofsc
cachefsc
cifs
prm
vpar                                <--- vpar driver added here
STRMSGSZ                65535
nstrpty                 60
dump lvol
maxswapchunks           2048
#
```

The *vpar* driver is a master driver described in **/usr/conf/master.d**, as shown below:

```
# pwd
/usr/conf/master.d
# cat vpar
$CDIO
vpar       0
$$$

$DRIVER_INSTALL
vcn             -1              209
vcs             -1              -1
vpar_driver     -1              -1
$$$

$DRIVER_DEPENDENCY
vcn             vpar
vcs             vpar
vpar            vcs vcn vpar_driver
vpar_driver     vpar
$$$

$DRIVER_LIBRARY
*
* The driver/library table.  This table defines which libraries a given
* driver depends on.  If the driver is included in the dfile, then the
* libraries that driver depends on will be included on the ld(1) command
* line.  Only optional libraries *need* to be specified in this table,
* (but required ones can be included, as well).
*
* Driver handle    <libraries>
*
* subsystems first
vcn             libvpar-pdk.a
vcs             libvpar-pdk.a
vpar            libvpar-pdk.a
vpar_driver     libvpar-pdk.a
$$$

$LIBRARY
*
* The library table.  Each element in the library table describes
* one unique library.  The flag member is a boolean value, it is
```

```
* initialized to 1 if the library should *always* be included on
* the ld(1) command line, or 0 if the library is optional (i.e. it
* is only included when one or more drivers require it).  The order
* of the library table determines the order of the libraries on the
* ld(1) command line, (i.e. defines an implicit load order).  New
* libraries must be added to this table.
* Note: libhp-ux.a must be the last entry, do not place
* anything after it.
*
* Library   <required>
*
libvpar-pdk.a     0
$$$
#
```

You can see in that this file there are multiple drivers present. The *vcn* and *vcs* drivers are used to support the console in a vPars environment. Since you'll probably only have one physical console for multiple partitions, you need a way to share the physical device. The use of these drivers is described in Chapter 4 in the vPars book, in which kernel configuration is covered. For now it is sufficient to know that these drivers exist as part of the vPars installation and must be built into the kernel.

Now that the kernel has what it needs to be built relocatable and the drivers are present for vPars, we can run **mk_kernel** to build the new kernel and **kmupdate** to move the new kernel-related files into place. This is done automatically for you, but the following commands show how you would perform this process:

```
# mk_kernel
Generating module: krm...
Compiling /stand/build/conf.c...
Loading the kernel...
Generating kernel symbol table...
# kmupdate

  Kernel update request is scheduled.

  Default kernel /stand/vmunix will be updated by
  newly built kernel /stand/build/vmunix_test
  at next system shutdown or startup time.

#
```

Keep in mind that this procedure needs to be performed for all HP-UX 11i operating systems that will run a Virtual Partition.

6) Create the First Virtual Partition



The **vparcreate** command is used to create a vPar. The summary of this command is shown in Table 16-1 and its man page appears in Appendix B. The general form of the command is as follows:

```
vparcreate -p vp_name [-B boot_attr] [-D db_file] [-S static_attr]
[-b kernel_path] [-o boot_opts] [-a rsrc] [-a...]
```

When creating this vPar, I placed the **vparcreate** command in a file so that I could modify it for the second vPar and execute it. The **vparcreate** command is shown below:

```
# cat /tmp/cable1
vparcreate -p cable1 -B manual -b /stand/vmunix -a cpu::1
-a cpu:::1:3 -a mem::1024 -a io:0/0 -a io:0/0/1/1.2.0:boot
#
```

After changing the permissions on this file and running it, the vPar *cable1* was successfully created. Next, we'll boot the vPar we just created.

7) Boot the First Virtual Partition

Now that the first vPar has been created and the kernel automatically rebuilt to support vPars, we can boot the first vPar which we named *cable1*.

We'll both boot off the first vPar and check its status. We need to load the Virtual Partition Monitor (**vpmon**) at the *ISL*> prompt. **vpmon** is a *ramdisk* kernel, similar to *vmunix*, that needs to be loaded at the time of boot. From the *ISL*> prompt we are going to run **vpmon** to get the *MON*> prompt. From the *MON*> prompt we boot our Virtual Partition with **vparload,** as shown in the following example:

```
ISL> hpux /stand/vpmon

Boot
: disk(0/0/1/1.2.0.0.0.0.0;0)/stand/vpmon
421888 + 142056 + 4247112 start 0x23000
cable1: WARNING: No boot device specified

Welcome to VPMON (type '?' for a list of commands)

MON> vparload -p cable1

[MON] Console client set to cable1

[MON] cable1 loaded

              -
              -
              -
```

You may see messages different from those shown in the example after the **vparload** command was issued. In any event, the system progressed through the remainder of the boot process and booted the one Virtual Partition *cable1* that we created. We now have a subset of the system components dedicated to this Virtual Partition.

The **vparload** command has the following three forms:

```
form1: vparload -all

form2: vparload -auto

form3: vparload -p vp_name [-b kernelpath] [-o boot_options]
[-B hardware_path]
```

We issued the third form shown above.

Now that the partition has booted, let's first obtain the status of the one Virtual Partition we created, called *cable1*, that we have running:




```
# vparstatus -p cable1 -v


[Virtual Partition Details]
Name:         cable1
State:        Up
Attributes:   Dynamic,Manual
Kernel Path:  /stand/vmunix
Boot Opts:

[CPU Details]
Min/Max:  1/3
Bound by User [Path]:
Bound by Monitor [Path]:  33
Unbound [Path]:

[IO Details]
   0.0
   0.0.1.1.2.0  BOOT
```

```
[Memory Details]
Specified [Base  /Range]:
          (bytes) (MB)
Total Memory (MB):  1024
#
```



The output of **vparstatus** shows that *cable1* is *up*. The *-v* option is used to obtain a verbose output. You can see from this listing that the bound CPU at hardware path *33* (the *bound* CPU we specified with the *min*) is part of the partition, that there is one GByte of memory in the partition, and that the I/O components we specifiied are in the partition. Had there been other partitions configured, we would have seen their output as well.



Note that the console at *0/0/4/0* is an implied component of this vPar. So, too, is the LAN interface at *0/0/0/0*. Both of these components are part of the Core I/O card that we specified as part of *cable1* with the *-a io:0/0* argument to the **vparcreate** command.

We can now run **vparstatus -A** to view the available components of our system. Since we created a first partition with only one CPU, we should see three CPUs and many other system components available, as shown in the following listing:

```
# vparstatus -A

[Unbound CPUs (path)]:  37
                        97
                        101
[Available CPUs]:  3

[Available I/O devices (path)]:  0.1
                                 0.2
                                 0.3
                                 0.4
                                 0.5
                                 0.8
                                 0.9
                                 0.10
                                 0.12
                                 32
                                 36
                                 96
                                 100

[Unbound memory (Base  /Range)]:  0x0/128
                (bytes) (MB)      0xc000000/1856
                                  0x180000000/1088
[Available memory (MB)]:  3072
#
```

This output shows many components available for our second partition. Based on our earlier planning exercise, we know the components that we wish to include in the second vPar, and this **vparstatus -A** command confirms that they are indeed available.

For *cable2* we want one CPU initially, and there are three available. We want the I/O cards for boot and LAN at *0/8* and *0/10* respectively We want one GByte of memory and there are now roughly three GBytes available. We have all of the components we need to proceed to our next step of creating *cable2*.

8) Create the Second Virtual Partition

We earlier listed all of the components of which our second partition is to be comprised and confirmed that these components are still available with the **vparstatus -A** command. HP-UX 11i has already been loaded on a second disk on the same system used to create our first Virtual Partition *cable1*. We can create our second Virtual Partition, which we'll call *cable2.* We'll create the second while the first is running and boot the second vPar from the first.



Here are the components we earlier listed for our second vPar:

```
name          cable2
processors    min of one (bound) max of three (two unbound)
              with num (bound + umnbound) equal to one
memory        1024 MB
LAN           0/10
LBA           0/8
boot disk     0/8/0/0.8.0.5.0.0.0
kernel        /stand/vmunix (this is the default)
autoboot      off (manual)
console       virtual console to be created
```



There are several differences between the list of components for the two vPars. We have devoted a different LAN card and boot disk. We have specified the CPUs in the same manner in both vPars with *min* (this will be *bound*) of one and *max* of three, *num* of one, and let the the vPars software identify the one *bound* processor. These two Virtual Partitions will use different I/O paths for their devices. Let's now run the **vparcreate** command to create *cable2*:

We can now proceed to create the second partition with the command shown in the following file:

```
# cat /tmp/cable2
vparcreate -p cable2 -B manual -b /stand/vmunix -a cpu::1
-a cpu:::1:3 -a mem::1024 -a io:0/8
-a io:0/8/0/0.8.0.5.0.0.0:boot -a io:0/10
#
```

It is important to obtain the path of the boot device exactly as it appears in the *search* at boot time. When you issue the **vparcreate** you use the path as it appears in the *search*; however, when you issue **vparstatus -v** you may see a path with additional *0*s added to the boot device.



After executing this file, we can determine if the second vPar has been created and the components of which it is comprised by running **vparstatus**:

```
# vparstatus -v

[Virtual Partition Details]
Name:         cable1
State:        Up
Attributes:   Dynamic,Manual
Kernel Path:  /stand/vmunix
Boot Opts:

[CPU Details]
Min/Max:  1/3
Bound by User [Path]:
Bound by Monitor [Path]:  33
Unbound [Path]:

[IO Details]
   0.0
   0.0.1.1.2.0  BOOT

[Memory Details]
Specified [Base  /Range]:
          (bytes) (MB)
Total Memory (MB):  1024
```

```
[Virtual Partition Details]
Name:         cable2
State:        Down
Attributes:   Dynamic,Manual
Kernel Path:  /stand/vmunix
Boot Opts:


[CPU Details]
Min/Max:  1/3
Bound by User [Path]:
Bound by Monitor [Path]:  37
Unbound [Path]:

[IO Details]
   0.8
   0.8.0.0.8.0.5.0.0.0, BOOT
   0.10

[Memory Details]
Specified [Base  /Range]:
          (bytes) (MB)
Total Memory (MB):  1024
#
```



This output shows that the first vPar is intact and that the second has been successfully created with the name, kernel file, CPU, I/O, and memory components we specified. Note that each vPar has one bound CPU assigned to it. The LAN card assigned to *cable2* appears in the output because we specifed LBA *0/10* as one of the components of *cable2*. The console at *0/0/4/0* and the LAN interface at *0/0/0/0* are implied components of *cable1* and do not appear in the **vparstatus -v** output.

With the second vPar created, we can proceed to the next step and boot it.

9) Boot the Second Virtual Partition



Since we already have the first vPar running, called *cable1*, and the second vPar created, called *cable2*, we can boot the second vPar from the first. There are many options to boot vPars. Since we already have the first vPar running, we'll simply boot the second from the first with **vparboot** and then run **vparstatus -v** as shown in the following example. If we type subsequent **vparstatus** commands we can see the status of vPar *cable2* progress from *Load*, to *Boot* in the next output, and finally, *Up* when the vPar is running, as shown in the following listing:

```
# vparboot -p cable2
vparboot: Booting cable2.  Please wait...

# vparstatus
[Virtual Partition]
                                                                  Boot
Virtual Partition Name         State Attributes Kernel Path       Opts
============================== ===== ========== ================ =====
cable1                         Up    Dyn,Manl   /stand/vmunix
cable2                         Load  Dyn,Manl   /stand/vmunix

[Virtual Partition Resource Summary]
                                        CPU    Num        Memory (MB)
                                 CPU    Bound/   IO   # Ranges/
Virtual Partition Name          Min/Max  Unbound  devs  Total MB    Total MB
==============================  =======  =======  ====  ==========  ========
cable1                            1/  3    1   0     4    0/  0         1024
cable2                            1/  3    1   0     4    0/  0         1024

# vparstatus
[Virtual Partition]
                                                                  Boot
Virtual Partition Name         State Attributes Kernel Path       Opts
============================== ===== ========== ================ =====
```

```
cable1                          Up    Dyn,Manl   /stand/vmunix
cable2                          Boot  Dyn,Manl   /stand/vmunix

[Virtual Partition Resource Summary]
                                          CPU    Num        Memory (MB)
                                  CPU     Bound/   IO   # Ranges/
Virtual Partition Name          Min/Max  Unbound  devs  Total MB    Total MB
==============================  =======  =======  ====  ==========  ==========
cable1                            1/  3    1   0     4    0/  0         1024
cable2                            1/  3    1   0     4    0/  0         1024

# vparstatus
[Virtual Partition]
                                                                        Boot
Virtual Partition Name          State Attributes Kernel Path            Opts
============================== ===== ========== ======================== =====
cable1                          Up    Dyn,Manl   /stand/vmunix
cable2                          Up    Dyn,Manl   /stand/vmunix

[Virtual Partition Resource Summary]
                                          CPU    Num        Memory (MB)
                                  CPU     Bound/   IO   # Ranges/
Virtual Partition Name          Min/Max  Unbound  devs  Total MB    Total MB
==============================  =======  =======  ====  ==========  ==========
cable1                            1/  3    1   0     4    0/  0         1024
cable2                            1/  3    1   0     4    0/  0         1024

#
```

This progression of states of *cable2* reflects the time it takes to boot the operating system from the second volume on which this vPar is run.

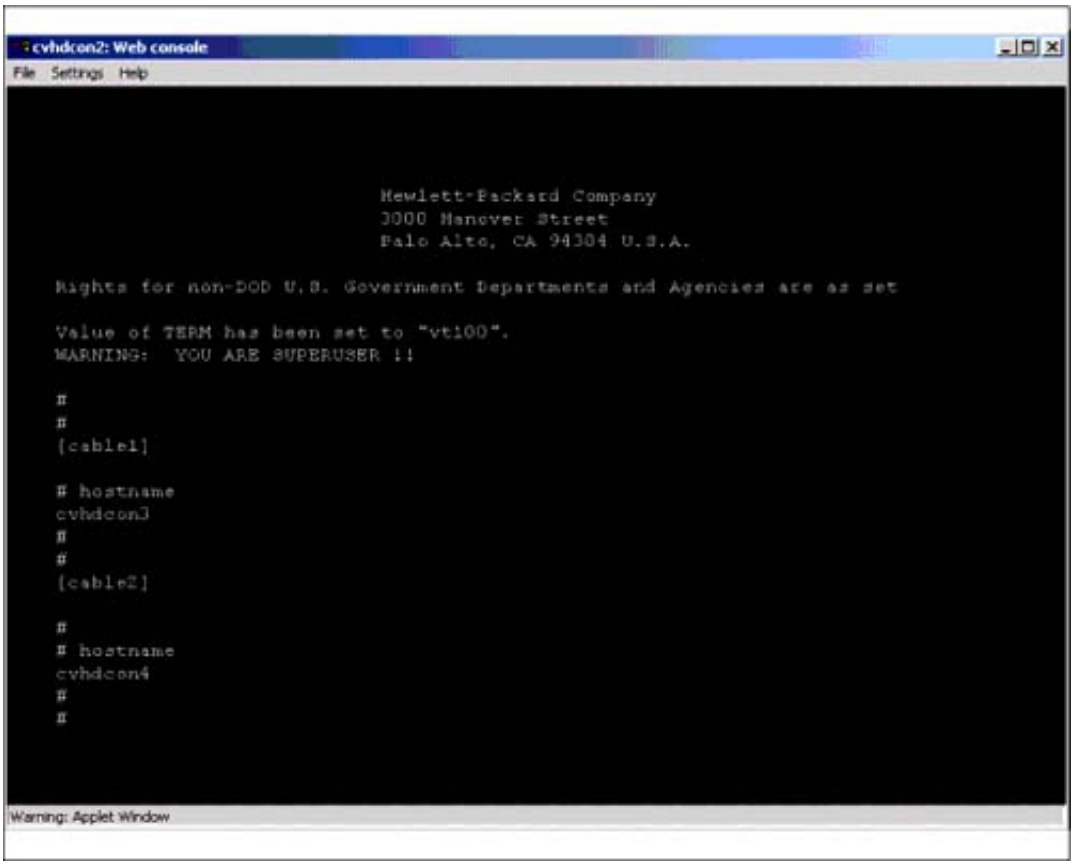
In addition to *load*, *boot*, and *up*, there are other states in which you may find a Virtual Partition as well. Table 16-3 summarizes the states of Virtual Partitions at the time of this writing:

Table 16-3. Virtual Partitions States

vPars State	Description
load	The kernel image of a Virtual Partition is being loaded into memory. This is done by the Virtual Partition monitor.
boot	The Virtual Partition is in the process of booting. The kernel image has been successfully loaded by the Virtual Partition monitor.
up	The Virtual Partition has been successfully booted and is running.
shut	The Virtual Partition is in the process of shutting down.
down	The Virtual Partition is not running and is down.
crash	The Virtual Partition has experienced a panic and is crashing.
hung	The Virtual Partition is not responding and is hung.

With more than one vPar running, you would use the built-in vPars drivers to toggle the console between any number of Virtual Partitions using *Ctrl-A.* Figure 16-6 shows using the console to view *cable1* with a hostname of *cvhdcon3.* Issuing *Ctrl-A* connects to vPar *cable2* with a hostname of *cvhdcon4*. When you issue *Ctrl-A* to switch to the next vPar in the console you are supplied with the name of the vPar to which you have connected in brackets, such as *[cable1]*.

Figure 16-6. Console Shown Using *Ctrl-A* to Toggle Between vPars



In addition to using the console to switch between vPars, you can also use the LAN cards configured into the respective vPars to open a *TELNET* or other type of session to the vPars. This is the same technique that you would use to connect to any system over the network and is one of the primary reasons you always want to have a LAN card configured as part of every vPar.

We did not cover the configuration of the two LAN cards, one in each vPar, in this chapter. The LAN configuration would have to be completed for both vPars in order to use the networking cards for such operations as a *TELNET* session. Chapter 13 of the vPars book covers many networking topics, including the **/etc/hosts** file; **/etc/rc.config.d/netconf** file, which must be configured on each vPar; and many others.

10) Modify the Virtual Partition

It is likely that you'll want to modify your Virtual Partitions in a variety of ways. You may want to add or remove a CPU, for instance. Let's take a look at an example of adding a CPU to a Virtual Partition.



In the upcoming example there is a four-processor system on which there are the two Virtual Partitions we just created: *cable1* and *cable2*. Each vPar has one *bound* CPU that was assigned by *min* when the vPars were created. Let's run **vparstatus** to see the components of which these two Virtual Partitions are comprised and confirm that each has one bound CPU:

```
# vparstatus -p cable1 -v

[Virtual Partition Details]
Name:         cable1
State:        Up
Attributes:   Dynamic,Manual
Kernel Path:  /stand/vmunix
Boot Opts:
```

```
[CPU Details]
Min/Max:  1/3
Bound by User [Path]:
Bound by Monitor [Path]:  33     <-- one bound CPU @ 33
Unbound [Path]:

[IO Details]
   0.0
   0.0.1.1.2.0  BOOT

[Memory Details]
Specified [Base  /Range]:
          (bytes) (MB)
Total Memory (MB):  1024
#


# vparstatus -p cable2 -v


[Virtual Partition Details]
Name:         cable2
State:        Up
Attributes:   Dynamic,Manual
Kernel Path:  /stand/vmunix
Boot Opts:

[CPU Details]
Min/Max:  1/3
Bound by User [Path]:
Bound by Monitor [Path]:  37    <-- one CPU in use at 37
Unbound [Path]:

[IO Details]
   0.8.0.0.8.0.5.0.0.0  BOOT
   0.10.0.0

[Memory Details]
Specified [Base  /Range]:
          (bytes) (MB)
Total Memory (MB):  1024
#
```



The output of these two **vparstatus** commands shows that *cable1* has one *bound* CPU and *cable2* has one *bound* CPU. On the rp5400 (formerly know as L-Class) system on which these vPars were created there are a total of four CPUs. This means that two CPUs should be available. Let's run **vparstatus -A** to view the available components on a system:

```
# vparstatus -A

[Unbound CPUs (path)]:  97
                        101
[Available CPUs]:  2

[Available I/O devices (path)]:  0.1
                                 0.2
                                 0.3
                                 0.4
                                 0.5
                                 0.9
                                 0.12
                                 32
                                 36
                                 96
                                 100

[Unbound memory (Base  /Range)]:  0x0/64
                (bytes) (MB)      0xc000000/1856
                                  0x180000000/128
[Available memory (MB)]:  2048
```

```
#
```



This output confirms that there are two CPUs available at hardware paths *97* and *101*. We can add these CPUs in a variety of ways. Let's use the **vparmodify** command to change the *num* of CPUs in *cable1* to two CPUs. We do this by *adding* one to the current number of CPUs with *-a*. This is a relative operation in that one CPU will be added to the current number of CPUs. You can use **vparmodify -m** if you want to specify the absolute number of CPUs for the vPar rather than the relative number. The following shows this **vparmodify** command:

```
# vparmodify -p cable1 -a cpu::1
#
```



We can now run **vparstatus -p cable1 -v** to confirm that the CPU has been added, shown in the following listing:

```
# vparstatus -p cable1 -v


[Virtual Partition Details]
Name:         cable1
State:        Up
Attributes:   Dynamic,Manual
Kernel Path:  /stand/vmunix
Boot Opts:

[CPU Details]
Min/Max:  1/3
Bound by User [Path]:
Bound by Monitor [Path]:  33          <-- original CPU @ 33
Unbound [Path]:  97                   <-- unbound CPU @ 97

[IO Details]
   0.0
   0.0.1.1.2.0  BOOT

[Memory Details]
Specified [Base  /Range]:
          (bytes) (MB)
Total Memory (MB):  1024
#
```

The **vparstatus** output shows that the CPU at hardware path *97* has indeed been added to *cable1* with the **vparmodify** command as *unbound*.

In addition, we can run GlancePlus or **top** to confirm that there are two CPUs in use on *cable2*. The following is a **top** output run on *cable2*:

```
# top

System: cvhdcon3                                    Thu Oct  4 15:30:42 2001
Load averages: 0.19, 0.51, 0.62
124 processes: 110 sleeping, 14 running
Cpu states:
CPU   LOAD   USER   NICE    SYS   IDLE  BLOCK  SWAIT   INTR   SSYS
 0    0.37   0.0%   0.2%   0.0%  99.8%   0.0%   0.0%   0.0%   0.0%
 1    0.02   0.0%   0.0%   0.8%  99.2%   0.0%   0.0%   0.0%   0.0%
---   ----  -----  -----  -----  -----  -----  -----  -----  -----
avg   0.19   0.0%   0.2%   0.4%  99.4%   0.0%   0.0%   0.0%   0.0%

Memory: 93636K (57816K) real, 322124K (239536K) virtual, 746284K free  Page# 1/4

CPU TTY        PID USERNAME PRI NI   SIZE    RES STATE    TIME %WCPU  %CPU COMMAND
 0   ?          36 root     152 20     0K   832K run      0:00  0.33  0.33 vxfsd
 1   ?        1342 root     158 10    80K   212K sleep    0:10  0.28  0.28 cclogd
 0   ?        1149 root     152 20  4644K  7260K run      0:06  0.21  0.21 prm3d
 1   ?         922 root     154 24   540K   808K sleep    0:00  0.15  0.15 hpterm
```

```
 0 pty/ttyp1  3114 root     186 24   596K   528K run      0:00  0.17  0.15 top
 1   ?        1146 root     -16 20  7788K  7240K run      0:03  0.14  0.13 midaemon
 1   ?           3 root     128 20     0K    32K sleep    0:04  0.11  0.11 statdaemon
 0   ?        2018 root     154 20  3908K  1908K sleep    0:00  0.05  0.04 alarmgen
 1   ?        1272 root     152 20   856K   960K run      0:00  0.04  0.04 opcmona
 1   ?        1372 root     152 20  1076K  2356K run      0:00  0.04  0.04 samd
 0   ?           0 root     128 20     0K     0K sleep    0:11  0.02  0.02 swapper
 1   ?           1 root     168 20   448K   204K sleep    0:00  0.02  0.02 init
 0   ?           2 root     128 20     0K    32K sleep    0:00  0.02  0.02 vhand
 0   ?           4 root     128 20     0K    32K sleep    0:00  0.02  0.02 unhashdaemo
 1   ?          20 root     147 20     0K    32K sleep    0:00  0.02  0.02 lvmkd
 0   ?          22 root     147 20     0K    32K sleep    0:00  0.02  0.02 lvmkd
 1   ?          24 root     147 20     0K    32K sleep    0:00  0.02  0.02 lvmkd
 0   ?         339 root     154 20   152K   204K sleep    0:00  0.02  0.02 syncer
 0   ?         342 root     168 20    76K   192K sleep    0:00  0.02  0.02 vphbd
 0   ?         345 root     168 20   156K   216K sleep    0:00  0.02  0.02 vpard
 0   ?         410 root     154 20    80K   224K sleep    0:00  0.02  0.02 syslogd
 0   ?         446 root     127 20   156K   424K sleep    0:00  0.02  0.02 netfmt
 0   ?         552 root     154 20   740K   816K sleep    0:00  0.02  0.02 rpc.statd
 0   ?         558 root     154 20  1004K  1032K sleep    0:00  0.02  0.02 rpc.lockd
 0   ?         586 root     154 20   180K   316K sleep    0:00  0.02  0.02 inetd
 0   ?         855 root     154 20  1064K   472K sleep    0:00  0.02  0.02 sendmail:
 0   ?         863 root     154 20   772K   712K sleep    0:00  0.02  0.02 snmpdm
 0   ?         896 root     154 20   620K   552K sleep    0:00  0.02  0.02 mib2agt
 0   ?         914 root     154 20  1332K   444K sleep    0:00  0.02  0.02 cmsnmpd
 1   ?         951 root     154 20  4044K  1840K sleep    0:00  0.02  0.02 rpcd
 1 pty/ttyp1   952 root     158 24   512K   180K sleep    0:00  0.02  0.02 sh
 0   ?         974 root     168 20   152K   304K sleep    0:04  0.02  0.02 scrdaemon
 0   ?         996 root     154 20   200K   336K sleep    0:00  0.02  0.02 pwgrd
 0   ?        1039 root     154 10   308K   428K sleep    0:00  0.02  0.02 diagmond
 0   ?        1093 root     154 20  1224K   816K sleep    0:00  0.02  0.02 ttd
 1   ?        1135 root     154 20  2588K  1624K sleep    0:00  0.02  0.02 perflbd
 0   ?        1156 root     154 20  2952K  1572K sleep    0:00  0.02  0.02 swagentd
 0   ?        1167 root     154 20   224K   252K sleep    0:00  0.02  0.02 emsagent
 0   ?        1168 root     127 20  2380K  2204K sleep    0:00  0.02  0.02 scopeux
```

This output shows two CPUs, labeled *0* and *1*, in *cable1*. The *System* name of *cvhdcon3* is shown at the output because the hostname for *cable1* is *cvhdcon3*.



Although this is a simple example showing how a Virtual Partition can be modified, it also demonstrates the power of vPars. While both vPars on the system are running, a processor can be added to one or both without interruption of the programs running in the vPars.



Note that the *-a* option to **vparmodify** changed the number of CPUs *relative* to the current number. In our case the current number of CPUs was one and using *-a cpu::1* added one CPU to the current number of one, resulting in two CPUs. This is true also when we use the *-d* option to **vparmodify** to remove processors. The following example shows running **vparstatus** to see the two CPUs, using **vparmodify** to change the number of CPUs back to one (this is also relative to the current number of CPUs, which is two) and a **vparstatus** to confirm that this change has taken place:

```
# vparstatus -p cable1 -v

# vparstatus -p cable1 -v

[Virtual Partition Details]
Name:         cable1
State:        Up
Attributes:   Dynamic,Manual
Kernel Path:  /stand/vmunix
Boot Opts:

[CPU Details]
Min/Max:  1/3
Bound by User [Path]:
Bound by Monitor [Path]:  33          <-- bound CPU @ 33
Unbound [Path]:  97                   <-- unbound CPU @ 97
```

```
[IO Details]
   0.0
   0.0.0.0
   0.0.1.1.2.0  BOOT
   0.0.4.0  CONSOLE

[Memory Details]
Specified [Base  /Range]:
          (bytes) (MB)
Total Memory (MB):  1024
# vparmodify -p cable1 -d cpu::1
# vparstatus -p cable1 -v

[Virtual Partition Details]
Name:         cable1
State:        Up
Attributes:   Dynamic,Manual
Kernel Path:  /stand/vmunix
Boot Opts:

[CPU Details]
Min/Max:  1/3
Bound by User [Path]:
Bound by Monitor [Path]:  33         <-- original CPU @ 33
Unbound [Path]:                      <-- no unbound CPUs

[IO Details]
   0.0
   0.0.0.0
   0.0.1.1.2.0  BOOT
   0.0.4.0  CONSOLE

[Memory Details]
Specified [Base  /Range]:
          (bytes) (MB)
Total Memory (MB):  1024
#
```

We could perform many other modifications to the vPars with the two *unbound* CPUs that are available, such as adding two CPUs to one of the vPars or one CPU to each vPar.



Please keep in mind the relative nature of components when using **vparmodify** and that some changes, such as modifying memory or adding I/O components, require the vPar to be down at the time of this writing.

Virtual Partition Dump Files

When a Virtual Partition crashes, a dump file is created in **/stand/vmpon.dmp**. When the Virtual Partition boots, files are created in **/var/adm/crash/vpar**. The files have an extension with a number indicating the number of the dump that occurred. For instance, **vpmon.1**, **vpmon.dmp.1**, and **summary.1** indicate the first set of files that are saved in **/var/adm/crash/vpar**.

An example of what you might see in **/stand** and **/var/adm/crash/vpar** related to dumps are shown in the following listing:

```
VPARNAME = extraq1

# ll /stand
total 100400
-rw-r--r--   1 root       sys            19 Jul 13 15:04 bootconf
drwxr-xr-x   4 root       sys          2048 Oct 18 11:43 build
drwxrwxrwx   5 root       sys          1024 Oct 18 13:06 dlkm
drwxrwxrwx   5 root       root         1024 Oct 18 11:21 dlkm.vmunix.prev
-rw-r--r--   1 root       sys          3388 Oct 18 13:16 ioconfig
-r--r--r--   1 root       sys            82 Jul 13 15:34 kernrel
drwxr-xr-x   2 root       sys          1024 Oct 18 13:18 krs
drwxr-xr-x   2 root       root         1024 Oct 18 13:16 krs_lkg
drwxr-xr-x   2 root       root         1024 Oct 18 13:18 krs_tmp
drwxr-xr-x   2 root       root         8192 Jul 13 15:04 lost+found
-rw-------   1 root       root           12 Oct 18 13:16 rootconf
-r--r--r--   1 root       sys          2035 Oct 18 11:42 system
-r--r--r--   1 root       sys           994 Jul 13 15:28 system.01
-r--r--r--   1 root       sys           999 Jul 13 15:56 system.02
-r--r--r--   1 root       sys           994 Jul 13 15:28 system.base
drwxr-xr-x   2 root       sys          1024 Jul 13 15:37 system.d
-r--r--r--   1 root       sys          2035 Oct 18 10:55 system.prev
-rwxr-xr-x   1 root       root      22682568 Oct 18 13:16 vmunix
-rwxr-xr-x   1 root       root      22682568 Oct 18 11:04 vmunix.prev
-rw-------   1 root       sys          8232 Oct 18 13:36 vpdb
-rw-------   1 root       root         8232 Jul 17 14:11 vpdb.OLD
-r-xr-xr-x   1 bin        bin        837616 Aug 31 18:59 vpmon
-rw-------   1 root       root      5078504 Oct 10 10:43 vpmon.dmp <-- vPar dump


# ll /var/adm/crash/vpar                                  <-- vPar dump directory
total 46464
-rw-r--r--   1 root       root           2 Oct 10 10:43 count
-rw-r--r--   1 root       root       16794 Jul 17 13:26 summary.0
-rw-r--r--   1 root       root       17953 Jul 18 10:35 summary.1
-rw-r--r--   1 root       root       19538 Jul 18 11:36 summary.2
-rw-r--r--   1 root       root       10012 Oct 10 10:43 summary.3
-r-xr-xr-x   1 root       root      855928 Jul 17 13:26 vpmon.0
-r-xr-xr-x   1 root       root      855928 Jul 18 10:35 vpmon.1
-r-xr-xr-x   1 root       root      855928 Jul 18 11:36 vpmon.2
-r-xr-xr-x   1 root       root      837616 Oct 10 10:43 vpmon.3
-rw-------   1 root       root     5078504 Jul 17 13:26 vpmon.dmp.0
-rw-------   1 root       root     5078504 Jul 18 10:35 vpmon.dmp.1
-rw-------   1 root       root     5078504 Jul 18 11:36 vpmon.dmp.2
-rw-------   1 root       root     5078504 Oct 10 10:43 vpmon.dmp.3
```

The **/var/adm/crash/vpar** directory has in it the vPar dump-related files for four (*0-3*) crashes.

The dump file created in **/stand** is saved in **/var/adm/crash/vpar** and extended with the crash number. The dump file in **/stand** is overwritten with each crash, but you have a history with all of the dump files and related information in **/var/adm/crash/vpar**. Please leave in place the **/stand/vpmon.dmp** file.

Summary





Not all of the vPars-related commands were issued in this chapter. The **vparreset** and **vparremove** commands summarized in Table 16-1 were not issued at all for instance. The **vparremove** command can be run on any vPar provided that it is in the *down* state. The general steps to get vPars up and running and to perform some modification were covered to give you a simple framework from which to work. There are some other commands that were not covered, for which there are manual pages in Appendix B as well. More detail in specific areas appears in other chapters of the vPars book and I encourage you to review the online man pages for all of the vPars in Appendix B.

There are also some considerations related to server technology that we did not cover. If you have Instant Capacity on Demand (iCOD) employed on your server, all CPUs must be activitated in order for vPars to work. When employing Processor Sets (psets) in a vPar, use only bound CPUs.

vparmgr GUI

There is always the age-old question when working with UNIX whether a task should be done at the command line or using a Graphical User Interface (GUI). The **vparmgr** GUI can perform many of the tasks described in this chapter.

Figure 16-7 shows the main Virtual Partition Manager page that the you first see when the application is started that provides status. It gives basic information about the number of vPars and the resources on the system. From this screen you can perform a variety of tasks, including creating new vPars, modifying existing vPars, booting vPars, and getting more detailed information about the system's vPar configuration.

Figure 16-7. Main vparmgr Screen

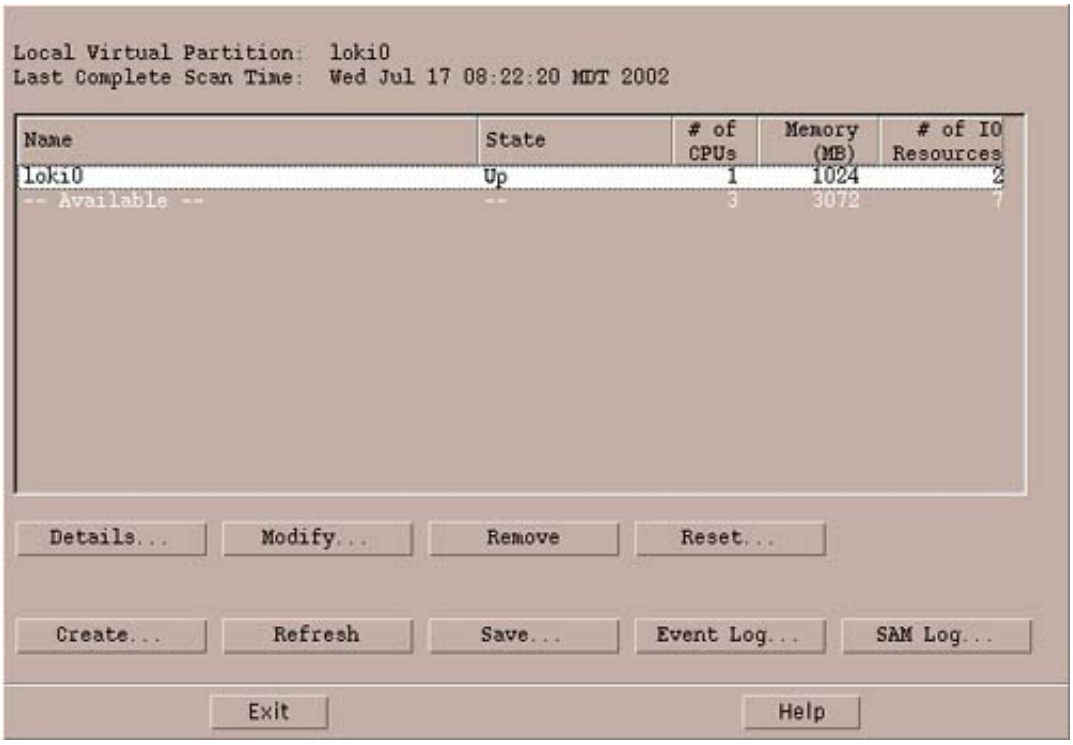


Figure 16-8 is the vPar modify screen that is displayed when a user selects an existing vPar and click *modify*. The user can then change the configuration of this vPar. The actions available through the modify screen are similar to what is available in the create wizard.

Figure 16-8. vparmgr *Modify* Screen

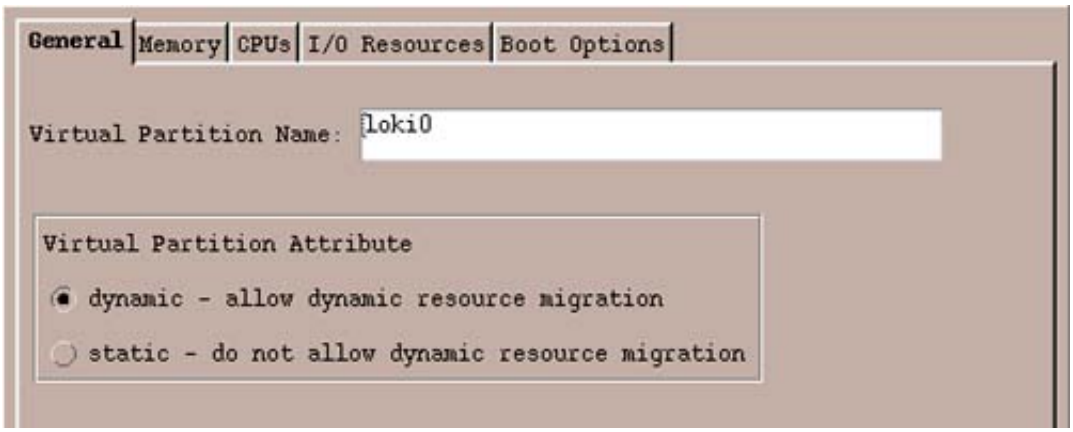
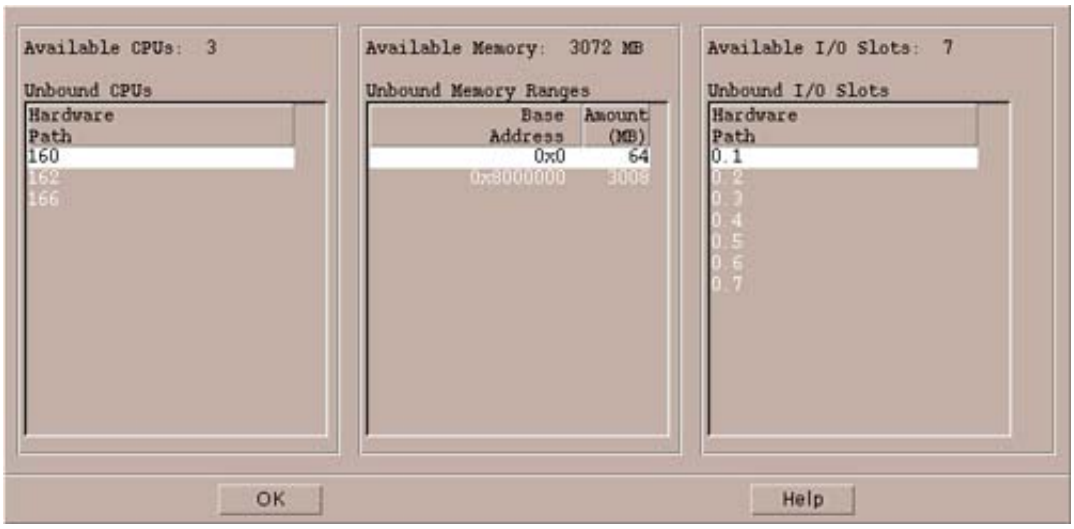


Figure 16-9 is the screen displayed when the user selects the available row on the status screen and clicks *details*. It shows more detailed information about the available resources on the system.

Figure 16-9. vparmgr *Available* Screen



With **vparmgr** you have the flexibility of using the graphical interface or command line when working with vPars.

[Team LiB]

PREVIOUS NEXT

Manual Pages for Commands Used in Chapter 16

There are manual pages in Appendix B for the following commands used in this chapter:

vparboot Boot (start) a virtual partition.

vparcreate Create a new virtual partition.

vparmodify Modify an existing virtual partition.

vparremove Remove (delete) an existing virtual partition.

vparreset Send a TOC or hard reset to a virtual partition.

vparresources: Provides description of Virtual Partitions and their resources.

vparstatus Display virtual partition and available resources information.

vpartition: Displays information about the vPars Command Line Interface



Chapter 17. nPartitions Overview

Introduction to nPartitions

Cells and nPartitions

Ways of Managing nPartitions

HP-UX nPartition Commands

Modifying and Creating an nPartition

Partition Manager (parmgr)

Management Processor nPartition Commands

The Boot Process Boot Console Handler (BCH) Commands



Introduction to nPartitions

Hard Partitions (nPartitions) have dedicated hardware within an HP 9000 that work independently from other nPartitions within the same server. This independence includes both hardware and software isolation from other nPartitions. Virtual Partitions (vPars) covered in an Chapter 16 have full software isolation from one another. nPartitions have both software *and* hardware isolation from one another. An nPartition could be divided into multiple vPars.

The basic building block of an nPartition is the *cell*. Each cell has on it processors and memory that are assigned exclusively to that nPartition. I/O chassis are also exclusively assigned to nPartitions. Multiple cells can be combined to produce large nPartitions.

nPartitions are supported on many HP 9000s and operate in a similar manner on all models. I'll use a few different models in the examples in this chapter; however, the operation of nPartitions varies very little going from model to model, so most of the commands and principles apply regardless of the model on which you're using nPartitions.

Although the operation of nPartitions is nearly identical going from model to model, the number of supported nPartitions does indeed vary based on the cell capacity of a given model. Some models support only two nPartitions and others support many more. I'll focus on working with nPartitions in this chapter so you can use the topics covered on any cell-based system that supports nPartitions.

This chapter provides an introduction to working with nPartitions. There is a lot involved in managing nPartitions, so this chapter covers some of the more common nPartition-related work you might perform. For a complete guide to managing nPartitions please print the document *HP System Partitions Guide: Administration for nPartitions*, available on docs.hp.com. This document covers a lot more detail than is found in this chapter. Although nPartitions are supported on a number of HP 9000 platforms, there are some additional aspects of managing a Superdome *complex* (the overall system on which nPartitions run) that are covered in detail in *Managing Superdome Complexes: A Guide for HP-UX System Administrators* also found on docs.hp.com. There are many fine documents on this Web site related to many topics covered in this book that can help you in your work.



Cells and nPartitions

Since the cell is the basic building block of nPartitions, let's spend a little time covering some of the basics of cells and nPartitions. An HP 9000 that is capable of supporting nPartitions is sometimes called a *server complex* or just *complex*. In this chapter I sometimes use *complex* and other times use system when referring to the overall system. The following list covers some important background you'll need to know in order to manage nPartitions.

Base Cells	All cells within an nPartition are base cells. When you create an nPartition, the cells to which youassignitarebasecells.
Assigned Cells	Assigned cells are included in an nPartition and unassigned cells are not included. Similarly, an I/O chassis that is attached to a cell takes on the assigned or unassigned attribute of the cell.
Unassigned Cells	Cells that are unassigned are available resources that can be used to create new nPartitions or assigned to existing nPartitions.
Core Cells	Every nPartition must have a cell that is attached to an I/O chassis that has a core I/O card in it. You can have multiple core cells, meaning cells to which an I/O chassis is attached with a core I/O card in it, but only one of these is the *active* core cell. If you have multiple core cells in an nPartition and the primary core cell fails, then you have a backup core cell. Not all HP 9000s allow you to have multiple core cells.
Active Cells	Cells that are both assigned to an nPartition and have booted are active cells. There is the concept of *rendezvous* on HP 9000s. During the boot process an nPartition rendezvous takes place when all of the available cells in an nPartition *join* together. Those that rendezvous are active cells.
Inactive Cells	These are cells that are not assigned to an nPartition or have not yet rendezvoused to form a partition. Some cells are in a boot-is-blocked (BIB) state, which means that they will not rendezvous and are inactive. There are a variety of reasons that a cell is in a BIB state including: it is in an *n* state for the next boot; it boots too late to rendezvous; the cell fails a self-test; the partition in which the cell exists is ready for reconfiguration.
Genesis nPartition	The Genesis nPartition is a one-cell nPartition used to create all other nPartitions. It can be expanded to include any number of cells but is initially only one cell. This is done for you by HP when your nPartition-capable system is delivered, if you have requested it. You may create the genesis partition at any time and reconfigure your server with a new nPartition structure. The genesis nPartition is number *0.* All cells in an nPartition must have the same clock speed and revision level.
Partition Numbers	Every nPartition has a unique number used by the commands and utilities to specify the nPartition.
Local Partition	When working with one of the commands or utilities, the nPartition you are accessing is local and all others are remote.

Now that we've covered some of the basics of nPartitions and cells, let's take a look at some of the tools and utilities used to manage them.

There are many supported nPartition configurations on all HP 9000s that support nPartitions. To see a complete list of the supported nPartition configurations see the aforementioned *HP System Partitions Guide: Administration for nPartitions* on docs.hp.com.



Ways of Managing nPartitions

nPartitions can be managed across a wide range of system operation. You can work with nPartitions at a very low level of the system such as the Guardian Service Processor (GSP,) or what I'll be calling the Management Processor (MP) in this chapter, or after HP-UX has booted with nPartitions commands or the menu driven Partition Manager (**/opt/parmgr/bin/parmgr**.) The following list summarizes the many options available to you for managing nPartitions, beginning with the lowest level tools and progressing to the highest level tools:

Management Processor	The Management Processor is a low level means of accessing the system. The Management Processor has its own unique set of commands, some of which we'll cover later. You gain access to the Service Process using telnet. because it has two network connections on it, one called a *Customer* LAN and the other called a *Private* LAN. You would typically use the *Customer* LAN to access the Management Processor. The Management Processor is also known as the Guardian Service Processor (GSP) and Service Processor so you may see these names used for the Management Processor, all of which represent the same method of accessing the system.
Virtual Front Panel	The Virtual Front Panel (VFP) shows activity of one or more nPartitions and is invoked through the Management Processor. Although this is a subset of the overall Management Processor functionality, it is widely used and is therefore considered its own tool.
HP-UX Commands	There are a variety of commands which you can issue that allow you to get useful information, and to configure and modify nPartitions from HP-UX. I'll provide examples of some of these commands and list all of them in an upcoming section.
Partition Manager	Partition Manager provides an interface (**parmgr**) through which you can interact with nPartitions. This interface can also be invoked from System Administration Manager (SAM.)
Boot-Related Commands	There are several ways of interacting with nPartitions at the time of boot including: Boot Control Handler (BCH), Initial System Loader (ISL), and other boot-related interfaces. We'll cover booting nPartitions in an upcoming section and describe these.

As usual, the best way to learn about a topic is to work through some examples, so let's do just that in the upcoming sections.

HP-UX nPartition Commands

Let's assume that you nPartition-based system is up and running and we want to issue some commands to see how it is configured. The first command we'll issue is **parstatus -P** to get some high-level information about the nPartitions in our system:

```
mtvnhp01:/>parstatus -P
[Partition]
Par              # of  # of I/O
Num Status       Cells Chassis  Core cell  Partition Name
=== ============ ===== ======== ========== ==============
 0  active       2     1      cab0,cell0 mtvnhp01
 1  active       1     1      cab0,cell1 mtvnhp02
mtvnhp01:/>
```

This output shows that we have two nPartitions on our rp8400. The first of the nPartitions has two cells and the second has one.

Next, we'll issue the **parstatus** command with a verbose output of nPartition *p0:*

```
# parstatus -V -p0
[Partition]
Partition Number       : 0
Partition Name         : P1
Status                 : active
IP address             : 0.0.0.0
Primary Boot Path      : 0/0/1/0/0.0.0
Alternate Boot Path    : 0/0/6/0/0.1.0
HA Alternate Boot Path : 0/0/1/0/0.6.0
PDC Revision           : 32.5
IODCH Version          : 5E70
CPU Speed              : 750 MHz
Core Cell              : cab0,cell0

[Cell]
                        CPU     Memory                                  Use
                        OK/     (GB)                          Core      On
Hardware   Actual       Deconf/ OK/                           Cell      Next Par
Location   Usage        Max     Deconf    Connected To        Capable   Boot Num
========== ============ ======= ========= ================== ======= ==== ===
cab0,cell0 active core  4/0/4    4.0/12.0 cab0,bay1,chassis3  yes     yes  0
cab0,cell1 active base  4/0/4    4.0/12.0 cab0,bay1,chassis1  yes     yes  0
cab0,cell2 active base  4/0/4    4.0/12.0 -                   no      yes  0
cab0,cell3 active base  4/0/4    4.0/12.0 -                   no      yes  0
cab0,cell4 active base  4/0/4    4.0/12.0 cab0,bay0,chassis1  yes     yes  0

[Chassis]
                                Core Connected  Par
Hardware Location   Usage       IO   To         Num
=================== =========== ==== ========== ===
cab0,bay1,chassis3  active      yes  cab0,cell0 0
cab0,bay1,chassis1  active      yes  cab0,cell1 0
cab0,bay0,chassis1  active      yes  cab0,cell4 0
```

This listing shows that we have one nPartition configured on the Superdome complex and that *cell0-4* are part of *p0*.

To determine the local nPartition on a system we'd issue **parstatus -w** as shown in the following output:

```
mtvnhp01:/>parstatus -w
The local partition number is 0.
mtvnhp01:/>
```

The next command we'll issue is **parstatus -C** to get details about the way in which our cells are configured, as shown in the following listing:

```
mtvnhp01:/>parstatus -C
[Cell]
                        CPU     Memory                                  Use
```

```
                        OK/     (GB)                            Core    On
Hardware   Actual       Deconf/ OK/                             Cell    Next Par
Location   Usage        Max     Deconf    Connected To          Capable Boot Num
========== ============ ======= ========= ==================== ======= ==== ===
cab0,cell0 active core  4/0/4    8.0/ 0.0 cab0,bay0,chassis0  yes     yes  0
cab0,cell1 active core  4/0/4    4.0/ 0.0 cab0,bay0,chassis1  yes     yes  1
cab0,cell2 active base  4/0/4    8.0/ 0.0 -                   no      yes  0
cab0,cell3 absent       -       -        -                   -       -    -

mtvnhp01:/>
```

This output shows that we have several cells in our system. The headings indicate that we are provided with a lot of information about the physical aspects of the cells, such as their location, hardware paths, and so on, as well as configuration information such as whether they are *active* or *inactive*.

Now let's issue **parstatus -I** to list all of the I/O chassis on our system:

```
mtvnhp01:/>parstatus -I
[Chassis]
                                Core Connected  Par
Hardware Location   Usage       IO   To         Num
=================== =========== ==== ========== ===
cab0,bay0,chassis0  active      yes  cab0,cell0 0
cab0,bay0,chassis1  active      yes  cab0,cell1 1

mtvnhp01:/>
```

This output lists everything related to I/O on our system.

The **parstatus** command without any options produces a more detailed list of results, as shown in the following output:

```
mtvnhp02:/>parstatus
Warning: No action specified. Default behaviour is display all.

[Complex]
   Complex Name :
   Complex Capacity
     Compute Cabinet (4 cell capable) : 1
   Active GSP Location : cabinet 0
   Model : 9000/800/S16K-A
   Serial Number : USR41520CK
   Current Product Number : A6093A
   Original Product Number : A6093A
   Complex Profile Revision : 1.0
   The total number of Partitions Present : 2

[Cabinet]
                 Cabinet   I/O       Bulk Power
                 Fans      Fans      Supplies
                 OK/       OK/       OK/
Cab              Failed/   Failed/   Failed/
Num Cabinet Type N Status  N Status  N Status     GSP
=== ============ ========= ========= ==========   ======
 0  S16K-A       21/ 0/ N+  6/ 0/ N+  6/ 0/ N+    active

Notes: N+ = There are one or more spare items (fans/power supplies).
       N  = The number of items meets but does not exceed the need.
       N- = There are insufficient items to meet the need.
       ?  = The adequacy of the cooling system/power supplies is unknown.

[Cell]
                        CPU     Memory                                  Use
                        OK/     (GB)                            Core    On
Hardware   Actual       Deconf/ OK/                             Cell    Next Par
Location   Usage        Max     Deconf    Connected To          Capable Boot Num
========== ============ ======= ========= ==================== ======= ==== ===
cab0,cell0 active core  4/0/4    8.0/ 0.0 cab0,bay0,chassis0  yes     yes  0
cab0,cell1 active core  4/0/4    4.0/ 0.0 cab0,bay0,chassis1  yes     yes  1
cab0,cell2 active base  4/0/4    8.0/ 0.0 -                   no      yes  0
cab0,cell3 absent       -       -        -                   -       -    -

[Chassis]
                                Core Connected  Par
Hardware Location   Usage       IO   To         Num
```

```
================== ============ ==== ========== ===
cab0,bay0,chassis0  active       yes  cab0,cell0 0
cab0,bay0,chassis1  active       yes  cab0,cell1 1

[Partition]
Par              # of  # of I/O
Num Status       Cells Chassis  Core cell  Partition Name (first 30 chars)
=== ============ ===== ======== ========== ================================
 0  active         2      1     cab0,cell0 mtvnhp01
 1  active         1      1     cab0,cell1 mtvnhp02
mtvnhp02:/>
```

This is a useful summary of the system showing the capacity of the system and which cell boards and chassis are devoted to which nPartitions.

Next let's take a look at detailed information for the local nPartition. We're interested in all of the components of our local nPartition as well as a detail of PCI I/O. The following listing shows the PCI detail with the **rad -q** command:

```
# rad -q
                                                                  Driver(s)
Slot         Path         Bus    Speed    Power    Occupied    Suspended   Capable
0-0-1-0      4/0/0        0      33       On       Yes         No          No
0-0-1-1      4/0/1/0      8      33       On       Yes         No          Yes
0-0-1-2      4/0/2/0      16     33       On       Yes         No          Yes
0-0-1-3      4/0/3/0      24     33       On       Yes         No          Yes
0-0-1-4      4/0/4/0      32     66       On       Yes         No          Yes
0-0-1-5      4/0/6/0      48     33       On       Yes         No          Yes
0-0-1-6      4/0/14/0     112    66       On       Yes         No          Yes
0-0-1-7      4/0/12/0     96     33       On       Yes         No          Yes
0-0-1-8      4/0/11/0     88     66       On       Yes         No          Yes
0-0-1-9      4/0/10/0     80     33       On       Yes         No          Yes
0-0-1-10     4/0/9/0      72     33       On       Yes         No          Yes
0-0-1-11     4/0/8/0      64     33       On       Yes         No          Yes
0-0-3-0      5/0/0        0      33       On       Yes         No          No
0-0-3-1      5/0/1/0      8      33       On       Yes         No          Yes
0-0-3-2      5/0/2/0      16     33       On       Yes         No          Yes
0-0-3-3      5/0/3/0      24     33       On       Yes         No          Yes
0-0-3-4      5/0/4/0      32     66       On       Yes         No          Yes
0-0-3-5      5/0/6/0      48     33       On       Yes         No          Yes
0-0-3-6      5/0/14/0     112    66       On       Yes         No          Yes
0-0-3-7      5/0/12/0     96     33       On       Yes         No          Yes
0-0-3-8      5/0/11/0     88     66       On       Yes         No          Yes
0-0-3-9      5/0/10/0     80     33       On       Yes         No          Yes
0-0-3-10     5/0/9/0      72     33       On       Yes         No          Yes
0-0-3-11     5/0/8/0      64     33       On       Yes         No          Yes
0-1-1-0      1/0/0        0      33       On       Yes         No          No
0-1-1-1      1/0/1/0      8      33       On       Yes         No          Yes
0-1-1-2      1/0/2/0      16     33       On       Yes         No          Yes
0-1-1-3      1/0/3/0      24     33       On       Yes         No          Yes
0-1-1-4      1/0/4/0      32     66       On       Yes         No          Yes
0-1-1-5      1/0/6/0      48     33       On       Yes         No          Yes
0-1-1-6      1/0/14/0     112    66       On       Yes         No          Yes
0-1-1-7      1/0/12/0     96     33       On       Yes         No          Yes
0-1-1-8      1/0/11/0     88     66       On       Yes         No          Yes
0-1-1-9      1/0/10/0     80     33       On       Yes         No          Yes
0-1-1-10     1/0/9/0      72     33       On       Yes         No          Yes
0-1-1-11     1/0/8/0      64     33       On       Yes         No          Yes
0-1-3-0      0/0/0        0      33       On       Yes         No          No
0-1-3-1      0/0/1/0      8      33       On       Yes         No          Yes
0-1-3-2      0/0/2/0      16     33       On       Yes         No          Yes
0-1-3-3      0/0/3/0      24     33       On       Yes         No          Yes
0-1-3-4      0/0/4/0      32     66       On       Yes         No          Yes
0-1-3-5      0/0/6/0      48     33       On       Yes         No          Yes
0-1-3-6      0/0/14/0     112    66       On       Yes         No          Yes
0-1-3-7      0/0/12/0     96     33       On       Yes         No          Yes
0-1-3-8      0/0/11/0     88     66       On       Yes         No          Yes
0-1-3-9      0/0/10/0     80     33       On       Yes         No          Yes
0-1-3-10     0/0/9/0      72     33       On       Yes         No          Yes
0-1-3-11     0/0/8/0      64     33       On       Yes         No          Yes
#
```

rad produces an output that gives us a lot of detail about the nPartitions in our system. You should run this command on your Superdome complex so that you can see all of the components of which your Superdome

is comprised. There is a lot of detail in this output that is not as easy to understand as the **parstatus** output, so let's take a closer look at a couple of these fields.

The first field is the slot information, which is in the following form:

```
Cabinet-Bay-Chassis-Slot    such as 0-0-1-0 for the first entry
```

The second field, which is the *Path*, contains the following

```
Cell/SBA/LBA/Device     such as 0/0/1/1 for the second entry
```

As with Virtual Partitions (vPars) covered in Chapter 16, you need to know something about the structure of your system in order to work with nPartitions. The System Bus Adapter (SBA) and Local Bus Adapters (LBA) are not components that you would typically worry about if you were not working with partitions. For the purpose of working through the information in this chapter, it is sufficient to know that there is a hierarchical I/O structure on HP 9000s in which the SBA exists at a higher level than the LBA and there are typically several LBAs per SBA. That is why for the second field, SBA of *0,* for instance, you'll see several third fields, which are many LBAs per SBA.

Next let's issue **ioscan -f** to see all of the components of our local nPartition number *0* in a two-nPartition rp8400:

```
mtvnhp01:/>ioscan -f
Class       I  H/W Path        Driver    S/W State   H/W Type     Description
============================================================================
root        0                   root      CLAIMED     BUS_NEXUS
cell        0  0                cell      CLAIMED     BUS_NEXUS
ioa         0  0/0              sba       CLAIMED     BUS_NEXUS    System Bus Ada
pter (804)
ba          0  0/0/0            lba       CLAIMED     BUS_NEXUS    Local PCI Bus
Adapter (782)
unknown    -1  0/0/0/0/0                 UNCLAIMED   UNKNOWN      PCI BaseSystem
 (103c128d)
tty         0  0/0/0/0/1        asio0     CLAIMED     INTERFACE    PCI Serial (10
3c1048)
lan         0  0/0/0/1/0        gelan     CLAIMED     INTERFACE    HP A3639-60019
 1000Base-T Built-in I/O
ext_bus     0  0/0/0/2/0        c720      CLAIMED     INTERFACE    SCSI C87x Ultr
a Wide Single-Ended
target      0  0/0/0/2/0.6      tgt       CLAIMED     DEVICE
disk        0  0/0/0/2/0.6.0    sdisk     CLAIMED     DEVICE       HP 36.4GST3367
06LC
target      1  0/0/0/2/0.7      tgt       CLAIMED     DEVICE
ctl         0  0/0/0/2/0.7.0    sctl      CLAIMED     DEVICE       Initiator
ext_bus     1  0/0/0/2/1        c720      CLAIMED     INTERFACE    SCSI C87x Fast
 Wide Single-Ended
target      2  0/0/0/2/1.2      tgt       CLAIMED     DEVICE
disk        1  0/0/0/2/1.2.0    sdisk     CLAIMED     DEVICE       HP      DVD-RO
M 305
target      3  0/0/0/2/1.7      tgt       CLAIMED     DEVICE
ctl         1  0/0/0/2/1.7.0    sctl      CLAIMED     DEVICE       Initiator
ext_bus     2  0/0/0/3/0        c720      CLAIMED     INTERFACE    SCSI C896 Fast
 Wide Single-Ended
target      4  0/0/0/3/0.6      tgt       CLAIMED     DEVICE
disk        9  0/0/0/3/0.6.0    sdisk     CLAIMED     DEVICE       HP 36.4GST3367
04LC
target      5  0/0/0/3/0.7      tgt       CLAIMED     DEVICE
ctl         2  0/0/0/3/0.7.0    sctl      CLAIMED     DEVICE       Initiator
ext_bus     3  0/0/0/3/1        c720      CLAIMED     INTERFACE    SCSI C896 Fast
 Wide LVD
target      6  0/0/0/3/1.7      tgt       CLAIMED     DEVICE
ctl         3  0/0/0/3/1.7.0    sctl      CLAIMED     DEVICE       Initiator
ba          1  0/0/1            lba       CLAIMED     BUS_NEXUS    Local PCI Bus
Adapter (782)
ba          2  0/0/2            lba       CLAIMED     BUS_NEXUS    Local PCI Bus
Adapter (782)
ba          3  0/0/4            lba       CLAIMED     BUS_NEXUS    Local PCI Bus
Adapter (782)
ba          4  0/0/6            lba       CLAIMED     BUS_NEXUS    Local PCI Bus
Adapter (782)
ba          5  0/0/8            lba       CLAIMED     BUS_NEXUS    Local PCI Bus
Adapter (782)
fc          0  0/0/8/0/0        td        CLAIMED     INTERFACE    HP Tachyon TL/
TS Fibre Channel Mass Storage Adapter
```

```
fcp          2  0/0/8/0/0.10     fcp       CLAIMED     INTERFACE    FCP Domain
ext_bus     20  0/0/8/0/0.10.30.0.0         fcparray  CLAIMED     INTERFACE    FCP
 Array Interface
target       7  0/0/8/0/0.10.30.0.0.0       tgt       CLAIMED     DEVICE
disk        12  0/0/8/0/0.10.30.0.0.0.0     sdisk     CLAIMED     DEVICE       HP
     A6188A
disk        13  0/0/8/0/0.10.30.0.0.0.1     sdisk     CLAIMED     DEVICE       HP
     A6188A
disk        16  0/0/8/0/0.10.30.0.0.0.2     sdisk     CLAIMED     DEVICE       HP
     A6188A
ext_bus     21  0/0/8/0/0.10.30.255.0       fcpdev    CLAIMED     INTERFACE    FCP
 Device Interface
target       8  0/0/8/0/0.10.30.255.0.0     tgt       CLAIMED     DEVICE
ctl         14  0/0/8/0/0.10.30.255.0.0.0   sctl      CLAIMED     DEVICE       HP
     A6188A
ext_bus     22  0/0/8/0/0.10.31.0.0         fcparray  CLAIMED     INTERFACE    FCP
 Array Interface
target       9  0/0/8/0/0.10.31.0.0.0       tgt       CLAIMED     DEVICE
disk        14  0/0/8/0/0.10.31.0.0.0.0     sdisk     CLAIMED     DEVICE       HP
     A6188A
disk        15  0/0/8/0/0.10.31.0.0.0.1     sdisk     CLAIMED     DEVICE       HP
     A6188A
disk        17  0/0/8/0/0.10.31.0.0.0.2     sdisk     CLAIMED     DEVICE       HP
     A6188A
ext_bus     23  0/0/8/0/0.10.31.255.0       fcpdev    CLAIMED     INTERFACE    FCP
 Device Interface
target      10  0/0/8/0/0.10.31.255.0.0     tgt       CLAIMED     DEVICE
ctl         15  0/0/8/0/0.10.31.255.0.0.0   sctl      CLAIMED     DEVICE       HP
     A6188A
ba           6  0/0/10           lba       CLAIMED   BUS_NEXUS   Local PCI Bus
Adapter (782)
ba           7  0/0/10/0/0       PCItoPCI  CLAIMED   BUS_NEXUS   PCItoPCI Bridg
e
lan          1  0/0/10/0/0/4/0   btlan     CLAIMED   INTERFACE   HP A5506B PCI
10/100Base-TX 4 Port
lan          2  0/0/10/0/0/5/0   btlan     CLAIMED   INTERFACE   HP A5506B PCI
10/100Base-TX 4 Port
lan          3  0/0/10/0/0/6/0   btlan     CLAIMED   INTERFACE   HP A5506B PCI
10/100Base-TX 4 Port
lan          4  0/0/10/0/0/7/0   btlan     CLAIMED   INTERFACE   HP A5506B PCI
10/100Base-TX 4 Port
ba           8  0/0/12           lba       CLAIMED   BUS_NEXUS   Local PCI Bus
Adapter (782)
ba           9  0/0/14           lba       CLAIMED   BUS_NEXUS   Local PCI Bus
Adapter (782)
memory       0  0/5              memory    CLAIMED   MEMORY      Memory
processor    0  0/10             processor CLAIMED   PROCESSOR   Processor
processor    1  0/11             processor CLAIMED   PROCESSOR   Processor
processor    2  0/12             processor CLAIMED   PROCESSOR   Processor
processor    3  0/13             processor CLAIMED   PROCESSOR   Processor
cell         1  2                cell      CLAIMED   BUS_NEXUS
memory       1 2/5              memory    CLAIMED   MEMORY      Memory
processor    4 2/10             processor CLAIMED   PROCESSOR   Processor
processor    5 2/11             processor CLAIMED   PROCESSOR   Processor
processor    6 2/12             processor CLAIMED   PROCESSOR   Processor
processor    7 2/13             processor CLAIMED   PROCESSOR   Processor
mtvnhp01:/>
```

This **ioscan** is context sensitive in that all of the components in nPartition *0* are shown in its output including a total of eight processors. Cell *0* and *2* are in this nPartition. We'll confirm this later. The following listing shows an **ioscan** output of the second nPartition in the same rp8400:

```
mtvnhp02:/>ioscan -f
Class        I  H/W Path         Driver    S/W State   H/W Type     Description
==========================================================================
root         0                   root      CLAIMED     BUS_NEXUS
cell         0  1                cell      CLAIMED     BUS_NEXUS
ioa          0  1/0              sba       CLAIMED     BUS_NEXUS    System Bus Ada
pter (804)
ba           0  1/0/0            lba       CLAIMED     BUS_NEXUS    Local PCI Bus
Adapter (782)
unknown     -1  1/0/0/0/0                  UNCLAIMED   UNKNOWN      PCI BaseSystem
 (103c128d)
tty          0  1/0/0/0/1        asio0     CLAIMED     INTERFACE    PCI Serial (10
3c1048)
```

```
lan         0  1/0/0/1/0       gelan     CLAIMED     INTERFACE    HP A3639-60019
 1000Base-T Built-in I/O
ext_bus     0  1/0/0/2/0       c720      CLAIMED     INTERFACE    SCSI C87x Ultr
a Wide Single-Ended
target      0  1/0/0/2/0.6     tgt       CLAIMED     DEVICE
disk        0  1/0/0/2/0.6.0   sdisk     CLAIMED     DEVICE       HP 36.4GST3367
06LC
target      1  1/0/0/2/0.7     tgt       CLAIMED     DEVICE
ctl         0  1/0/0/2/0.7.0   sctl      CLAIMED     DEVICE       Initiator
ext_bus     1  1/0/0/2/1       c720      CLAIMED     INTERFACE    SCSI C87x Ultr
a Wide Single-Ended
target      2  1/0/0/2/1.0     tgt       CLAIMED     DEVICE
tape        0  1/0/0/2/1.0.0   stape     CLAIMED     DEVICE       HP      C5683A
target      3  1/0/0/2/1.7     tgt       CLAIMED     DEVICE
ctl         1  1/0/0/2/1.7.0   sctl      CLAIMED     DEVICE       Initiator
ext_bus     2  1/0/0/3/0       c720      CLAIMED     INTERFACE    SCSI C896 Ultr
a2 Wide Single-Ended
target      4  1/0/0/3/0.6     tgt       CLAIMED     DEVICE
disk        1  1/0/0/3/0.6.0   sdisk     CLAIMED     DEVICE       HP 36.4GST3367
04LC
target      5  1/0/0/3/0.7     tgt       CLAIMED     DEVICE
ctl         2  1/0/0/3/0.7.0   sctl      CLAIMED     DEVICE       Initiator
ext_bus     3  1/0/0/3/1       c720      CLAIMED     INTERFACE    SCSI C896 Ultr
a2 Wide LVD
target      6  1/0/0/3/1.7     tgt       CLAIMED     DEVICE
ctl         3  1/0/0/3/1.7.0   sctl      CLAIMED     DEVICE       Initiator
ba          1  1/0/1           lba       CLAIMED     BUS_NEXUS    Local PCI Bus
Adapter (782)
ba          2  1/0/2           lba       CLAIMED     BUS_NEXUS    Local PCI Bus
Adapter (782)
ba          3  1/0/4           lba       CLAIMED     BUS_NEXUS    Local PCI Bus
Adapter (782)
ba          4  1/0/6           lba       CLAIMED     BUS_NEXUS    Local PCI Bus
Adapter (782)
ba          5  1/0/8           lba       CLAIMED     BUS_NEXUS    Local PCI Bus
Adapter (782)
fc          0  1/0/8/0/0       td        CLAIMED     INTERFACE    HP Tachyon TL/
TS Fibre Channel Mass Storage Adapter
fcp         0  1/0/8/0/0.10    fcp       CLAIMED     INTERFACE    FCP Domain
ext_bus     4  1/0/8/0/0.10.30.0.0        fcparray  CLAIMED     INTERFACE    FCP
 Array Interface
target      7  1/0/8/0/0.10.30.0.0.0      tgt       CLAIMED     DEVICE
disk        2  1/0/8/0/0.10.30.0.0.0.0    sdisk     CLAIMED     DEVICE       HP
     A6188A
disk        3  1/0/8/0/0.10.30.0.0.0.1    sdisk     CLAIMED     DEVICE       HP
     A6188A
disk        4  1/0/8/0/0.10.30.0.0.0.2    sdisk     CLAIMED     DEVICE       HP
     A6188A
ext_bus     5  1/0/8/0/0.10.30.255.0      fcpdev    CLAIMED     INTERFACE    FCP
 Device Interface
target      8  1/0/8/0/0.10.30.255.0.0    tgt       CLAIMED     DEVICE
ctl         4  1/0/8/0/0.10.30.255.0.0.0  sctl      CLAIMED     DEVICE       HP
     A6188A
ext_bus     6  1/0/8/0/0.10.31.0.0        fcparray  CLAIMED     INTERFACE    FCP
 Array Interface
target      9  1/0/8/0/0.10.31.0.0.0      tgt       CLAIMED     DEVICE
disk        5  1/0/8/0/0.10.31.0.0.0.0    sdisk     CLAIMED     DEVICE       HP
     A6188A
disk        6  1/0/8/0/0.10.31.0.0.0.1    sdisk     CLAIMED     DEVICE       HP
     A6188A
disk        7  1/0/8/0/0.10.31.0.0.0.2    sdisk     CLAIMED     DEVICE       HP
     A6188A
ext_bus     7  1/0/8/0/0.10.31.255.0      fcpdev    CLAIMED     INTERFACE    FCP
 Device Interface
target     10  1/0/8/0/0.10.31.255.0.0    tgt      CLAIMED     DEVICE
ctl         5  1/0/8/0/0.10.31.255.0.0.0  sctl     CLAIMED     DEVICE       HP
     A6188A
ba          6  1/0/10          lba       CLAIMED    BUS_NEXUS    Local PCI Bus
Adapter (782)
ba          7  1/0/10/0/0      PCItoPCI  CLAIMED    BUS_NEXUS    PCItoPCI Bridg
e
lan         1  1/0/10/0/0/4/0  btlan     CLAIMED    INTERFACE    HP A5506B PCI
10/100Base-TX 4 Port
lan         2  1/0/10/0/0/5/0  btlan     CLAIMED    INTERFACE    HP A5506B PCI
10/100Base-TX 4 Port
```

```
lan         3  1/0/10/0/0/6/0  btlan     CLAIMED    INTERFACE    HP A5506B PCI
10/100Base-TX 4 Port
lan         4  1/0/10/0/0/7/0  btlan     CLAIMED    INTERFACE    HP A5506B PCI
10/100Base-TX 4 Port
ba          8  1/0/12          lba       CLAIMED    BUS_NEXUS    Local PCI Bus
Adapter (782)
ba          9  1/0/14          lba       CLAIMED    BUS_NEXUS    Local PCI Bus
Adapter (782)
memory      0  1/5             memory    CLAIMED    MEMORY       Memory
processor   0  1/10            processor CLAIMED    PROCESSOR    Processor
processor   1  1/11            processor CLAIMED    PROCESSOR    Processor
processor   2  1/12            processor CLAIMED    PROCESSOR    Processor
processor   3  1/13            processor CLAIMED    PROCESSOR    Processor
mtvnhp02:/>
```

This **ioscan** output includes all components in nPartition *0* in this rp8400. Cell *1* isshowninthis **ioscan** output which we'll confirm later.

The **ioscan** outputs produced a lot of components for our local partition. The form of the **ioscan** output for an nPartition looks like the following:

```
    Field 1         Field 2        3     Field 4    Field 5  Field 6

Global cell no./proc, mem, or SBA/LBA/Card address/Function/dev addr
```

The six fields in the **ioscan** provide you with a lot of useful information about the components in your local nPartition. The LBA number is not the same as the PCI card number in many cases. Be careful when configuring I/O on any HP 9000 to check the PCI card slot vs. the LBA number.

This **ioscan** output shows the global cell number in the first field. We know that this system has the capacity of four cell boards. Let's issue the **parstatus** command using the *-c* option and specify all four cell boards:

```
mtvnhp02:/>parstatus -c0
[Cell]
                        CPU     Memory                              Use
                        OK/     (GB)                        Core    On
Hardware   Actual       Deconf/ OK/                         Cell    Next Par
Location   Usage        Max     Deconf    Connected To      Capable Boot Num
========== ============ ======= ========= ================= ======= ==== ===
cab0,cell0 active core  4/0/4    8.0/ 0.0 cab0,bay0,chassis0 yes     yes  0

mtvnhp02:/>parstatus -c1
[Cell]
                        CPU     Memory                              Use
                        OK/     (GB)                        Core    On
Hardware   Actual       Deconf/ OK/                         Cell    Next Par
Location   Usage        Max     Deconf    Connected To      Capable Boot Num
========== ============ ======= ========= ================= ======= ==== ===
cab0,cell1 active core  4/0/4   4.0/ 0.0  cab0,bay0,chassis1 yes     yes  1

mtvnhp02:/>parstatus -c2
[Cell]
                        CPU     Memory                              Use
                        OK/     (GB)                        Core    On
Hardware   Actual       Deconf/ OK/                         Cell    Next Par
Location   Usage        Max     Deconf    Connected To      Capable Boot Num
========== ============ ======= ========= ================= ======= ==== ===
cab0,cell2 active base  4/0/4    8.0/ 0.0 -                 no      yes  0

mtvnhp02:/>parstatus -c3
[Cell]
                        CPU     Memory                              Use
                        OK/     (GB)                        Core    On
Hardware   Actual       Deconf/ OK/                         Cell    Next Par
Location   Usage        Max     Deconf    Connected To      Capable Boot Num
========== ============ ======= ========= ================= ======= ==== ===
cab0,cell3 absent       -       -         -                 -       -    -

mtvnhp02:/>
```

This output shows that cells *0* and *2* are part of nPartition *0* and that cell *1* is in nPartition *1.* This jives with the **ioscan** output we produced earlier. Cell *3* is *absent,* meaning that the cell board is not now in the system.

The commands we've issued in this section have produced a lot of useful information about the way in which our nPartitions are configured. There are also many commands that you can issue to create, modify, remove, and manipulate nPartitions in other ways. Table 17-1 summarizes some of the commonly used nPartition-related commands:

Table 17-1. Commonly Used nPartition-Related Commands

Command	Description
ioscan	Scan system hardware and produce results. This command is context sensitive in that it produces output based on the *local* nPartition.
rad	Perform online addition and replacement-related tasks. Provides a detailed hardware summary when issued with no options.
parmgr	Graphical interface for configuring and managing nPartitions.
parcreate	Create a new nPartition.
parmodify	Modify an existing nPartition.
parremove	Remove an existing nPartition.
parstatus	Show nPartition information.
parunlock	Unlock profile data.
fruled	Blink attention indicators.
frupower	Read status and control power for cells and I/O chassis.

[Team LiB]

PREVIOUS NEXT

Modifying and Creating an nPartition

Now that we know the basics of some HP-UX commands for working with nPartitions, let's put a few of these commands to use by modifying and creating some nPartitions. We're kind of getting ahead of ourselves in this section. There are a lot of basics related to working with nPartitions in upcoming sections, but most of the commands used in this section are at the HP-UX command line or **parmgr** interface, so we won't be using any information in upcoming sections. Let's go ahead and perform some modification on an nPartition and create a new nPartition.

We'll start our work on a 32-way Superdome that has one nPartition. In the following listing we'll issue **parstatus** with the *-P* option to see what partitions exist, of which there is only one on our Superdome, then issue **parstatus** with no options to get an overall rundown on the partitions, and finally, run **parstatus -I** to see the I/O that exists in the Superdome complex:

```
# parstatus -P
[Partition]
Par              # of  # of I/O
Num Status       Cells Chassis  Core cell  Partition Name (first 30 chars)
=== ============ ===== ======== ========== ==============================
 0  active         6      3     cab0,cell0 P1

# parstatus
[Partition]
Partition Number       : 0
Partition Name         : P1
Status                 : active
IP address             : 0.0.0.0
Primary Boot Path      : 0/0/1/0/0.0.0
Alternate Boot Path    : 0/0/6/0/0.1.0
HA Alternate Boot Path : 0/0/1/0/0.6.0
PDC Revision           : 32.5
IODCH Version          : 5E70
CPU Speed              : 750 MHz
Core Cell              : cab0,cell0

[Cell]
                        CPU     Memory                                 Use
                        OK/     (GB)                          Core     On
Hardware   Actual       Deconf/ OK/                           Cell     Next Par
Location   Usage        Max     Deconf    Connected To        Capable  Boot Num
========== ============ ======= ========= ================== ======= ==== ===
cab0,cell0 active core  4/0/4    4.0/12.0 cab0,bay1,chassis3  yes     yes  0
cab0,cell1 active base  4/0/4    4.0/12.0 cab0,bay1,chassis1  yes     yes  0
cab0,cell2 active base  4/0/4    4.0/12.0 -                   no      yes  0
cab0,cell3 active base  4/0/4    4.0/12.0 -                   no      yes  0
cab0,cell4 active base  4/0/4    4.0/12.0 cab0,bay0,chassis1  yes     yes  0
cab0,cell7 active base  4/0/4    4.0/12.0 -                   no      yes  0

[Chassis]
                                Core Connected  Par
Hardware Location   Usage       IO   To         Num
=================== =========== ==== ========== ===
cab0,bay1,chassis3  active      yes  cab0,cell0 0
cab0,bay1,chassis1  active      yes  cab0,cell1 0
cab0,bay0,chassis1  active      yes  cab0,cell4 0

# parstatus -I
[Chassis]
                                Core Connected  Par
Hardware Location   Usage       IO   To         Num
=================== =========== ==== ========== ===
cab0,bay0,chassis0  absent      -    -          -
cab0,bay0,chassis1  active      yes  cab0,cell4 0
cab0,bay0,chassis2  absent      -    -          -
cab0,bay0,chassis3  active      yes  cab0,cell5 0
cab0,bay1,chassis0  absent      -    -          -
cab0,bay1,chassis1  active      yes  cab0,cell1 0
cab0,bay1,chassis2  absent      -    -          -
```

```
cab0,bay1,chassis3  active       yes  cab0,cell0 0
```

Our one nPartition, with a name of *P1* and a number of *0*, has in it cell boards *cell0-4* and *cell7. cell0* is the *core* cell in nPartition *p0*.

Next we'll remove *cell7* from this nPartition with **parmodify**. Notice in the following listing that after we run **parmodify** on *p0*, **shutdown -R** is required to make the removal of *cell7* from *p0*:

```
# parmodify -p0 -d7 -B
Cell 7 is active.
Use shutdown -R to shutdown the system to ready for reconfig state.
Command succeeded.
#
```

After having issued the **parmodify** and **shutdown -R**, *cell7* is indeed removed from *p0*. This is confirmed on reboot with the following verbose listing of *p0*:

```
# parstatus -V -p0
[Partition]
Partition Number       : 0
Partition Name         : P1
Status                 : active
IP address             : 0.0.0.0
Primary Boot Path      : 0/0/1/0/0.0.0
Alternate Boot Path    : 0/0/6/0/0.1.0
HA Alternate Boot Path : 0/0/1/0/0.6.0
PDC Revision           : 32.5
IODCH Version          : 5E70
CPU Speed              : 750 MHz
Core Cell              : cab0,cell0

[Cell]
                       CPU     Memory                                 Use
                       OK/     (GB)                          Core     On
Hardware   Actual       Deconf/ OK/                           Cell     Next Par
Location   Usage        Max     Deconf    Connected To        Capable  Boot Num
========== ============ ======= ========= ================== ======= ==== ===
cab0,cell0 active core  4/0/4    4.0/12.0 cab0,bay1,chassis3  yes     yes  0
cab0,cell1 active base  4/0/4    4.0/12.0 cab0,bay1,chassis1  yes     yes  0
cab0,cell2 active base  4/0/4    4.0/12.0 -                   no      yes  0
cab0,cell3 active base  4/0/4    4.0/12.0 -                   no      yes  0
cab0,cell4 active base  4/0/4    4.0/12.0 cab0,bay0,chassis1  yes     yes  0

[Chassis]
                              Core Connected  Par
Hardware Location   Usage     IO   To         Num
=================== ========= ==== ========== ===
cab0,bay1,chassis3  active    yes  cab0,cell0 0
cab0,bay1,chassis1  active    yes  cab0,cell1 0
cab0,bay0,chassis1  active    yes  cab0,cell4 0
```

This listing shows that *cell0-4* are part of *p0* and that *cell7* has been successfully removed from *p0*. Let's now take a look at the list of available cell components in our Superdome complex with **parstatus -A -C**:

```
# parstatus -A -C
[Cell]
                       CPU     Memory                                 Use
                       OK/     (GB)                          Core     On
Hardware   Actual       Deconf/ OK/                           Cell     Next Par
Location   Usage        Max     Deconf    Connected To        Capable  Boot Num
========== ============ ======= ========= ================== ======= ==== ===
cab0,cell5 inactive     4/0/4   4.0/12.0  cab0,bay0,chassis3  yes     -    -
cab0,cell6 inactive     4/0/4   4.0/12.0 -                   no      -    -
cab0,cell7 inactive     4/0/4   4.0/12.0 -                   no      -    -
```

This listing shows that *cell5-7* are free. We just removed *cell7* from nPartition *p0,* which resulted in it being shown in the available cell board listing.

Next let's create a new nPartition. We're going to add many components to the nPartition, so we'll first figure out what disks are used and available since one will be required as a boot device. The following listing of **strings /etc/lvmtab** shows only one disk in use:

```
# strings /etc/lvmtab
/dev/vg00
```

```
S*,%=A
/dev/dsk/c0t0d0
```

Only *c0t0d0* is in use at this time. We know that we can't use this disk, and we can see all disks on the system with the following **ioscan** command:

```
# ioscan -funC disk
Class     I  H/W Path       Driver S/W State    H/W Type     Description
===================================================================
disk      0  0/0/1/0/0.0.0  sdisk CLAIMED      DEVICE       SEAGATE ST173404LC
                           /dev/dsk/c0t0d0    /dev/rdsk/c0t0d0
disk      1  0/0/1/0/0.1.0  sdisk CLAIMED      DEVICE       SEAGATE ST173404LC
                           /dev/dsk/c0t1d0    /dev/rdsk/c0t1d0
disk      2  0/0/1/0/0.2.0  sdisk CLAIMED      DEVICE       SEAGATE ST173404LC
                           /dev/dsk/c0t2d0    /dev/rdsk/c0t2d0
disk      3  0/0/1/0/0.3.0  sdisk CLAIMED      DEVICE       SEAGATE ST173404LC
                           /dev/dsk/c0t3d0    /dev/rdsk/c0t3d0
disk      4  0/0/6/0/0.1.0  sdisk CLAIMED      DEVICE       HP      DVD-ROM 305
                           /dev/dsk/c1t1d0    /dev/rdsk/c1t1d0
disk      8  1/0/1/0/0.0.0  sdisk CLAIMED      DEVICE       SEAGATE ST173404LC
                           /dev/dsk/c4t0d0    /dev/rdsk/c4t0d0
disk      9  1/0/1/0/0.1.0  sdisk CLAIMED      DEVICE       SEAGATE ST173404LC
                           /dev/dsk/c4t1d0    /dev/rdsk/c4t1d0
disk     10  1/0/1/0/0.2.0  sdisk CLAIMED      DEVICE       SEAGATE ST173404LC
                           /dev/dsk/c4t2d0    /dev/rdsk/c4t2d0
disk     11  1/0/1/0/0.3.0  sdisk CLAIMED      DEVICE       SEAGATE ST173404LC
                           /dev/dsk/c4t3d0    /dev/rdsk/c4t3d0
disk     15  4/0/1/0/0.0.0  sdisk CLAIMED      DEVICE       SEAGATE ST173404LC
                           /dev/dsk/c8t0d0    /dev/rdsk/c8t0d0
disk     16  4/0/1/0/0.1.0  sdisk CLAIMED      DEVICE       SEAGATE ST173404LC
                           /dev/dsk/c8t1d0    /dev/rdsk/c8t1d0
disk     17  4/0/1/0/0.2.0  sdisk CLAIMED      DEVICE       SEAGATE ST173404LC
                           /dev/dsk/c8t2d0    /dev/rdsk/c8t2d0
disk     18  4/0/1/0/0.3.0  sdisk CLAIMED      DEVICE       SEAGATE ST173404LC
                           /dev/dsk/c8t3d0    /dev/rdsk/c8t3d0
disk     19  4/0/6/0/0.1.0  sdisk CLAIMED      DEVICE       HP      DVD-ROM 305
                           /dev/dsk/c9t1d0    /dev/rdsk/c9t1d0
```

Many disks are connected to the Superdome complex that we could use as a boot device. We'll create a new nPartition, called *sddev*,using *cell5-6* and a boot device of the second disk in our earlier **ioscan** listing. The following **parcreate** commands create a new nPartition:

```
# parcreate -P sddev -c5:base:y:ri -c6:base:y:ri -b 0/0/1/0/0.1
Partition Created. The partition number is: 1
#
```

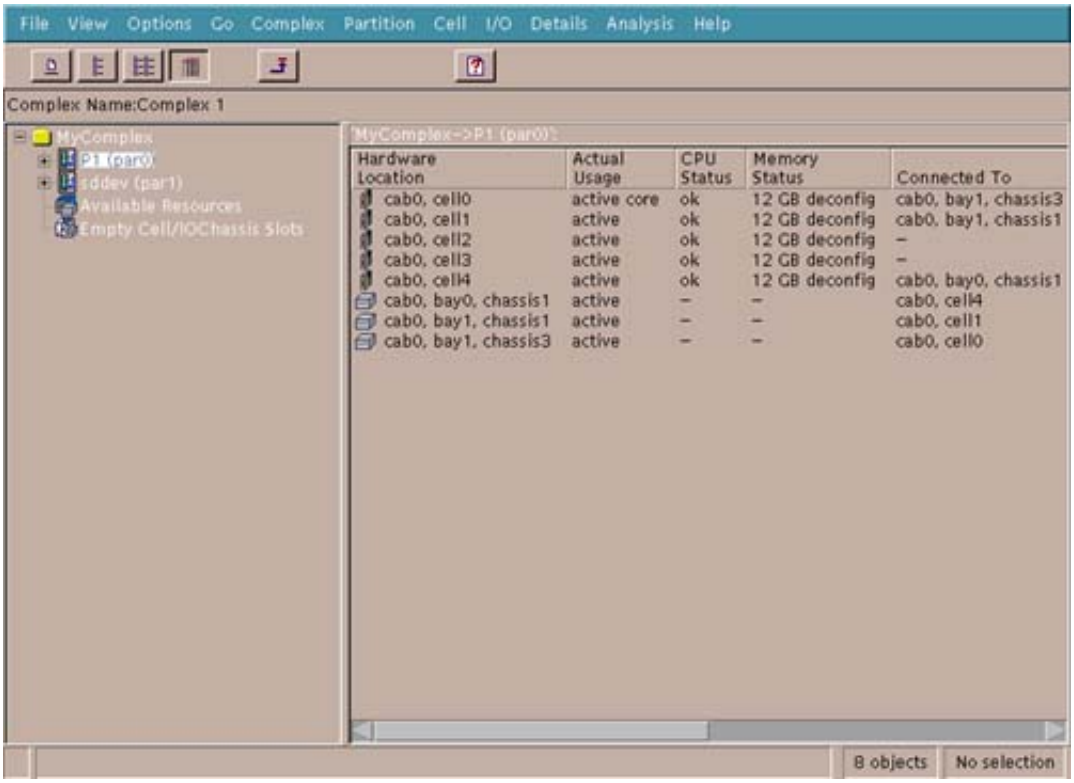
The two cell boards are added with *base* (*base* cell is the only valid type at this time,) *y* (which means that the cell will participate in the reboot), and *ri* (which is *reactive with interleave* which is the only valid value at this time). We left *cell7* unused by not including it in *sddev*.

The second partition has now been created. All of our work has been done at the command line. *cell7* was removed from the first partition and we created a second partition.

Partition Manager (parmgr)

As an alternative to performing all of our work at the command line we could have used the graphical tool **parmgr. parmgr**, with a full path of **/opt/parmgr/bin/parmgr**, can be used to perform most nPartition-related functions. Figure 17-1 shows the main **parmgr** screen:

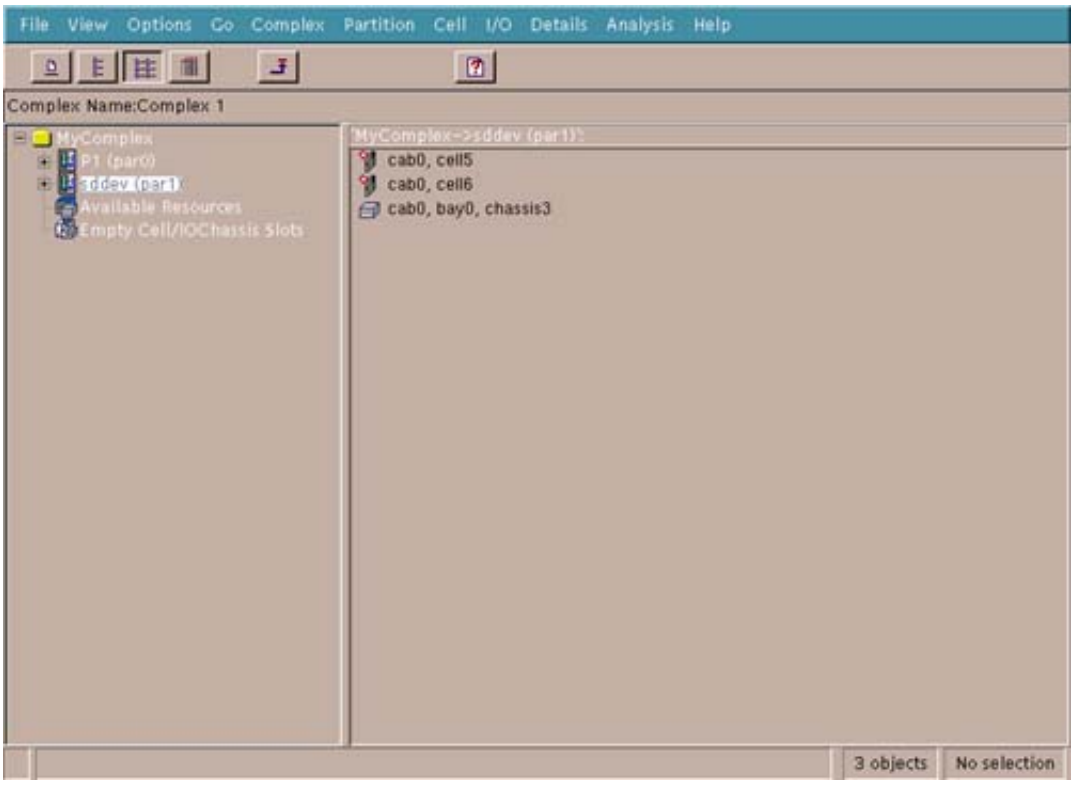
Figure 17-1. parmgr Main Screen



The left side of the **parmgr** screen shows the first nPartition (*P1*), the second nPartition (*sddev*), *Available Resources*,and *Empty* components. When one of these is selected, its details are shown in the right window. *P1* is selected in the left window so we see its components in the right window. Note that only *cell0-4* are part of *P1* because we removed *cell7* from it earlier at the command line.

Next let's select *sddev* in the left window which will show its details in the right window as shown in Figure 17-2:

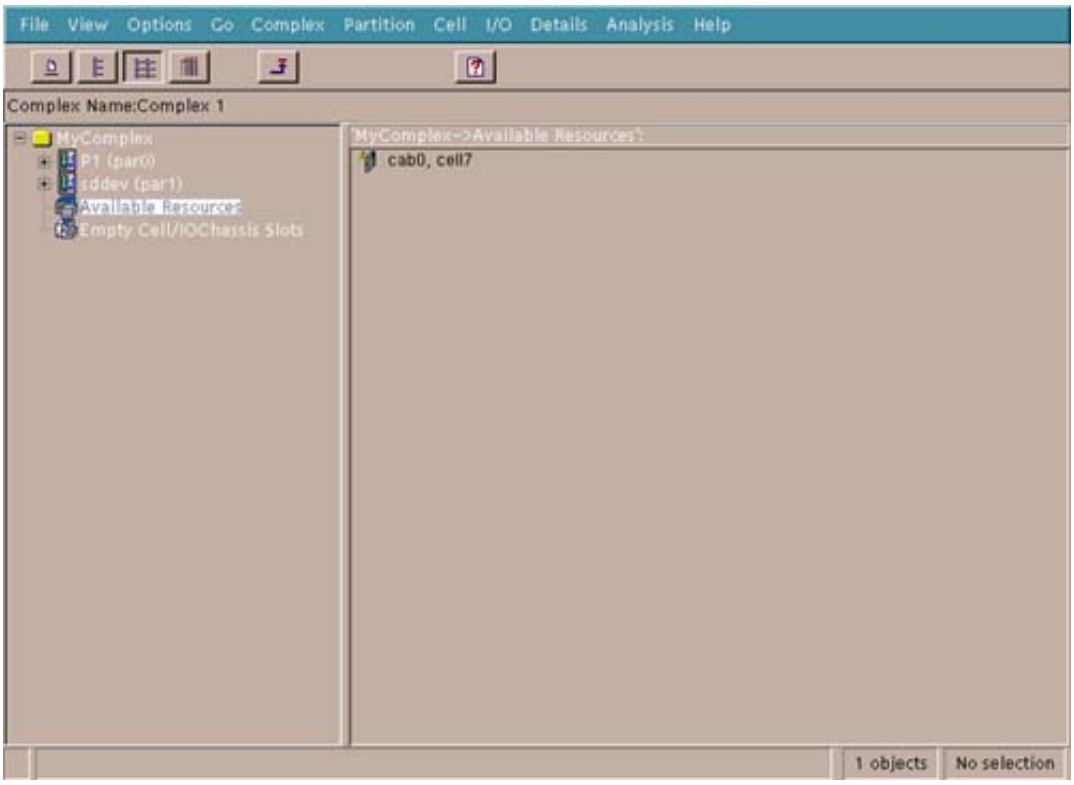
Figure 17-2. parmgr Highlighting *sddev*



You can see that *cell5-6* are part of this nPartition that we had earlier created at the command line.

Next, let's select *Available Resources* in the left window, which will display available components in the right window as shown in Figure 17-3:

Figure 17-3. parmgr Showing *cell7* Available



We could easily modify *sddev* to include *cell7* in the *Modify Partition* window shown in Figure 17-4:

Figure 17-4. parmgr Adding *cell7* to *sddev*



You can see in Figure 17-4 that we could add *cell7* to this nPartition or remove one of the existing cells from *sddev*. All of the work we performed at the command line earlier in this section could be easily performed with **parmgr**. As with most UNIX functions, I first wanted to show the command line means of performing some functions and then the graphical means of doing so. **parmgr** is so easy to use compared to the command line, most users prefer it over the command line.

The next section covers the Management Processor, which helps perform low-level functions. This is very important to know, because on a big system such as a Superdome complex, there are many levels and interfaces to work with that are covered in the next section.

[Team LiB]

PREVIOUS NEXT

Management Processor nPartition Commands

The Management Processor (also known as the Service Processor or Guardian Service Processor (GSP) in some cases) is a low level means of accessing the system. The Management Processor has its own unique set of commands, some of which we'll cover later. You gain access to the Service Process using telnet. because it has two network connections on it, one called a *Customer* LAN and the other called a *Private* LAN. You would typically use the *Customer* LAN to access the Management Processor. You could also access the Management Processor through the *Local Console* connection on the Core I/O card if you wish to plug in a terminal directly to the console port.

The Management Processor has the commands shown in Table 17-2:

Table 17-2. Management Processor Menus

Menu	Description

CO: Consoles	Lists the nPartitions available such as:
	
	`0) Partition 0`
	`1) Partition 1`

VFP: Virtual Front Panel	Access to all partitions including *System*.
CM: Command Menu	Access to many Management Processor commands.
CL: Console Log Menu	Provides a way of viewing nPartition logs.
SL: Chassis Log Menu	Provides a way of viewing chassis logs.
HE: Help	Help for using Management Processor.
X: Exit Connection	Exits Management Processor

Let's now run some examples of connecting to the Management Processor and running some commands. The following listing shows connecting to the Management Processor and the top-level menu presented after login. The connection to the Management Processor in this example is via a direct connection from the terminal to the *Local Console* onaCoreI/O board on an rp8400. The username and password for *MP* are both *Admin* in the following example:

```
MP login: Admin
MP password:


                        Welcome to the

              S Class 16K-A Management Processor

 (c) Copyright 1995-2001 Hewlett-Packard Co., All Rights Reserved.

                          Version 1.14


    MP MAIN MENU:

         CO: Consoles
        VFP: Virtual Front Panel
         CM: Command Menu
         CL: Console Logs
         SL: Show chassis Logs
         HE: Help
          X: Exit Connection

MP>
```

Next we'll select **CO** to view the nPartitions to which we can open a console. We'll select the first of the nPartitions with **0** andlogintothenPartition and see its hostname. After we logout of the nPartition we use **^b** (control b) to get back to the Management Processor.

```
MP> co


    Partitions available:

     #   Name
    ---  ----
     0)  mtvnhp01
     1)  mtvnhp02
     Q)  Quit

    Please select partition number: 0


        Connecting to Console: mtvnhp01

        (Use ^B to return to main menu.)

        [A few lines of context from the console log:]

- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -

Value of TERM has been set to "vt200".
WARNING: YOU ARE SUPERUSER !!

- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - --

GenericSysName [HP Release B.11.11] (see /etc/issue)
Console Login: root
Password:

Please wait...checking for disk quotas
(c)Copyright 1983-2000 Hewlett-Packard Co., All Rights Reserved.
(c)Copyright 1979, 1980, 1983, 1985-1993 The Regents of the Univ. of
California
(c)Copyright 1980, 1984, 1986 Novell, Inc.
(c)Copyright 1986-1992 Sun Microsystems, Inc.
(c)Copyright 1985, 1986, 1988 Massachusetts Institute of Technology
(c)Copyright 1989-1993 The Open Software Foundation, Inc.
(c)Copyright 1986 Digital Equipment Corp.
(c)Copyright 1990 Motorola, Inc.
(c)Copyright 1990, 1991, 1992 Cornell University
(c)Copyright 1989-1991 The University of Maryland
(c)Copyright 1988 Carnegie Mellon University
(c)Copyright 1991-2000 Mentat Inc.
(c)Copyright 1996 Morning Star Technologies, Inc.
(c)Copyright 1996 Progressive Systems, Inc.
(c)Copyright 1991-2000 Isogon Corporation, All Rights Reserved.

                          RESTRICTED RIGHTS LEGEND
Use, duplication, or disclosure by the U.S. Government is subject to
restrictions as set forth in sub-paragraph (c)(1)(ii) of the Rights in
Technical Data and Computer Software clause in DFARS 252.227-7013.

                          Hewlett-Packard Company
                          3000 Hanover Street
                          Palo Alto, CA 94304 U.S.A.

Rights for non-DOD U.S. Government Departments and Agencies are as set
forth in FAR 52.227-19(c)(1,2).
You have mail.

Value of TERM has been set to "vt200".
WARNING:  YOU ARE SUPERUSER !!

mtvnhp01:/>hostname
mtvnhp01
mtvnhp01:/>exit
logout root

GenericSysName [HP Release B.11.11] (see /etc/issue)
```

```
Console Login:

    MP MAIN MENU:

         CO: Consoles
        VFP: Virtual Front Panel
         CM: Command Menu
         CL: Console Logs
         SL: Show chassis Logs
         HE: Help
          X: Exit Connection

MP>
```

In the previous listing we chose to connect to the first nPartition; however, we could connect to any of the nPartitions through the Management Processor. We issued a **^b** immediately after logout from the first nPartition in order to get back to the Management Processor.

When we get back our *MP>* prompt, indicating that we are back at the Management Processor we issue **VFP** and **0** to see the virtual front panel of the first nPartition and **1** to see the virtual front panel of the second nPartition:

```
MP> VFP

    Partition VFP's available:

     #   Name
    ---  ----
     0)  mtvnhp01
     1)  mtvnhp02
     S)  System (all chassis codes)
     Q)  Quit

MP:VFP> 0



E indicates error since last boot
     Partition 0  state           Activity
     ------------------           --------
    HPUX heartbeat: *

  #  Cell state                    Activity
  -  ----------                    --------
  0 Cell has joined partition
  2 Cell has joined partition

MP:VFP (^B to Quit) >

    MP MAIN MENU:

         CO: Consoles
        VFP: Virtual Front Panel
         CM: Command Menu
         CL: Console Logs
         SL: Show chassis Logs
         HE: Help
          X: Exit Connection

MP> VFP

    Partition VFP's available:

     #   Name
    ---  ----
     0)  mtvnhp01
     1)  mtvnhp02
     S)  System (all chassis codes)
     Q)  Quit

MP:VFP> 1

E indicates error since last boot
     Partition 1 state            Activity
```

```
    ------------------            --------
    HPUX heartbeat:

  #  Cell state                   Activity
  -  ----------                   --------
  1 Cell has joined partition

MP:VFP (^B to Quit) >

    MP MAIN MENU:

         CO: Consoles
        VFP: Virtual Front Panel
         CM: Command Menu
         CL: Console Logs
         SL: Show chassis Logs
         HE: Help
          X: Exit Connection

MP>
```

Note that the first nPartition has two cell boards and the second has only one cell board. We are back at the main Management Processor prompt *MP>* and can now move to the *CM* menu. We'll issue an *he* for help and an *ls* to list information about the Management Processor LAN interface (*Customer LAN,* as opposed to *Private LAN*), then **sysrev** toseesomesystem revision-related information, and then issue **ma** to return the main Management Processor menu. Note that this is the only area of the Management Processor menu that requires an **ma** to get back to the main menu, as opposed to a **q** or **^b**.

```
   MP MAIN MENU:

        CO: Consoles
       VFP: Virtual Front Panel
        CM: Command Menu
        CL: Console Logs
        SL: Show chassis Logs
        HE: Help
         X: Exit Connection

MP> cm


              Enter HE to get a list of available commands


MP:CM> he


Management Processor Firmware revision 1.14, Nov 2 2001 at 17:21:31


The following are the Service commands that are available

BO      : Boot a partition.
DF      : Display FRU information of an Entity.
MA      : Return to the Main menu.
MR      : Modem Reset
PE      : Power entities on or off.
RE      : Reset entity.
RR      : Reset a partition for reconfiguration.
RS      : Reset a partition.
SYSREV  : Returns all System Revisions
TC      : Send a TOC signal to a partition.
TE     : Broadcast a message to all users of the MP Command Handler.
WHO     : Display a list of MP connected users

The following are the Status commands that are available

CP      : Display partition cell assignments
HE      : Display the list of available commands
LS      : Display LAN connected console status
MS      : Display the status of the Modem
PS      : Display detailed power and hardware configuration status.
```

```
MP:CM> ls

Current configuration of MP customer LAN interface
  MAC address   : 00:30:6e:05:d9:35
  IP address    : 166.77.2.58     (0xa64d023a)
  Hostname      : gsp0
  Subnet mask   : 255.255.255.0   (0xffffff00)
  Gateway       : 166.77.2.1      (0xa64d0201)
  Status        : UP and RUNNING
  AutoNegotiate : Enabled
  Data Rate     : 100 Mb/s
  Duplex        : Half
  Error Count   : 106
  Last Error    : late collision during tx

MP:CM> sysrev

    Cabinet FPGA and Firmware revision report

System Backplane : GPM    FM    OSP
                   ---    --    ---
                   1.00   1.01  1.0

PCI Backplane    : LPM   HS
                   ---   --
                   1.0   1.0

            LPM   BOB    Drain   SINC    PDC
            ---   ---    -----   ----    ---
CELL 0 :    2.01  1.00   1.00    1.12   14.004
CELL 1 :    2.01  1.00   1.00    1.12   14.004
CELL 2 :    2.01  1.00   1.00    1.12   14.004
CELL 3 :    0.00  0.00   0.00    0.00    0.000

                  FPGA    MP
                  ----    --
Master Core IO :  1.00    1.14
Slave Core IO  :  1.00    1.14


MP:CM> ma
MP:CM>

    MP MAIN MENU:

         CO: Consoles
        VFP: Virtual Front Panel
         CM: Command Menu
         CL: Console Logs
         SL: Show chassis Logs
         HE: Help
          X: Exit Connection

MP>
```

The **sysrev** output shows that there are three cell boards in the system, two of which we earlier saw in the first nPartition and one in the second. I configured the *Customer LAN* with updated IP information when the system was first installed with the **LC** command shown as one of the *CM* menu selections.

You may have to use the *RS* command to reset an nPartition at this level if the nPartition becomes hung.

The last two menu selections, **CL** and **SL** show logs for the consoles of the nPartitions and chassis logs, respectively.

Youhavetobecarefulinthe *CM* area because you can issue a reset or TOC to an nPartition.

You'll be using the Management Processor in your nPartition work often since it gives you access to a lot of important information and commands related to nPartitions.

[Team LiB]

PREVIOUS NEXT

The Boot Process Boot Console Handler (BCH) Commands

nPartitions boot in a similar manner to any HP 9000. Figure 17-2 shows the high-level steps involved in booting an nPartition:

Figure 17-5. Boot Steps

State	Description
Login	HP-UX login available on nPartition
vmunix	HP-UX boots on nPartition after ISL and Secondary System Loader (hpux) specify kernel.
hpux	Secondary System Load(hpux)
ISL	Initial System Load (ISL) available for single user, LVM maintenance, or alternate kernel boot.
BCH (interact here)	Boot Control Handler(BCH) is available for interacting with nPartition during boot.
Rendezvous	Rendezvous of active cells or Boot is Blocked (BIB)of inactive cells.
I/O	Each cell performs I/O discovery
POST	Each cell performs Power on Self Test(POST)
PDC	Processor Dependent Code (PDC) is run on Monarch processor in each cell. Cell may have Boot is Blocked.
Power-up	TOC Reset or Turn on Power

BCH is the primary way to interact with an nPartition at boot time, just as it is on any HP 9000. The steps and interfaces of the boot process meld together. There are a few different interfaces and levels of boot through which you work. When you get to the main menu of the Management Processor and issue commands you are using BCH. When you get past BCH, you reach ISL, which is considered the next step of the boot process beyond BCH. Then you reach the secondary system loader, which we'll see in the upcoming example by issuing the **hpux** command, which attempts to load the kernel. You need to know at what level of the boot process you are work ing so you can issue the appropriate commands.

When you access the console for an nPartition that is idle, you would issue the **BOOT** command and then stop the boot process if you wish to get an *ISL>* prompt or allow the nPartition to boot automatically. The BCH and boot process in general are not much different for an nPartition than they are for a much smaller system covered in Chapter 1.

The following listing shows several steps that we'll perform to boot an nPartition in the upcoming example. I'll issue a variety of commands to give you an idea of some steps you can perform during the boot process

at various prompts. We'll do the following:

- Go from the Management Processor to the *Command Menu* with **cm**.
- List the default customer LAN settings of the Management Processor with **ls**.
- Change the customer LAN settings with **lc**.
- List the new LAN settings with **ls**.
- Go back to the main Management Processor menu with **ma**.
- Go to the nPartition consoles with **co**.
- Select nPartition *0* with **0**.
- Boot from the primary boot path with **bo pri**.
- Specify that we want to stop at the *ISL* prompt.
- List *ISL* commands with **?**.
- Proceed the boot from the primary path with **hpux**.

```
Main Menu: Enter command or menu >

    GSP MAIN MENU:

         CO: Consoles
        VFP: Virtual Front Panel
         CM: Command Menu
         CL: Console Logs
         SL: Show chassis Logs
         HE: Help
          X: Exit Connection

GSP> cm                                                  <-- Go to Command Menu

               Enter HE to get a list of available commands

GSP:CM> ls                                              <-- List customer LAN

Current configuration of GSP customer LAN interface
  MAC address : 00:10:83:fd:77:f7
  IP address  : 192.168.1.1     0xc0a80101
  Name        : gsp0
  Subnet mask : 255.255.255.0   0xffffff00
  Gateway     : 192.168.1.1     0xc0a80101
  Status      : UP and RUNNING

Current configuration of GSP private LAN interface
  MAC address : 00:a0:f0:00:ce:1f
  IP address  : 192.168.2.11    0xc0a8020b
  Name        : priv-01
  Subnet mask : 255.255.255.0   0xffffff00
  Gateway     : 192.168.2.11    0xc0a8020b
  Status      : UP and RUNNING

GSP:CM> lc                                               <-- Modify customer LAN

This command modifies the LAN parameters.

Current configuration of GSP customer LAN interface
  MAC address : 00:10:83:fd:77:f7
  IP address  : 192.168.1.1     0xc0a80101
```

```
  Name        : gsp0
  Subnet mask : 255.255.255.0   0xffffff00
  Gateway     : 192.168.1.1     0xc0a80101
  Status      : UP and RUNNING


    Do you want to modify the configuration for the customer LAN? (Y/[N]) y

    Current IP Address is: 192.168.1.1
    Do you want to modify it? (Y/[N]) y
    Enter new IP Address: 157.235.10.249
    New IP Address will be: 157.235.10.249
    Please confirm (Y/[N]) y

    -> IP Address will be updated.

    Current GSP Network Name is: gsp0
    Do you want to modify it? (Y/[N]) n

    Current Subnet Mask is: 255.255.255.0
    Do you want to modify it? (Y/[N]) n

    Current Gateway is: 192.168.1.1
    Do you want to modify it? (Y/[N]) (Default will be IP address.) 157.235.10.2
46y

    Current Gateway is: 192.168.1.1
    Do you want to modify it? (Y/[N]) (Default will be IP address.) y
    Enter new Gateway: 157.235.10.246
    New Gateway will be: 157.235.10.246
    Please confirm (Y/[N]) y

    -> Gateway will be updated.

Current configuration of GSP private LAN interface
  MAC address : 00:a0:f0:00:ce:1f
  IP address  : 192.168.2.11    0xc0a8020b
  Name        : priv-01
  Subnet mask : 255.255.255.0   0xffffff00
  Gateway     : 192.168.2.11    0xc0a8020b
  Status      : UP and RUNNING

    Do you want to modify the configuration for the private LAN? (Y/[N]) n

    -> Parameters have been updated.

GSP:CM> ls                                          <-- List new LAN attributes

Current configuration of GSP customer LAN interface
  MAC address : 00:10:83:fd:77:f7
  IP address  : 157.235.10.249  0x9deb0af9
  Name        : gsp0
  Subnet mask : 255.255.255.0   0xffffff00
  Gateway     : 157.235.10.246  0x9deb0af6
  Status      : UP and RUNNING


Current configuration of GSP private LAN interface
  MAC address : 00:a0:f0:00:ce:1f
  IP address  : 192.168.2.11    0xc0a8020b
  Name        : priv-01
  Subnet mask : 255.255.255.0   0xffffff00
  Gateway     : 192.168.2.11    0xc0a8020b
  Status      : UP and RUNNING

GSP:CM> ma                                          <-- Go back to MAIN Menu
GSP:CM>

    GSP MAIN MENU:

         CO: Consoles
        VFP: Virtual Front Panel
         CM: Command Menu
         CL: Console Logs
         SL: Show chassis Logs
```

```
         HE: Help
          X: Exit Connection

GSP> co                                          <-- Go to Consoles

    Partitions available:

     #   Name
    ---  ----
     0)  asap60
     Q)  Quit

    Please select partition number: 0        <-- Select nPartition 0


        Connecting to Console: asap60

        (Use ^B to return to main menu.)

        [A few lines of context from the console log:]

- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -

     SERvice menu                       Displays service commands

     DIsplay                            Redisplay the current menu
     HElp [<menu>|<command>]            Display help for menu or command
     REBOOT                             Restart Partition
     RECONFIGRESET                      Reset to allow Reconfig Complex Profile

Main Menu: Enter command or menu >

- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -

Main Menu: Enter command or menu > bo pri              <-- Boot from pri path

     Primary Boot Path:  0/0/1/0/0.0


 Do you wish to stop at the ISL prompt prior to booting? (y/n) >> y

                                                  ^-- Stop at ISL

Initializing boot Device.


Boot IO Dependent Code (IODC) Revision 0


Boot Path Initialized.


HARD Booted.

ISL Revision A.00.43 Apr 12, 2000

ISL> ?                                           <-- List ISL commands

    HELP          Help Facility
    LS            List ISL utilities
    AUTOBOOT      Set or clear autoboot flag in stable storage
    AUTOSEARCH    Set or clear autosearch flag in stable storage
    PRIMPATH      Modify primary boot path in stable storage
    ALTPATH       Modify alternate boot path in stable storage
    CONSPATH      Modify system console path in stable storage
    DISPLAY       Display boot and console paths in stable storage
    LSAUTOFL      List contents of autoboot file
    FASTSIZE      Sets or displays FASTSIZE
    800SUPPORT    Boots the s800 Support Kernel from the boot device
    700SUPPORT    Boot the s700 Support Kernel from the boot device
    READNVM       Displays contents of one word of NVM
    READSS        Displays contents of one word of stable storage
    LSBATCH       List contents of batch file
    BATCH         Execute commands in batch file
```

```
    LSEST         List contents of EST (Extended Self Test) file
    EST           Execute commands in EST (Extended Self Test) file
    EXIT          Return to the Boot Console Handler

Enter 'LS' to see a list of the ISL utilities.

ISL> hpux                                          <-- Proceed to boot from pri

Boot
: disk(0/0/1/0/0.0.0.0.0.0.0;0)/stand/vmunix
9814016 + 1875968 + 1340312 start 0x1f4b68


gate64: sysvec_vaddr = 0xc0002000 for 2 pages
NOTICE: nfs3_link(): File system was registered at index 4.
NOTICE: autofs_link(): File system was registered at index 6.
NOTICE: cachefs_link(): File system was registered at index 7.
td: claimed Tachyon TL/TS Fibre Channel Mass Storage card at 0/0/2/0/0
td: claimed Tachyon TL/TS Fibre Channel Mass Storage card at 0/0/3/0/0
td: claimed Tachyon TL/TS Fibre Channel Mass Storage card at 0/0/10/0/0
td: claimed Tachyon TL/TS Fibre Channel Mass Storage card at 2/0/2/0/0
td: claimed Tachyon TL/TS Fibre Channel Mass Storage card at 2/0/3/0/0
td: claimed Tachyon TL/TS Fibre Channel Mass Storage card at 2/0/10/0/0

    System Console is on the Built-In Serial Interface
    Entering cifs_init...
    Initialization finished successfully... slot is 9
    Logical volume 64, 0x3 configured as ROOT
    Logical volume 64, 0x2 configured as SWAP
    Logical volume 64, 0x2 configured as DUMP
    Logical volume 64, 0xa configured as DUMP
        Swap device table: (start & size given in 512-byte blocks)
            entry 0 - major is 64, minor is 0x2; start = 0, size = 8388608
    Starting the STREAMS daemons-phase 1
    Checking root file system.
    file system is clean - log replay is not required
    Root check done.
    Create STCP device files
    Starting the STREAMS daemons-phase 2
            $Revision: vmunix:    vw: -proj    selectors: CUPI80_BL2000_1108 -c 'Vw
    for CUPI80_BL2000_1108 build' -- cupi80_bl2000_1108 'CUPI80_BL2000_1108'  Wed No
    v  8 19:24:56 PST 2000 $
    Memory Information:
        physical page size = 4096 bytes, logical page size = 4096 bytes
        Physical: 16760832 Kbytes, lockable: 13462944 Kbytes, available: 15450828 Kb
    ytes

    /sbin/ioinitrc:
    /sbin/krs_sysinit:


    /sbin/bcheckrc:
    Checking for LVM volume groups and Activating (if any exist)
    Volume group "/dev/vg00" has been successfully changed.
    Resynchronized volume group /dev/vg00
    vxfs fsck: sanity check: root file system OK (mounted read/write)
    Checking hfs file systems
    /sbin/fsclean: /dev/vg00/lvol1 (mounted) ok
    HFS file systems are OK, not running fsck
    Checking vxfs file systems
    /dev/vg00/lvol8 :
    vxfs fsck: sanity check: /dev/vg00/lvol8 OK
    /dev/vg00/lvol9 :
    vxfs fsck: sanity check: /dev/vg00/lvol9 OK
    /dev/vg00/lvol3 :
    vxfs fsck: sanity check: root file system OK (mounted read/write)
    /dev/vg00/lvol4 :
    vxfs fsck: sanity check: /dev/vg00/lvol4 OK
    /dev/vg00/lvol5 :
    vxfs fsck: sanity check: /dev/vg00/lvol5 OK
    /dev/vg00/lvol6 :
    vxfs fsck: sanity check: /dev/vg00/lvol6 OK
    /dev/vg00/lvol7 :
    vxfs fsck: sanity check: /dev/vg00/lvol7 OK
```

```
    Cleaning /etc/ptmp...
    (c)Copyright 1983-2000 Hewlett-Packard Co.,  All Rights Reserved.
    (c)Copyright 1979, 1980, 1983, 1985-1993 The Regents of the Univ. of California
    (c)Copyright 1980, 1984, 1986 Novell, Inc.
    (c)Copyright 1986-1992 Sun Microsystems, Inc.
    (c)Copyright 1985, 1986, 1988 Massachusetts Institute of Technology
    (c)Copyright 1989-1993  The Open Software Foundation, Inc.
    (c)Copyright 1986 Digital Equipment Corp.
    (c)Copyright 1990 Motorola, Inc.
    (c)Copyright 1990, 1991, 1992 Cornell University
    (c)Copyright 1989-1991 The University of Maryland
    (c)Copyright 1988 Carnegie Mellon University
    (c)Copyright 1991-2000 Mentat Inc.
    (c)Copyright 1996 Morning Star Technologies, Inc.
    (c)Copyright 1996 Progressive Systems, Inc.
    (c)Copyright 1991-2000 Isogon Corporation, All Rights Reserved.


                              RESTRICTED RIGHTS LEGEND
    Use, duplication, or disclosure by the U.S. Government is subject to
    restrictions as set forth in sub-paragraph (c)(1)(ii) of the Rights in
    Technical Data and Computer Software clause in DFARS 252.227-7013.

                              Hewlett-Packard Company
                              3000 Hanover Street
                          Palo Alto, CA 94304 U.S.A.

Rights for non-DOD U.S. Government Departments and Agencies are as set
forth in FAR 52.227-19(c)(1,2).

/sbin/auto_parms: DHCP access is disabled (see /etc/auto_parms.log)


     HP-UX Start-up in progress
     ___________________________
                      <-- All startup programs run and Login prompt provided


The system is ready.

GenericSysName [HP Release B.11.11] (see /etc/issue)

Console Login:
```

We have only scratched the surface of BCH and other boot-related commands in this example and section. Knowing where you are in the boot process and the commands available to you is important in working with nPartitions, so that you can perform functions on the nPartition on which you're working or the complex in general.



Chapter 18. HP Process Resource Manager (PRM)

PRM and WLM Overview

PRM Software

Working with PRM

PRM and Oracle Setup



PRM and WLM Overview

This chapter provides a quick overview and some examples of using HP Process Resource Manager (PRM). PRM is a tool that controls the amount of resources processes use. The control over the amount of resources used is normally done when the system is at peak load with 100% utilization of CPU, memory, or disk. Another way of controlling resources is to use caps. The cap is in place regardless of the overall loading on the system. We'll cover both the peak load resource control as well as the capping approach in our examples in this chapter. Then we'll cover an Oracle-specific example of using PRM. The following list describes the three examples we'll cover:

1. Manual PRM setup and resource caps.
2. GUI-based PRM setup using **xprm**.
3. Oracle PRM setup with five database instances.

The document *HP Process Resource Manager User's Guide* that can be found on www.docs.hp.com is a great manual that thoroughly covers all aspects of PRM. This chapter will give you an idea of how PRM works and some good examples of PRM setup. If you think PRM would be useful in your computing environment, I recommend that you download the full *User's Guide.*

Not covered in this book is Workload Manager (WLM). Workload Manager adjusts system resources to help achieve service-level objectives (SLOs) that you define. WLM uses PRM including PRM's *APPL, CPU, DISK,* and *MEM* managers. Many PRM commands covered in this chapter are also available. WLM also works across Virtual Partitions (vPars) moving resources within and between vPars as required. In a vPars environment you setup and activate a WLM configuration file for each vPar. You then setup the configuration file for the global arbiter that is aware of all of the vPars and their resources. If you think WLM would be useful in your computing environment I recommend that you download the full *HP-UX Workload Manager User's Guide.*

The next section covers PRM software and files.

PRM Software

In HP-UX 11i PRM is one of the bundled products of the Enterprise Operating Environment. The following listing shows PRM as part of an **swlist**:

```
root@dwdbp1[/.root] > swlist HPUX11i-OE-Ent
# Initializing...
# Contacting target "dwdbp1"...
#
# Target:  dwdbp1:/
#

# HPUX11i-OE-Ent        B.11.11.0109      HP-UX Enterprise Operating Environment Component
  HPUX11i-OE-Ent.ApacheStrong1.3.19.20    HP Apache-based Web Server with
                                               Strong (128bit) Encryption
  HPUX11i-OE-Ent.ServControlMgrA.02.03    HP-UX ServiceControl Manager
  HPUX11i-OE-Ent.Proc-Resrc-MgrC.02.00    Process Resource Manager Proc-Resrc-Mgr product
  HPUX11i-OE-Ent.PRM-Sw-LibC.02.00        Process Resource Manager PRM-Sw-Lib product
  HPUX11i-OE-Ent.PAM-NTLMA.01.06          HP NTLM Pluggable Authentication Module
  HPUX11i-OE-Ent.PAM-KerberosB.11.11     PAM-Kerberos Version 1.0
  HPUX11i-OE-Ent.OnlineJFSB.11.11         Online features of the VxFS File System
  HPUX11i-OE-Ent.OVOPC-SEA.05.57          IT/Operations Special Edition Product
  HPUX11i-OE-Ent.OVOPC-SE-GUIA.05.57      IT/Operations Special Edition Java UI
  HPUX11i-OE-Ent.OVOPC-SE-DOCA.05.57      IT/Operations Special Edition Documentation
  HPUX11i-OE-Ent.OVOPC-DFLTA.05.57        IT/Operations Special Edition Default
  HPUX11i-OE-Ent.OVOPC-AGTA.05.57         IT/Operations Agent
  HPUX11i-OE-Ent.OE     B.11.11.0109      HP-UX OE control script product
  HPUX11i-OE-Ent.NS-communicateB.11.00.05  Netscape Communicator 4.75
  HPUX11i-OE-Ent.MeasurementIntC.03.35.00  HP-UX Measurement Interface for 11i
  HPUX11i-OE-Ent.MeasureWareC.03.35.00     MeasureWare Software/UX
  HPUX11i-OE-Ent.LVM    B.11.11           LVM
  HPUX11i-OE-Ent.Java2RTE_perf1.2.2.08B.02 Java 2 RTE (700/800), PA2.0 Performance
  HPUX11i-OE-Ent.Java2RTE_doc1.2.2.08B.02  Java 2 RTE (700/800), Distribution License
  HPUX11i-OE-Ent.Java2RTE_base1.2.2.08B.02 Java 2 RTE (700/800), PA1.1 Portable
  HPUX11i-OE-Ent.Java-PlugIn1-21.2.2.08B.02    HPplug-In for Java 2
  HPUX11i-OE-Ent.Glance C.03.35.00          HP GlancePlus/UX
  HPUX11i-OE-Ent.EMS-RdbmsMonA.03.20.01        EMS Database Monitor
  HPUX11i-OE-Ent.EMS-MIBMonitorA.03.20.01      EMS MIB Resource Monitor Product
  HPUX11i-OE-Ent.EMS-DiskMonitorA.03.20.01     EMS Disk Resource Monitor
  HPUX11i-OE-Ent.CIFS-ServerA.01.07            CIFS/9000 (Samba)
  HPUX11i-OE-Ent.CIFS-DevelopmentA.01.07       CIFS/9000 server source code
  HPUX11i-OE-Ent.CIFS-ClientA.01.06            HP CIFS/9000 Client
root@dwdbp1[/.root] >
```

The PRM-related software is shown in the 11i Enterprise Operating Environment.

PRM is loaded in **/opt/prm**. The following is a long listing of this directory:

```
root@dwdbp1[/.root] > ls -l /opt/prm
total 44
dr-xr-xr-x   3 bin        bin         1024 Oct 29 18:28 bin
drwxr-xr-x   2 root       sys           96 Oct 29 18:29 conf
dr-xr-xr-x   2 bin        bin           96 Oct 29 18:28 doc
-rw-r--r--   1 root       sys          110 Jul 19  2001 exempt
dr-xr-xr-x   3 bin        bin           96 Oct 29 18:29 help
dr-xr-xr-x   3 bin        bin           96 Oct 29 18:29 lbin
dr-xr-xr-x   3 bin        bin         1024 Oct 29 18:29 lib
dr-xr-xr-x   6 bin        bin           96 Oct 29 18:28 man
dr-xr-xr-x   3 bin        bin           96 Oct 29 18:29 newconfig
-r--r--r--   1 bin        bin        18847 Jul 19  2001 prm.my
```

The release notes for PRM are in the directory **/opt/prm/newconfig/RelNotes.**

After PRM is loaded, we have to set the value of variables in **/etc/rc.config.d/prm** to suit our needs. The variables in this file give you control over such functions as whether or not capping is enabled, configuring PRM on reboot, and other such basic setup functions. These variables are described in both the startup file and *User's Guide*. You can set all of the variables to *1* to enable everything, or pick and choose which functions you want enabled. The following is a list of the variables with all functions enabled except PRM's SNMP agent:

```
PRM_CONFIG=1        # PRM configured on reboot
PRM_ENABLE=1        # Enable resource managers
PRM_CAPPING=1       # Enable CPU capping
PRM_SNMPAGT=0       # Don't start PRM's SNMP agent
PRM_RMTCONF=1       # Enable PRM remote configuration
```

The following listing shows the entire contents of the **/etc/rc.config.d/prm** file.

```
root@dwdbp1[/.root] > cat /etc/rc.config.d/prm
#!/sbin/sh
#
# PRM configuration.  See prmconfig(1)
#
# To configure (and enable) PRM automatically at boot time, PRM must have
# been previously configured.  The most recent configuration will be used,
# that is the configuration in place prior to system shutdown or before the
# most recent prmconfig -r (reset).
#
# PRM_CONFIG: Set to 1 to configure PRM
# PRM_ENABLE: Set to 1 to enable PRM
# PRM_CAPPING: Set to 1 to start PRM in CPU capping mode when enabled
# PRM_SNMPAGT: Set to 1, always spawn prmagt at boot time
# PRM_RMTCONF: Set to 1, always spawn remote config daemon at boot
#
# NOTE:  in order to use PRM_ENABLE=1, you must set PRM_CONFIG=1
# NOTE:  in order to use PRM_CAPPING=1, you must set PRM_ENABLE=1
#
PRM_CONFIG=1
PRM_ENABLE=1
PRM_CAPPING=1
PRM_SNMPAGT=0
PRM_RMTCONF=1
```

With PRM loaded and running, we can now begin to configure PRM to manage resources.

[Team LiB]

PREVIOUS NEXT

Working with PRM

You can work with PRM at the command line or through the **xprm** graphical interface. We'll start with command-line work in our examples because you learn more about the application at the command line than at the graphical interface. We'll then show the results of our command-line work in the graphical interface.

Table 18-1 includes a brief description of PRM-related commands and some files of interest. There is much more detail on these commands and files in the *User's Guide*.

Table 18-1. PRM Command Summary and Files

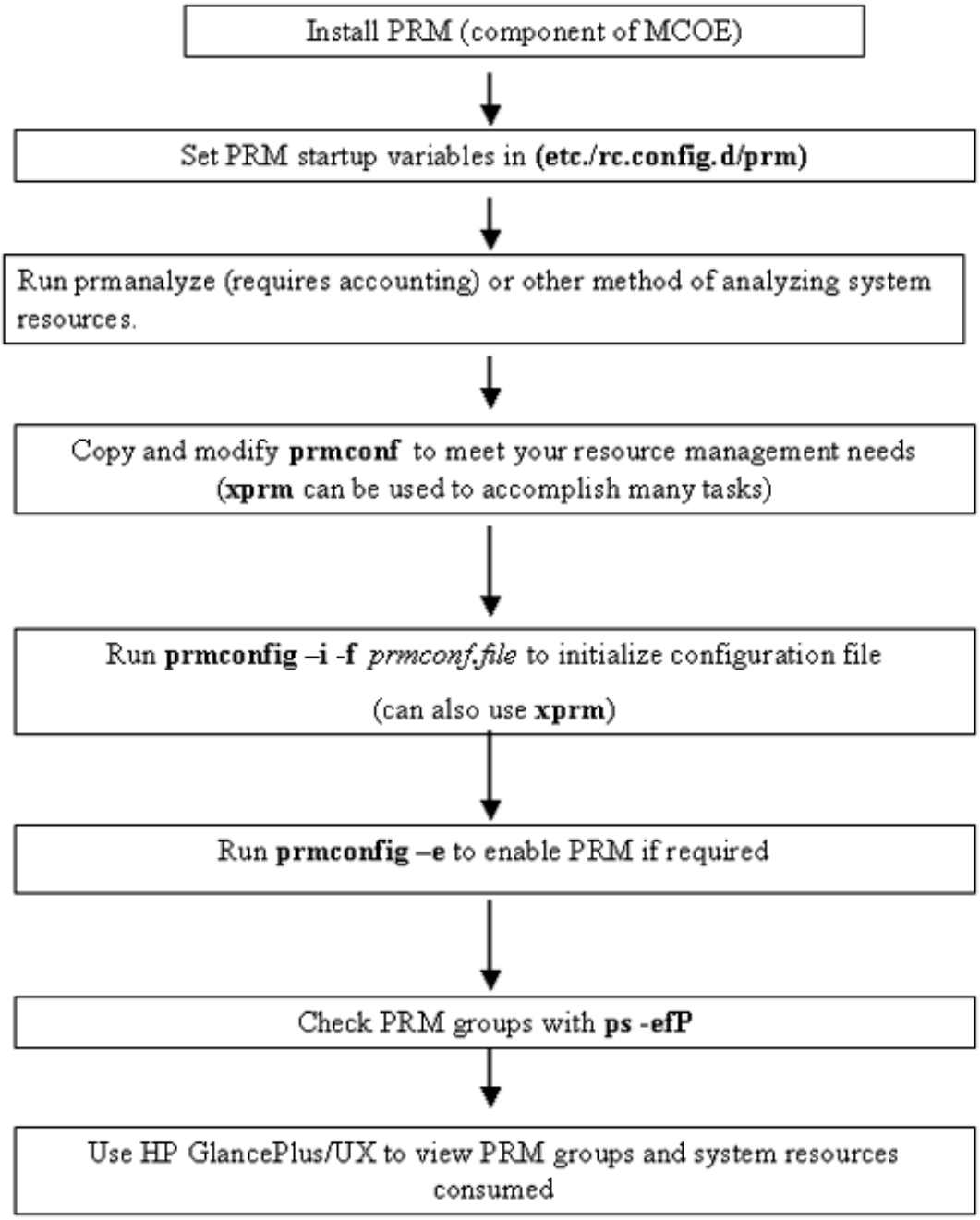
Command	Description
prmagt	Start and stop PMR SNMP agent. If *PRM_SNMPAGT* is set to *1* in **/etc/rcconfig.d/prm** then the SNMP agent is started.
prmanalyze	Analyze system resource utilization. This command is useful in the planning of system resource utilization with PRM. You must have HP-UX accounting running in order to run **prmanalyze**.
prmavail	Displays estimated resource availability.
prmconfig	Perform various PRM configuration-related tasks with this command such as: checking configuration files; and enabling and disabling PRM.
prmlist	Displays current PRM information such as: group: user; and so on.
prmloadconf	Running this command creates or updates an existing PRM configuration file.
prmmonitor	This command monitors and current PRM configuration and resource usage.
prmmove	This command moves processes or groups of processes to a different PRM group.
prmrecover	Returns the memory manager to its default state.
prmrun	Runs an application in its assigned PRM group or in the group specified.
ps -efP	Produce a list of processes including the PRM group to which they belong under the *PRMID* heading in the **ps** output.
xprm	Starts the X-Windows PRM graphical user interface.
/etc/rc.config.d/prm	Startup configuration file for PRM.
/etc/prmconf	This file defines PRM groups and how you want resources controlled.
time	Time a command to see the affect PRM resource assignments have on time-to-completion.
glance (or **gpm** for graphical interface to GlancePlus)	GlancePlus is used to view PRM groups and the amount of resources they are consuming using the PRM List tab which is F2.

We'll use some of these commands and files in our upcoming examples.

The key file to configuring PRM is **/etc/prmconf**. This file defines PRM groups and how you want resources controlled. You would typically configure this file to meet your needs. You could either update the file or a copy of it. You can also modify this file or a copy of it using **xprm**. After making modifications to the file you would check and load the configuration file with **prmconfig**. You can confirm that the processes are running in the correct PRM groups with **ps -efP**. If you are running accounting on your system you can use **prmanalyze** to scan accounting files and produce reports about various system resources. After you have

modified **/etc/prmconf** you can use GlancePlus/UX to view the resources consumed by PRM groups and confirm that the configuration you have put in place meets your needs. Figure 18-1 shows the high-level steps you might perform when working with PRM.

Figure 18-1. Possible PRM Steps



You might perform only some of these steps and add others that are not shown in the figure depending on your needs and your environment. These are only some recommended PRM-related steps.

The following listing shows the default **/etc/prmconf** file automatically produced on a system:

```
root@dwdbp1[/.root] > cat /etc/prmconf
#
#
#
```

```
# Group/CPU records
#
DW:2:100::
OTHERS:1:100::
#
# Memory records
#
#
# Application records
#
#
# Disk bandwidth records
#
#
# User records
#
adm:::OTHERS
bin:::OTHERS
daemon:::OTHERS
hpdb:::OTHERS
lp:::OTHERS
nobody:::OTHERS
nuucp:::OTHERS
opc_op:::OTHERS
smbnull:::OTHERS
sys:::OTHERS
uucp:::OTHERS
webadmin:::OTHERS
www:::OTHERS
root@dwdbp1[/.root] >
root@dwdbp1[/.root] >
```

This file has in it only the two PRM groups *DW* and *OTHERS*. In this file you'll typically produce your own list of PRM groups and define the amount of resources each group can consume when the system is heavily loaded. You can control a group's CPU, memory, and disk resources. PRM groups are assigned on a per-process basis and are not related to UNIX group assignments such as those in **/etc/group**. The PRM groups use the Fair Share Scheduler (FSS) to obtain the amount of overall system resources you have defined for a given group. Alternatively you could define groups on a Processor Set (PSET) basis, meaning that the group will obtain processor resources based on a subset of the server's total processors in the PSET. The examples in this chapter use the FSS method.

The resources defined by each group are in the form of shares. Resource shares are the minimum amount of resources assigned to PRM groups when the system is at peak load. You don't care about these shares at times other than peak load unless you are using resource caps. One of the upcoming sections in this chapter covers resource caps which are enforced all of the time, even if the system is not at peak load.

When you allocate shares to a PRM group they are converted to percentages of system resources. Figure 18-2 shows two examples of shares and the corresponding percentages:

Figure 18-2. Share and Percentage Examples

PRM GROUP NAME	PRM CPM SHARES	CPU %
Group 1	20	20/40= 40%
Group 2	10	10/40= 25%
Others	10	10/40= 25%
PRM Systems	Automatically defined and resources allocated by PRM	

PRM GROUP NAME	PRM CPM SHARES	CPU %
Group 1	20	20/50= 50%
Group 2	10	10/50= 20%
Group 3	10	10/50= 20%
Others	10	10/50= 20%
PRM Systems	Automatically defined and resources allocated by PRM	

The top example in Figure 18-2 shows a total of four groups. The *PRM_SYS* group is automatically created for you and assigned system resources. You can manually add it to the configuration file and assign its share if you wish or just let PRM handle it automatically. *PRM_SYS* has PRMID *0*. The *OTHERS* group has in it all users in **/etc/passwd** by default. *OTHERS* has a PRMID of *1*. You can assign users to PRM groups in **/etc/prmconf** as you see fit. Although all **/etc/passwd** users go into *OTHERS* by default, you can list the users and their PRM group in **/etc/prmconf**.

PRM groups have the following syntax for each record:

```
GPOUP:{ PRMID | HIER | PSET }:SHARES::[[CPUS] : [CPU_LIST]
```

In the upcoming capping example we created the following record for user *db_hp* to place her in the PRM group *db2*. Then there is a record for the PRM group *db2* showing its ID and share:

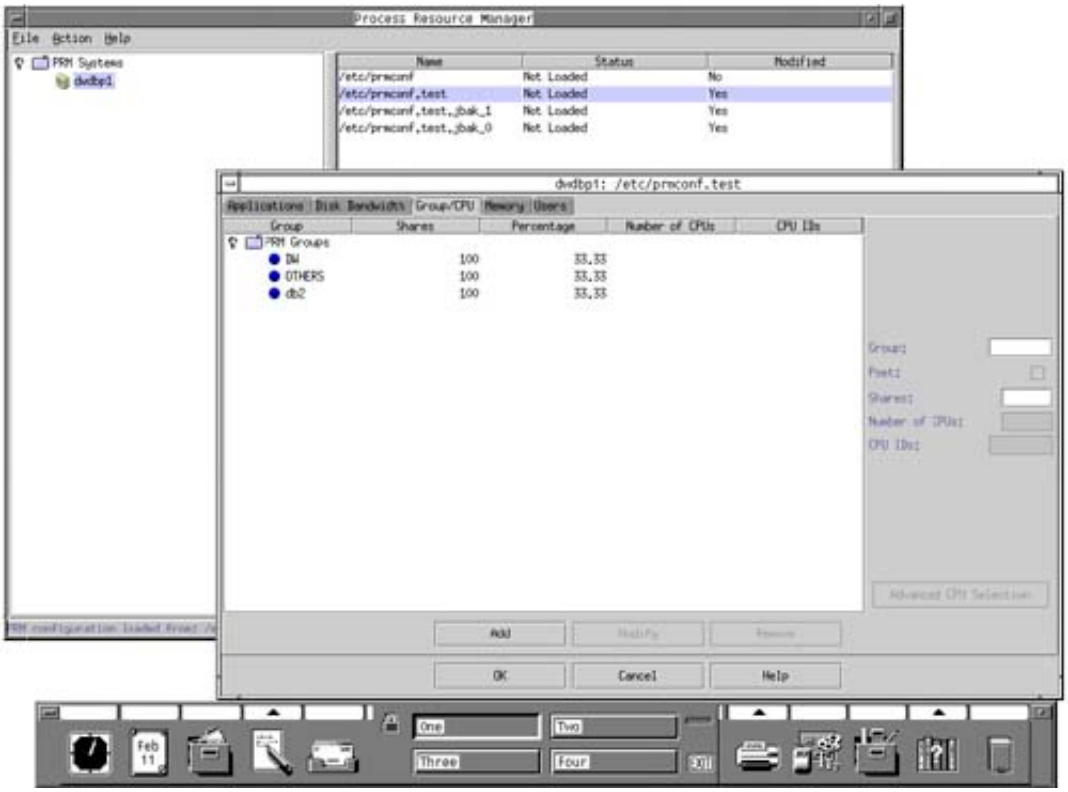
```
root@dwdbp1[/etc] > cat prmconf.test
#
#
#
# Group/CPU records
#
DW:2:100::
OTHERS:1:100::
db2:3:100::
#
# Memory records
#
#
# Application records
#
```

```
#
# Disk bandwidth records
#
#
# User records
#
db_hp::::db2
adm::::OTHERS
bin::::OTHERS
daemon::::OTHERS
hpdb::::OTHERS
lp::::OTHERS
nobody::::OTHERS
nuucp::::OTHERS
opc_op::::OTHERS
smbnull::::OTHERS
sys::::OTHERS
uucp::::OTHERS
webadmin::::OTHERS
www::::OTHERS
root@dwdbp1[/etc] >
```

The PRM group *db2* has a PRMID of *3* and a share of *100* in this example. There is one user in the group db2 with a name *db_hp*.

The **prmconf** file is easy to modify to meet your PRM needs. You can also use **xprm** to view and change PRM configuration information as shown in Figure 18-3:

Figure 18-3. xprm and Configuration Data prm1.bmp



This **xprm** window has several tabs across the top that are used to control various aspects of the PRM configuration file we have loaded. The *Group/CPU* tab is selected in this screen shot. Notice that the three groups have an equal share of *100,* which gives them each one third of the overall system resources. The *100* share was shown in the earlier **prmconf.test** listing. You can adjust these shares and any aspect of the groups through the **xprm** interface.

When assigning your groups and shares, it is useful to know the amount of resources you have at your

disposal to assign to PRM groups. The following shows the output of the **prmavail** command:

```
# prmavail

4 CPUs
846200 real memory pages or 3305 MB available (PRM estimate)
9 volume groups
              /dev/vg00
              /dev/vgmti1
              /dev/vgmti2
              /dev/vgmti3
              /dev/vgmti4
              /dev/vgmti5
              /dev/vgmti6
              /dev/vgmti7
              /dev/vgmti8
```

This output shows a Superdome hard partition that has the CPUs, memory, and volumes available for use.

Now that we've covered many of the basics related to PRM, let's jump into a couple of examples of using PRM. The first example implements capping of system resources regardless of whether the system is at peak load or not. The second example limits resources at the time of peak load. This is the mostly widely used implementation of PRM.

Manual (non-GUI) Setup of PRM with Resource Caps

You can cap the amount of system resources a given user or group of users consumes with PRM. In this example we have a group of Oracle users on a system. We're about to add a second group of non-Oracle users to the system so we want to cap the amount of CPU resources each group of users can consume. This is done because we don't want one group to become accustomed to unlimited resources and then add additional groups that degrade performance. The setup we perform is the same whether or not caps are used. We simply enable caps if we want them employed or disable caps if we don't want to use them. This section of the chapter focuses on manual PRM configuration.

The existing users are put into one PRM group and new users are put into a second group. In our example the second group will have only one user in it in order to keep it simple. Our example is a proof-of-concept to show that the resources that our new user consumes can be limited regardless of the overall loading on the system. We'll do the following in this example:

High Share	Set the share for the new user high and run a job that consumes a lot of systems resources. Use HP GlancePlus/UX to show the system resources consumed by the new user.
Low Share	Set the share for the new user much lower and run the same job. Use HP GlancePlus/UX to show that the system resources consumed are much lower and that the time to complete the task takes longer.

This is a good PRM example because it clearly demonstrates the dramatic effect that lowering the share has on system resources and the time required to complete the job.

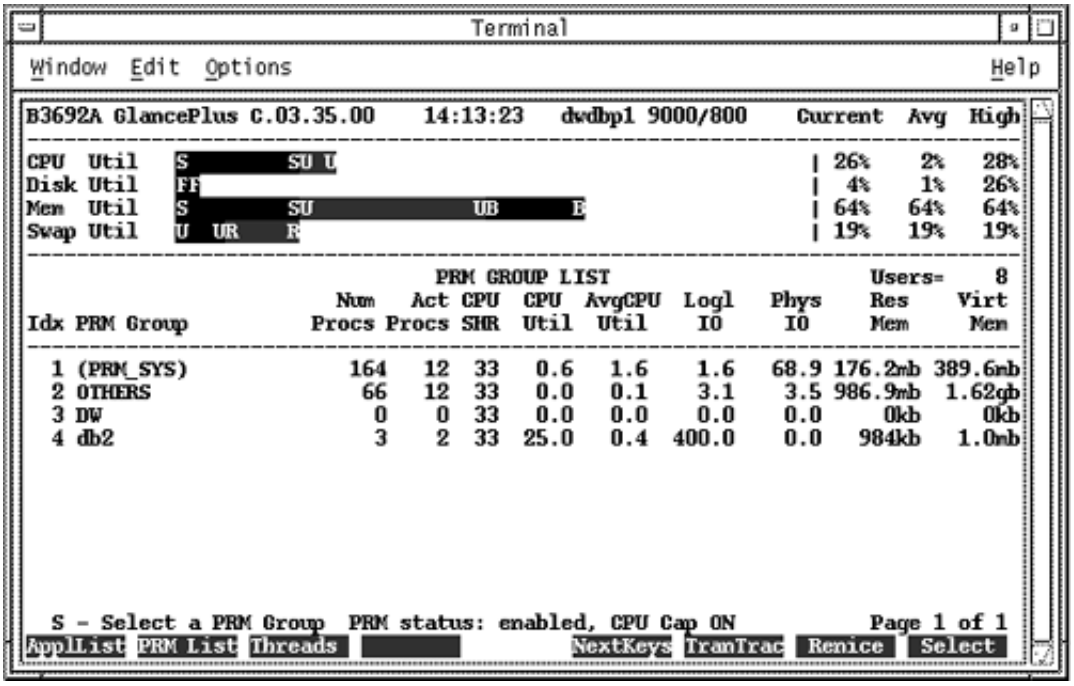
When we loaded and configured PRM, all of the existing users in **/etc/passwd** were put into the default PRM group of *OTHERS*. We then created a new user called *db_hp* and placed this user in its own PRM group called *db2*. Initially, all of the PRM groups had an equal share of 100 for system resources. The following is the **/etc/prmconf** listing, showing all PRM-related information:

```
++++++++++++++++++++++++
#
#
#
# Group/CPU records
#
DW:2:100::              <-- Group of OTHERS, share of 100
OTHERS:1:100::          <-- Default group of OTHERS, share of 100
db2:3:100::             <-- Group db2 with one user, share of 100
#
# Memory records
#
#
# Application records
```

```
#
#
# Disk bandwidth records
#
#
# User records
#
db_hp:::db2
adm:::OTHERS
bin:::OTHERS
daemon:::OTHERS
hpdb:::OTHERS
lp:::OTHERS
nobody:::OTHERS
nuucp:::OTHERS
opc_op:::OTHERS
smbnull:::OTHERS
sys:::OTHERS
uucp:::OTHERS
webadmin:::OTHERS
www:::OTHERS
```

User *db_hp* in PRM group *db2* will now run a job that consumes a substantial amount of system resources. Figure 18-4 shows a GlancePlus screen shot of the PRM groups and the amount of resources they are consuming using the *PRM List* tab, which is *F2*:

Figure 18-4. HP GlancePlus/UX Showing PRM Group System Resources.



Notice that all of the PRM groups have been allocated an equal amount of system resources, based on the share definitions of *100*, which is reflected in the GlancePlus screen shot as a percentage of CPU.

The job that PRM Group *db2* is running can't consume more than its maximum allocated cap of CPU resources even though the system is not heavily loaded and additional resources are available. At the time this screen shot was produced, PRM group *db2* was consuming its maximum of 25% system resources even though only 26% of the CPU was in use. *db2* would like to consume more system resources, but its cap is enforced, limiting it to 25%. Note also that the first PRM group *(PRM_SYS)* was not in the active PRM configuration file **/etc/prmconf.test**. This PRM group is produced by default.

Not shown in the *PRM List* screen is that this is a four-processor server (this is shown in the *CPU Rpt* screen). This server is a hard partition (nPartition) of a Superdome system. It is, for all intents and purposes, a separate server even though it is in the same Superdome complex as several other nPartitions.

The percentages of CPU resources shown in this example use all four processors and the FSS scheduling method.

We can confirm the number of processors, and other resources available to us, with the **prmavail** command as shown below:

```
# prmavail

4 CPUs
846200 real memory pages or 3305 MB available (PRM estimate)
9 volume groups
              /dev/vg00
              /dev/vgmti1
              /dev/vgmti2
              /dev/vgmti3
              /dev/vgmti4
              /dev/vgmti5
              /dev/vgmti6
              /dev/vgmti7
              /dev/vgmti8
```

This output shows a Superdome hard partition that has the CPUs, memory, and volumes available for use.

The **time** command was run to see the amount of time the job would take to complete with a share of *100,* as shown in the following output:

```
$ time /tmp/job
real       44.5
user        7.7
sys        35.6
$
```

This job took 44.5 seconds to complete with its share of *100* and maximum system resources cap of *25%*.

The following listing shows running the **ps -efP** command to view the PRM groups in which processes are running:

```
$ ps -efP
real       44.5
user        7.7
sys        35.6
$
```

We'll now reduce the share for PRM group *db2* from *100* to *10* in the active PRM configuration file **/etc/prmconf.test** to see how this affects the running of the job. After editing the file and reducing the share of *db2* from *100* to *10,* we'll run the **prmconfig** command shown below:

```
# prmconfig -i -f /etc/prmconf.test

PRM configured from file:  /etc/prmconf.test
File last modified:        Mon Feb 11 12:43:01 2002

PRM CPU scheduler state:  Enabled, CPU cap ON

PRM Group                       PRMID    CPU Entitlement
-----------------------------------------------------
DW                                 2             47.62%
OTHERS                             1             47.62%
db2                                3              4.76%

PRM memory manager state:  Not Running

PRM User                        Initial Group       Alternate Group(s)
-------------------------------------------------------------------
adm                             OTHERS
bin                             OTHERS
daemon                          OTHERS
db_hp                           db2
hpdb                            OTHERS
lp                              OTHERS
nobody                          OTHERS
nuucp                           OTHERS
opc_op                          OTHERS
root                            (PRM_SYS)
```

```
smbnull                         OTHERS
sys                             OTHERS
uucp                            OTHERS
webadmin                        OTHERS
www                             OTHERS

PRM application manager state: Enabled  (polling interval: 30 seconds)
PRM application manager logging state:  Disabled

Disk manager state:  Disabled
#
```

The *-i* option creates the PRM groups and moves all currently running applications and user processes into their assigned groups. In our case the modification to the shares also takes effect. The *-f* option specifies that the configuration file **/etc/prmconf.test** is to be used. The default configuration file **/etc/prmconf** is used if *-f* and the configuration file are not specified.

The previous output shows that the percentage of system resources that *db2* can use has dropped commensurate with the reduction in its share.

We can also run **prmlist** to see the PRM group-related information, as shown in the following listing:

```
# prmlist

PRM configured from file:  /etc/prmconf.test
File last modified:        Mon Feb 11 12:43:01 2002
PRM Group                       PRMID    CPU Entitlement
-------------------------------------------------------
DW                                 2             47.62%
OTHERS                             1             47.62%
db2                                3              4.76%

PRM User                        Initial Group         Alternate Group(s)
-------------------------------------------------------------------------
adm                             OTHERS
bin                             OTHERS
daemon                          OTHERS
db_hp                           db2
hpdb                            OTHERS
lp                              OTHERS
nobody                          OTHERS
nuucp                           OTHERS
opc_op                          OTHERS
root                            (PRM_SYS)
smbnull                         OTHERS
sys                             OTHERS
uucp                            OTHERS
webadmin                        OTHERS
www                             OTHERS
```

The **prmlist** output reflects the information that we had put in the configuration file.

Figure 18-5 shows the new percentages in the GlancePlus screen and the much lower cap on system resources.

Figure 18-5. HP GlancePlus/UX Showing Reduced *db2* Resources

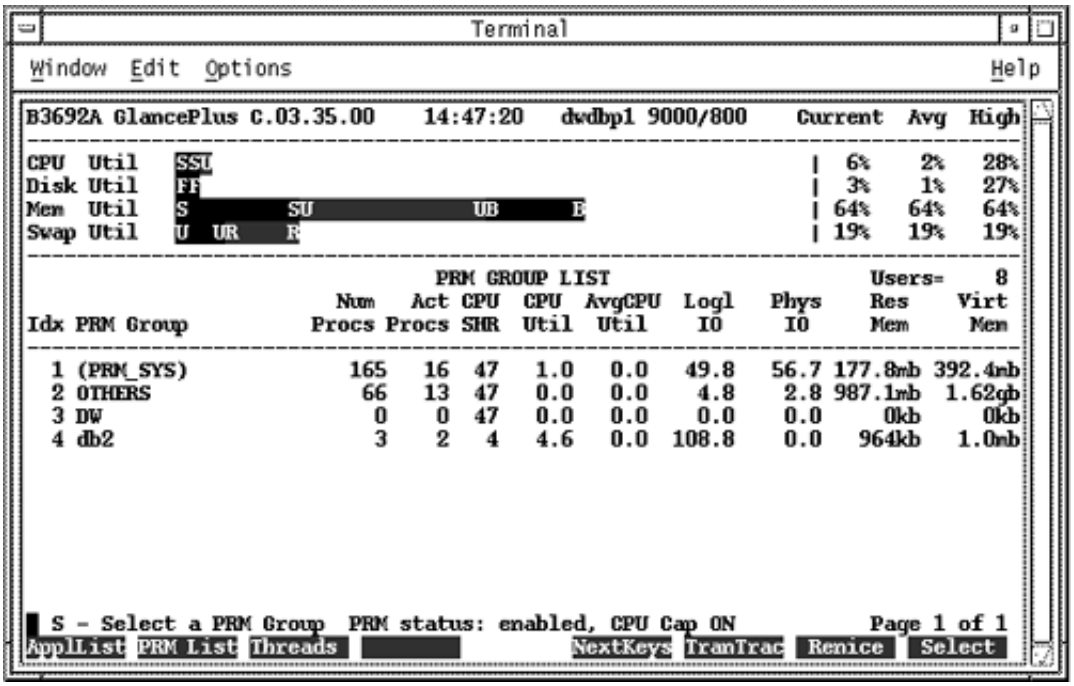


Figure 18-5 shows that a lot of system resources are available; however, our cap of *10* share greatly reduces the percentage of CPU resources PRM group *db2* can consume. The change of share and percentage of CPU for the PRM groups automatically appears in this GlancePlus window after the **prmconfig** command is run. This tight integration between PRM and GlancePlus is useful for making changes in PRM and immediately seeing the results in GlancePlus. There are even more free resources now that we have lowered the share of PRM group *db2*; however, these remain idle even though we need them because our cap is in place and will not assign any system resources beyond those capped by our share.

The job also takes much longer, as shown in the following output:

```
$ time /tmp/job              <-- results with 100 share
real       44.5
user        7.7
sys        35.6
$
$ time /tmp/job              <-- results with 10 share
real     4:10.5
user        7.7
sys        35.3
```

The cap we implemented had the huge effect of going from a *real* time of 44.5 seconds to 250.5 seconds as a result of having capped the resources of users in our PRM group *db2*.

We can also use the graphical form of GlancePlus invoked with **gpm** to view the job as it runs, as shown in Figure 18-6:

Figure 18-6. gpm Showing Reduced *db2* Resources

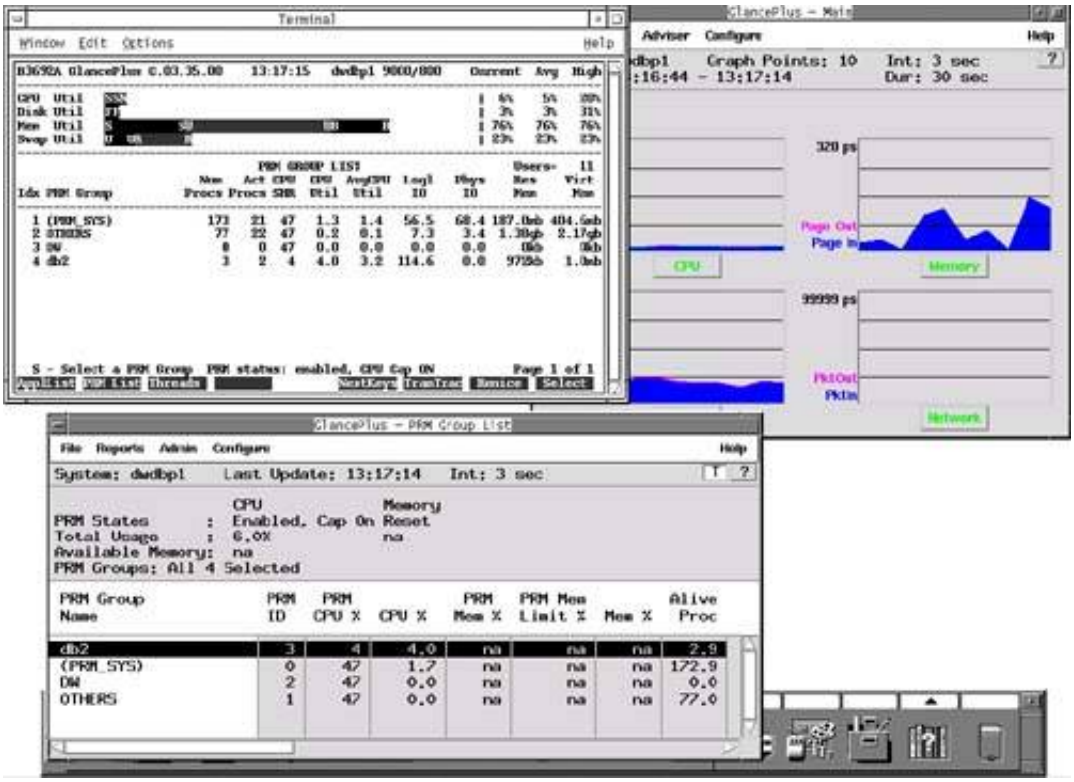


Figure 18-6 shows both the graphical and text windows of GlancePlus.

Issuing **ps -efP** will show the PRM groups in the **ps** output as shown in the following abbreviated listing:

[View full width]

```
# ps -efP
  UID        PRMID    PID  PPID  C    STIME TTY        TIME COMMAND
root     (PRM_SYS)      0     0  0   Jan 17  ?         0:14 swapper
root     (PRM_SYS)      8     0  0   Jan 17  ?         0:00 supsched
root     (PRM_SYS)      9     0  0   Jan 17  ?         0:00 strmem
root     (PRM_SYS)     10     0  0   Jan 17  ?         0:00 strweld
root     (PRM_SYS)     11     0  0   Jan 17  ?         0:00 strfreebd
root     (PRM_SYS)      2     0  0   Jan 17  ?         0:17 vhand
root     (PRM_SYS)      3     0  0   Jan 17  ?        33:16 statdaemon
root     (PRM_SYS)      4     0  0   Jan 17  ?         0:54 unhashdaemon
root     (PRM_SYS)     12     0  0   Jan 17  ?         0:00 ttisr
root     (PRM_SYS)     13     0  0   Jan 17  ?         0:00 ioconfigd
root     (PRM_SYS)      1     0  0   Jan 17  ?         1:16 init
root     (PRM_SYS)     19     0  0   Jan 17  ?         0:39 lvmkd
root     (PRM_SYS)     20     0  0   Jan 17  ?         0:39 lvmkd
root     (PRM_SYS)     21     0  0   Jan 17  ?         0:40 lvmkd
root     (PRM_SYS)     22     0  0   Jan 17  ?         0:39 lvmkd
root     (PRM_SYS)     23     0  0   Jan 17  ?         0:39 lvmkd
root     (PRM_SYS)     24     0  0   Jan 17  ?         0:40 lvmkd
root     (PRM_SYS)     25     0  0   Jan 17  ?         2:24 smpsched
root     (PRM_SYS)     26     0  0   Jan 17  ?         2:24 smpsched
root     (PRM_SYS)     27     0  0   Jan 17  ?         2:24 smpsched
root     (PRM_SYS)     28     0  0   Jan 17  ?         2:24 smpsched
root     (PRM_SYS)     29     0  0   Jan 17  ?         0:00 sblksched
root     (PRM_SYS)     30     0  0   Jan 17  ?         0:00 sblksched
root     (PRM_SYS)   2281  2170  0   Jan 17  ?         0:00 /usr/dt/bin/dtlogin
root     (PRM_SYS)    661     1  0   Jan 17  ?         0:01 /usr/sbin/biod 16
root     (PRM_SYS)    691     1  0   Jan 17  ?         0:00 /usr/sbin/rpc.lockd
root     (PRM_SYS)     36     0  0   Jan 17  ?        85:50 vxfsd
root     (PRM_SYS)    504     1  0   Jan 17  ?         0:03 /usr/sbin/syslogd -D
root     (PRM_SYS)   1100     1  0   Jan 17  ?        30:51 scrdaemon
root     (PRM_SYS)   1718  1167  0   Jan 17  ?        12:25 psmctd
root     (PRM_SYS)   1109     1  0   Jan 17  ?         0:00 /usr/sbin/rbootd
root     (PRM_SYS)   1724   720  0   Jan 17  ?         0:01 /etc/opt/resmon/lbin/registrar
```

```
  root    (PRM_SYS)   2095     1  0  Jan 17  ?        42:29 dm_ep_engine dwdbp1 hpux10 DM
  root    (PRM_SYS)   1221     1  0  Jan 17  ?         0:00 /opt/perf/bin/ttd
oracle       OTHERS  26435     1  0  Feb 16  ?         0:08 ora_snp0_dwarp1
  root    (PRM_SYS)   1145     1  0  Jan 17  ?         0:17 /usr/sbin/cron
  root    (PRM_SYS)   1170     1  0  Jan 17  ?         0:00 /usr/sbin/envd
oracle       OTHERS  10290     1  0 08:29:42 ?         0:00 oracledwsbp1
(DESCRIPTION=(LO    oracle       OTHERS   2197     1  1  Feb 16  ?         0:17
➥ ora_dbw2_dwmkp1
oracle       OTHERS   2201     1  1  Feb 16  ?         0:14 ora_lgwr_dwmkp1
oracle       OTHERS   2193     1  0  Feb 16  ?         0:18 ora_dbw0_dwmkp1
oracle       OTHERS   2212     1  0  Feb 16  ?         0:07 ora_snp1_dwmkp1
oracle       OTHERS   2215     1  0  Feb 16  ?         0:07 ora_snp2_dwmkp1
  root    (PRM_SYS)  10291   720  0 08:29:55 pts/tc     0:00 telnetd
oracle       OTHERS   2208     1  0  Feb 16  ?         0:00 ora_reco_dwmkp1

                      -

                      .

                      .

 db_hp          db2   5642  1218  0 12:59:24 pts/0      0:00 -su
 db_hp          db2   6683  6607 23 13:11:39 pts/0      0:00 find / -name wlm*
  root    (PRM_SYS)   2043     1  0  Jan 17  ?         0:00 /usr/sbin/rpc.mountd
oracle       OTHERS  26441     1  0  Feb 16  ?         0:07 ora_snp3_dwarp1
   www       OTHERS   2119  2096  0  Jan 17  ?         0:00 /opt/apache/bin/httpd
  root    (PRM_SYS)   2054     1  0  Jan 17  ?         0:00 /usr/sbin/nfsd 16
  root    (PRM_SYS)   2055     1  0  Jan 17  ?         0:47 /usr/sbin/nfsd 16
  root    (PRM_SYS)   2057  2055  0  Jan 17  ?         0:59 /usr/sbin/nfsd 16
  root    (PRM_SYS)   2058  2055  0  Jan 17  ?         1:00 /usr/sbin/nfsd 16
  root    (PRM_SYS)   2059  2055  0  Jan 17  ?         0:47 /usr/sbin/nfsd 16
  root    (PRM_SYS)   2060  2058  0  Jan 17  ?         1:00 /usr/sbin/nfsd 16
  root    (PRM_SYS)   2061  2058  0  Jan 17  ?         1:00 /usr/sbin/nfsd 16
  root    (PRM_SYS)   2062  2055  0  Jan 17  ?         0:47 /usr/sbin/nfsd 16
```

Two processes run by the *PRMID db2* are shown immediately after the three dots. The **find** process has already consumed *23* in the *C* column and could use additional resources if we had not capped it.

We can also view the output of system resources being used by PRM groups with the **prmmonitor** command, as shown in the following output:

```
# prmmonitor

PRM configured from file:  /etc/prmconf.test
File last modified:        Mon Feb 11 12:43:01 2002

HP-UX dwdbp1 B.11.11 U 9000/800    02/19/02

Tue Feb 19 13:24:18 2002    Sample:  1 second
CPU scheduler state:  Enabled, CPU cap ON
                                                CPU      CPU
PRM Group                         PRMID   Entitlement     Used
_____________________________________________________________
OTHERS                                1        47.62%   13.15%
DW                                    2        47.62%    0.00%
db2                                   3         4.76%    4.46%

PRM application manager state:  Enabled  (polling interval: 30 seconds)
```

This output shows that *db2* consumes very close to its *CPU Entitlement* when running the job. Note that the CPU scheduler state shows *CPU cap ON*. Capping is controlled at the command line with the following two commands to turn capping on and off, respectively:

```
# prmconfig -M CPUCAPON


# prmconfig -M CPUCAPOFF
```

Capping can also be controlled with the GUI shown in the upcoming figures.

We could have made all of the modifications to **/etc/prmconf** and the other PRM-related work using **xprm**. Figure 18-7 shows the **/etc/prmconf** file from our example and some of the modifications we could make using **xprm**:

Figure 18-7. xprm and Configuration Data

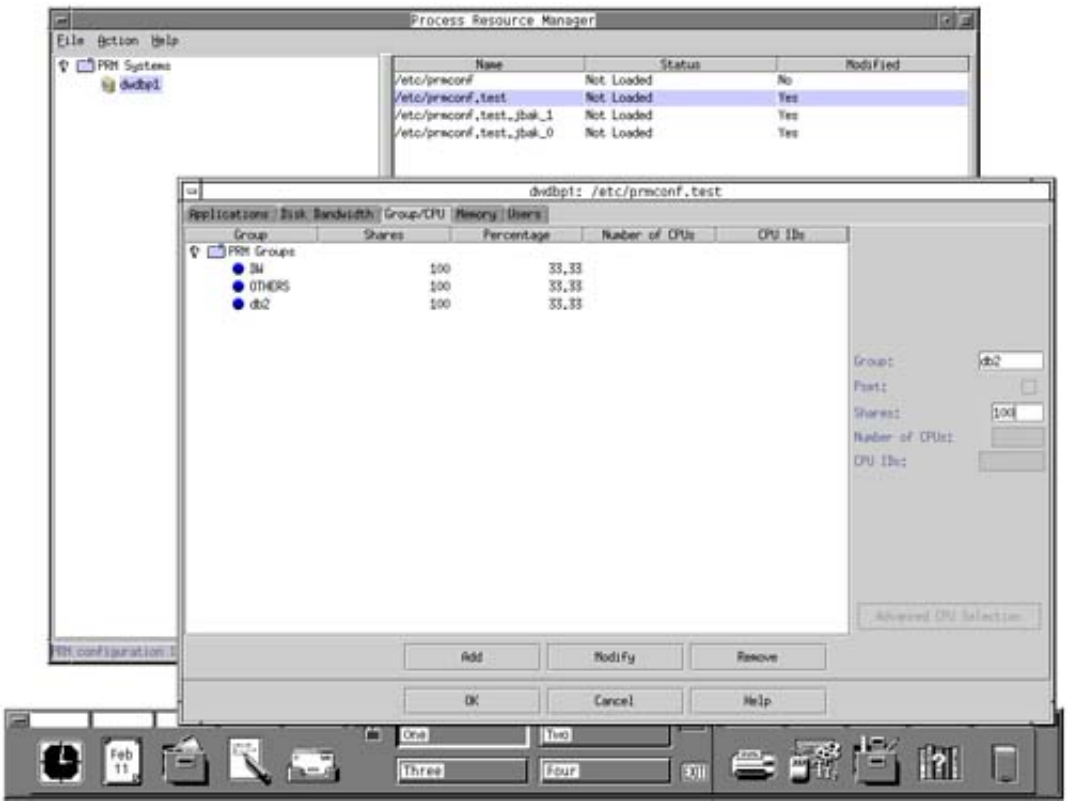
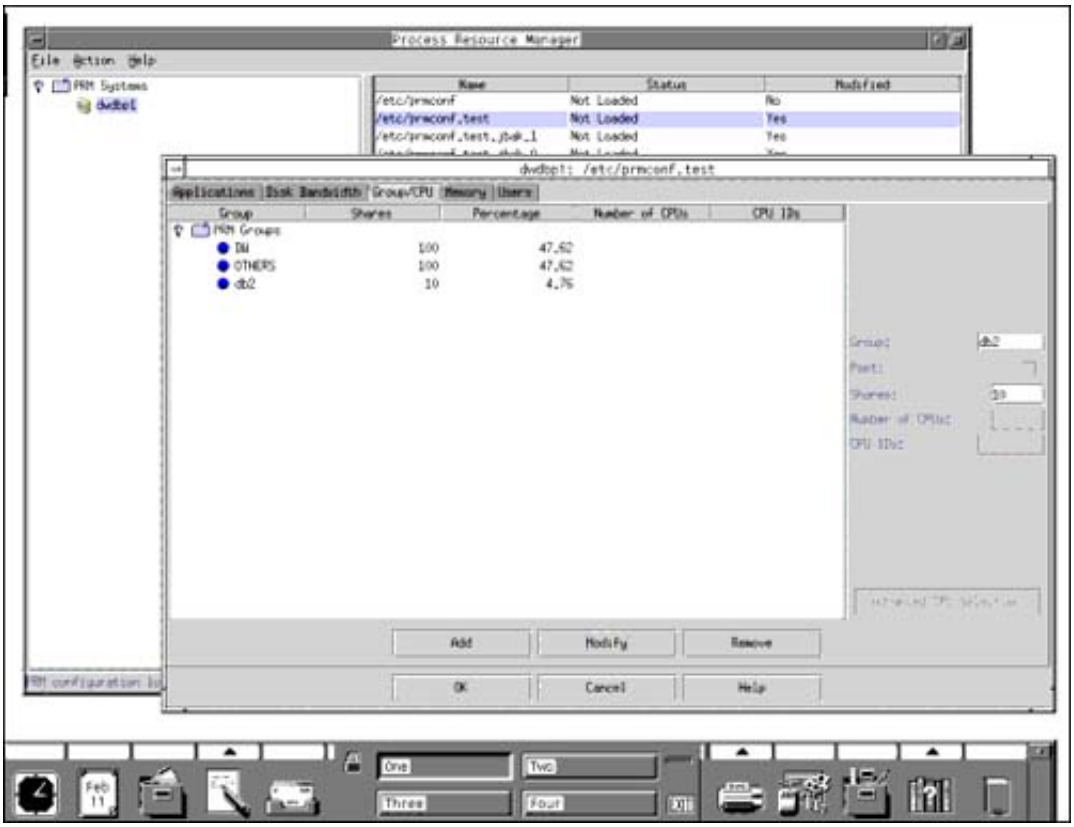


Figure 18-7 shows our original shares for the three PRM groups. In the right-hand side of the lower window we could modify the shares and cause them to take effect immediately by hitting the *Modify* button, as shown in Figure 18-8:

Figure 18-8. xprm and Configuration Share Changed from *100* to *10*



Through the **xprm** interface we can perform all PRM-related work; however, knowing the command-line method of working with PRM adds additional insight.

Keep in mind that the management of the *(PRM_SYS)* group is handled for us automatically.

The capping feature has many applications on enterprise systems, such as the case where users are funding a specific percentage of a system and will be allocated the same percentage of system resources that they are funding.

If you were to turn off the capping feature then the limits specified for the PRM groups would be employed only at times of peak system load. This is covered in the next section.

PRM Setup with xprm

This section describes the way in which you would configure PRM to control the amount of resources that processes use during peak system load. Using PRM, you can guarantee a minimum allocation of system resources at peak load. All of the prerequisites to PRM, such as installing and configuring the software covered earlier, must be done prior to performing the PRM-related work in this section.

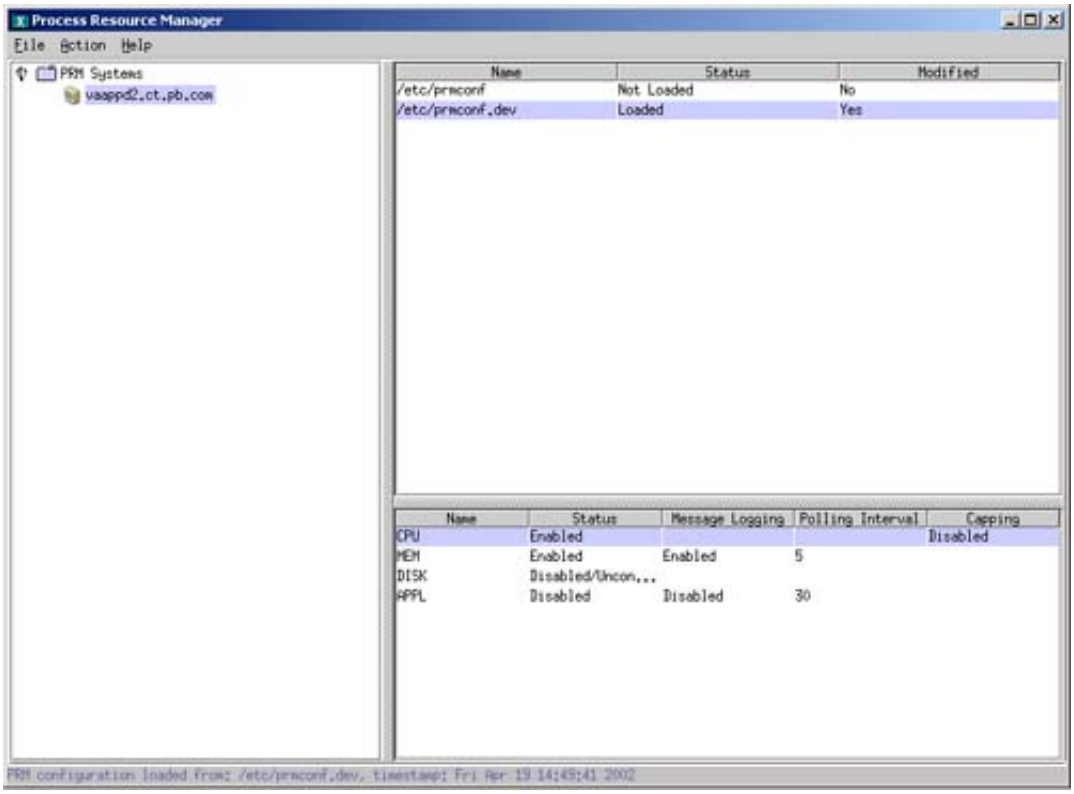
This procedure is nearly identical to the resource cap procedure covered in the previous section. In this procedure, however, we do not enable caps, thereby resulting in the restriction on system resources taking place only during system load. In the example in this section we'll use the PRM GUI (**xprm**) to perform configuration.

The key component to guaranteeing system resources is the PRM group described earlier. The example in this section will define PRM groups and their respective shares using the Fair Share Scheduler (FSS) and not Processor Sets (PSETs).

We have two development groups sharing this system. We'll create two PRM groups under the name of their respective development applications. We'll then specify shares for these two groups based on the amount of system resources we want them to consume at the time of peak system load.

We'll first invoke the PRM GUI with the **xprm** command and copy the default PRM configuration file **/etc/prmconf** to **/etc/prmconf.dev** as shown in Figure 18-9:

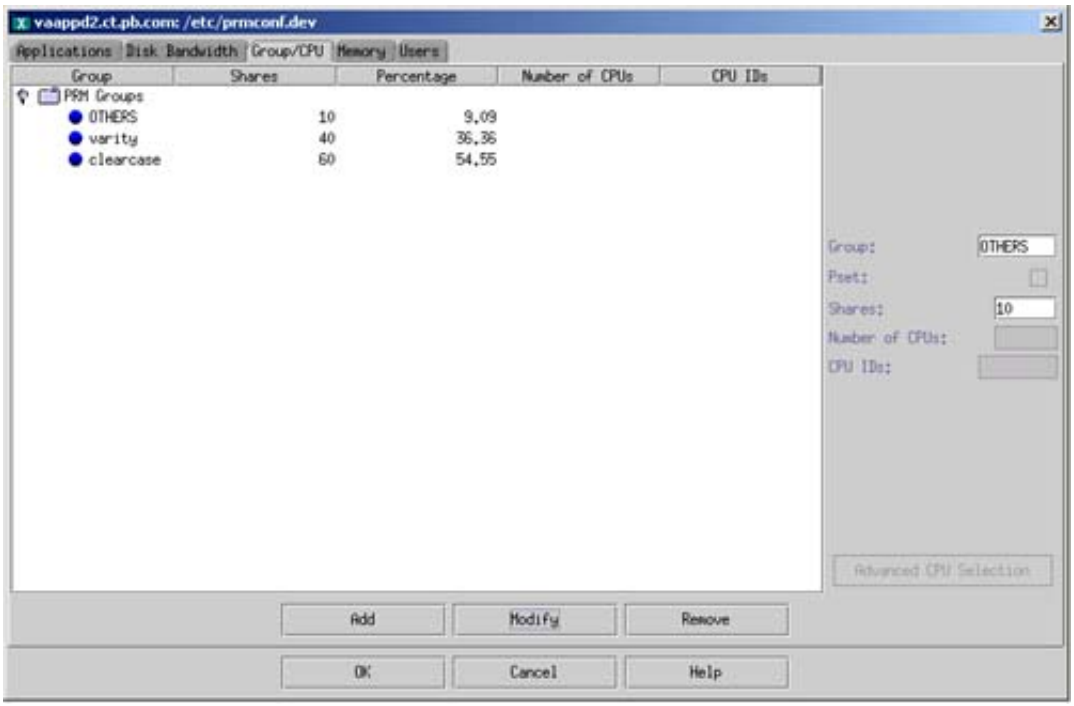
Figure 18-9. xprm With the Two Configuration Files



The **/etc/prmconf.dev** file that we'll be working with was created by copying it in **xprm**. When we double-click **prmconf.dev** an additional window opens up that allows us to specify all information related to our configuration. Notice in this diagram that *Capping* is *Disabled* so that the entitlements we specify will be employed only at the time of peak system use and not as a general cap as described earlier.

We'll first create our PRM groups and the amount of CPU we'd like each group to consume during peak system load, as shown in Figure 18-10:

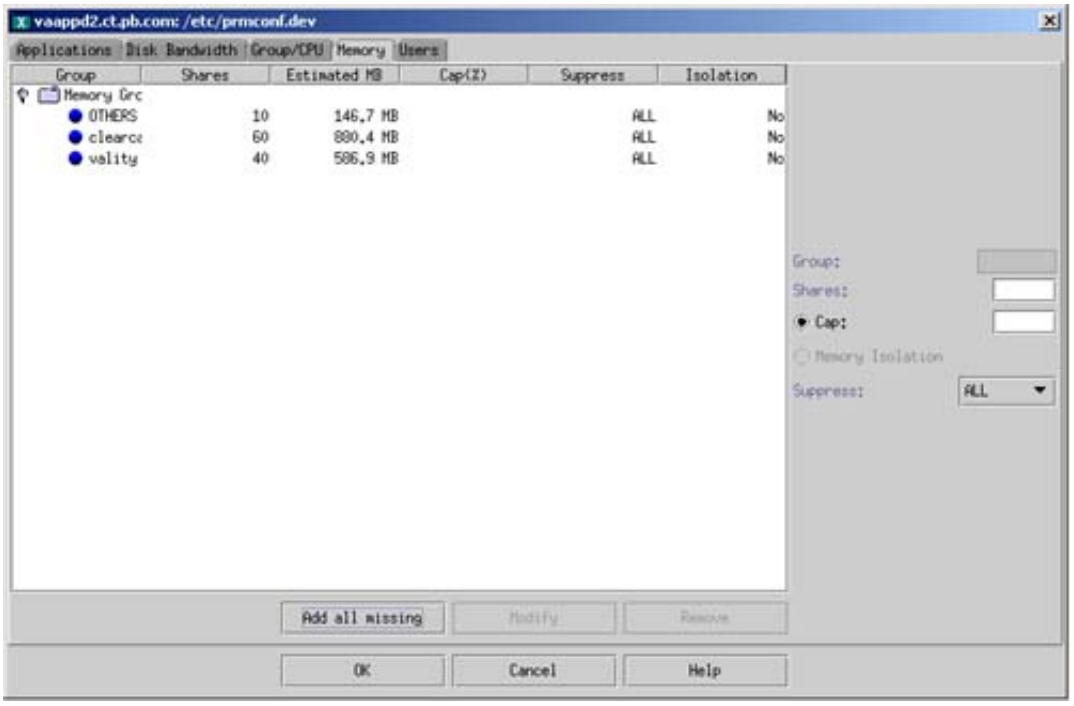
Figure 18-10. xprm With Groups and Amount of CPU Resources for Each



The *OTHERS* group was created by default and we created the *vality* and *clearcase* groups. In addition, we created the shares shown for each of the groups with *clearcase* having the largest share.

Next, we configure the memory shares for each group as shown in Figure 18-11:

Figure 18-11. xprm With Groups and Amount of Memory Resources for Each

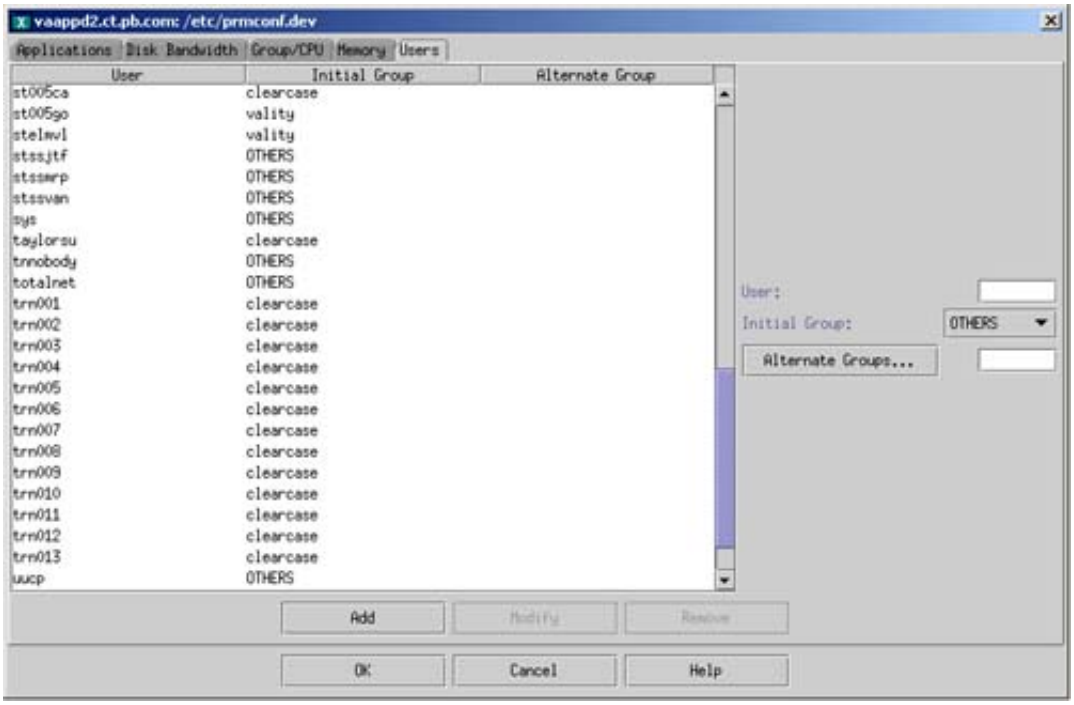


The memory shares shown in Figure 18-11 are the same as those earlier created for CPU. Notice that in the right of this diagram is the option to employ a cap. We will not make this selection so that our limits on

system resources will be employed only at times of peak system load. The *Cap %* column is blank indicating that we have not employed caps.

Next we'll run the **prmloadconf** command to load all users from **/etc/passwd** into the **/etc/prmconf.dev** with which we're working. We can then specify the PRM groups with which the users will be associated, as shown in Figure 18-12:

Figure 18-12. xprm with Users and Their Associated PRM Groups

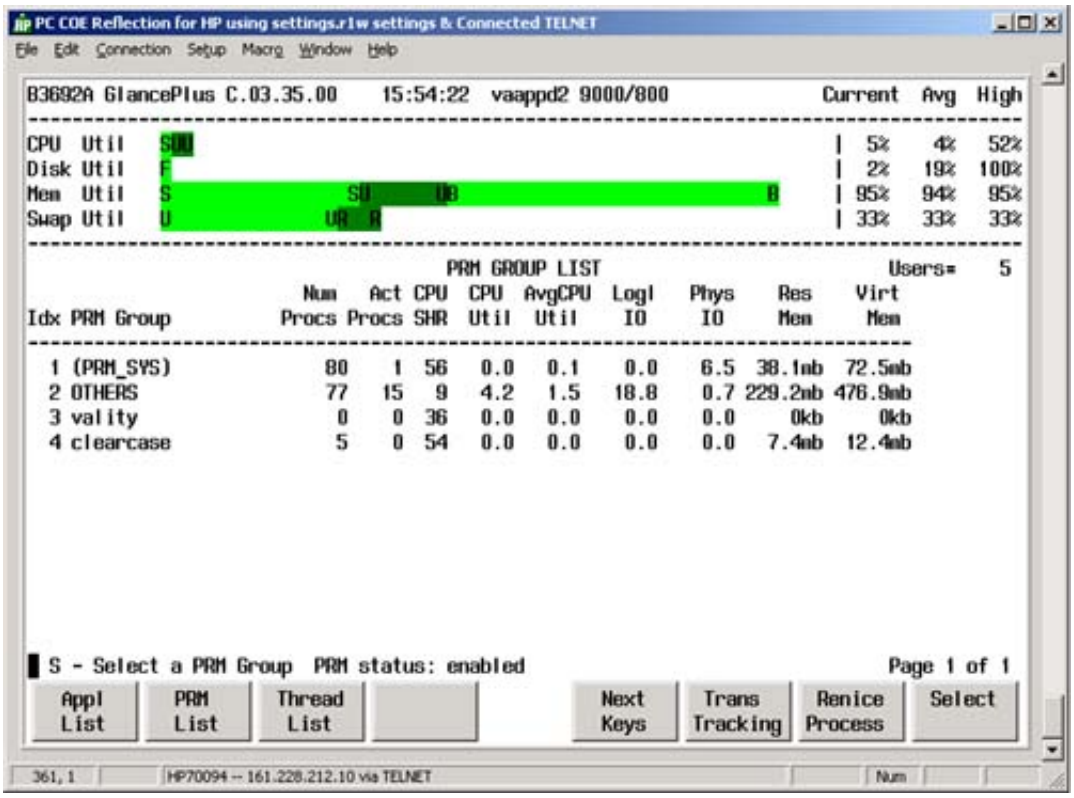


Most of the users on this system are in the *clearcase* PRM group, but when *vality* users start large sorts, all of the system resources are consumed by the sorts, and *clearcase* users get minimal system resources.

The configuration is now complete, so we go back to the first **xprm** screen and enable **/etc/prmconf.dev**.

We can use GlancePlus to view our PRM groups and see if our groups have been created, as shown in Figure 18-13:

Figure 18-13. xprm With Users and Their Associated PRM Groups



The PRM groups we configured as well as the CPU shares (*CPU SHR*) are shown in the figure. During times of peak system load, the PRM groups will be limited to the amount of resources we have specified.

Although all of the work was done in **xprm** the **/etc/prmconf.dev** file was updated when we made and saved our changes. The following listing shows part of the **/etc/prmconf.dev** file with our configuration, with some of the users not shown because there are many users on the system:

```
#
# Group/CPU records
#
OTHERS:1:10::
clearcase:3:60::
vality:2:40::
#
# Memory records
#
#!PRM_MEM:1:10::::
#!PRM_MEM:3:60::::
#!PRM_MEM:2:40::::
#
# Application records
#
#
# Disk bandwidth records
#
#
# User records
#
adm:::OTHERS
an001va::::clearcase
best1::::OTHERS
bin::::OTHERS
bo001ya::::clearcase
br001ca::::clearcase
br002no::::clearcase
careyji::::clearcase
ccalbd::::OTHERS
daemon::::OTHERS
do004sh::::clearcase
ga507ch::::clearcase
```

```
gabapr::::clearcase
hpdb::::OTHERS
itvobadm::::clearcase
janel::::clearcase
janelle::::clearcase
ju001ya::::clearcase
ke004do::::clearcase
kemekda::::clearcase
kiesibw::::OTHERS
leema::::clearcase
lp::::OTHERS
ma002we::::clearcase
ma003ch::::clearcase
maji::::clearcase
nemapa::::clearcase
nobody::::OTHERS
nuucp::::OTHERS
opc_op::::OTHERS
pursnani::::clearcase
root::::OTHERS
sa002ab::::clearcase
smbnull::::OTHERS
st002st::::clearcase
st005ca::::clearcase
st005go::::vality
stelmvl::::vality
stssjtf::::OTHERS
stssmrp::::OTHERS
stssvan::::OTHERS
sys::::OTHERS
taylorsu::::clearcase
tnnobody::::OTHERS
totalnet::::OTHERS
trn001::::clearcase
trn002::::clearcase
trn003::::clearcase
trn004::::clearcase
trn005::::clearcase
trn006::::clearcase
trn007::::clearcase
trn008::::clearcase
trn009::::clearcase
trn010::::clearcase
trn011::::clearcase
trn012::::clearcase
trn013::::clearcase
uucp::::OTHERS
vality::::vality
webadmin::::OTHERS
clearww::::OTHERS
root@vaappd2[/etc] >
```

We can also view our work by issuing the **prmlist** command to get a summary of our configuration:

```
root@vaappd2[/etc] > prmlist

PRM configured from file:  /etc/prmconf.dev
File last modified:        Fri Apr 19 15:35:55 2002

PRM Group                      PRMID    CPU Entitlement
-------------------------------------------------------
OTHERS                             1             9.09%
clearcase                          3            54.55%
vality                             2            36.36%

                                        Memory  Upper
PRM Group                      PRMID  Entitlement  Bound  Suppress
-----------------------------------------------------------------
OTHERS                             1        9.09%              ALL
clearcase                          3       54.55%              ALL
vality                             2       36.36%              ALL

PRM User                       Initial Group                 Alternate Group(s)
-------------------------------------------------------------------------------
adm                            OTHERS
```

```
an001va                         clearcase
best1                           OTHERS
bin                             OTHERS
bo001ya                         clearcase
br001ca                         clearcase
br002no                         clearcase
careyji                         clearcase
ccalbd                          OTHERS
daemon                          OTHERS
do004sh                         clearcase
ga507ch                         clearcase
gabapr                          clearcase
hpdb                            OTHERS
itvobadm                        clearcase
janel                           clearcase
janelle                         clearcase
ju001ya                         clearcase
ke004do                         clearcase
kemekda                         clearcase
kiesibw                         OTHERS
leema                           clearcase
lp                              OTHERS
ma002we                         clearcase
ma003ch                         clearcase
maji                            clearcase
nemapa                          clearcase
nobody                          OTHERS
nuucp                           OTHERS
opc_op                          OTHERS
pursnani                        clearcase
root                            OTHERS
sa002ab                         clearcase
smbnull                         OTHERS
st002st                         clearcase
st005ca                         clearcase
st005go                         vality
stelmvl                         vality
stssjtf                         OTHERS
stssmrp                         OTHERS
stssvan                         OTHERS
sys                             OTHERS
taylorsu                        clearcase
tnnobody                        OTHERS
totalnet                        OTHERS
trn001                          clearcase
trn002                          clearcase
trn003                          clearcase
trn004                          clearcase
trn005                          clearcase
trn006                          clearcase
trn007                          clearcase
trn008                          clearcase
trn009                          clearcase
trn010                          clearcase
trn011                          clearcase
trn012                          clearcase
trn013                          clearcase
uucp                            OTHERS
vality                          vality
webadmin                        OTHERS
www                             OTHERS
root@vaappd2[/etc] >
```

All of the processes on this system should be in either the PRM group *OTHERS* or *PRM_SYS* since we did not specify another PRM groups to which processes would be assigned. The following truncated **ps -efP** listing shows the *PRMID* of processes:

```
root@vaappd2[/etc] > ps -efP
     UID        PRMID    PID  PPID  C    STIME TTY       TIME COMMAND
    root    (PRM_SYS)      0     0  0  Apr  9  ?         0:04 swapper
    root    (PRM_SYS)      8     0  0  Apr  9  ?         0:00 supsched
    root    (PRM_SYS)      9     0  0  Apr  9  ?         0:00 strmem
    root    (PRM_SYS)     10     0  0  Apr  9  ?         0:00 strweld
    root    (PRM_SYS)     11     0  0  Apr  9  ?         0:00 strfreebd
    root    (PRM_SYS)      2     0  0  Apr  9  ?         0:22 vhand
```

```
    root    (PRM_SYS)      3     0  0  Apr  9  ?        13:37 statdaemon
    root    (PRM_SYS)      4     0  0  Apr  9  ?         0:11 unhashdaemon
    root    (PRM_SYS)     12     0  0  Apr  9  ?         0:00 ttisr
    root       OTHERS    383     1  0  Apr  9  ?         0:43 vpard
    root    (PRM_SYS)     14     0  0  Apr  9  ?         0:00 ioconfigd
    root    (PRM_SYS)      1     0  0  Apr  9  ?         0:05 init
    root    (PRM_SYS)     20     0  0  Apr  9  ?         0:51 lvmkd
    root    (PRM_SYS)     21     0  0  Apr  9  ?         0:51 lvmkd
    root    (PRM_SYS)     22     0  0  Apr  9  ?         0:52 lvmkd
    root    (PRM_SYS)     23     0  0  Apr  9  ?         0:52 lvmkd
    root    (PRM_SYS)     24     0  0  Apr  9  ?         0:52 lvmkd
    root    (PRM_SYS)     25     0  0  Apr  9  ?         0:51 lvmkd
    root    (PRM_SYS)     26     0  0  Apr  9  ?         0:02 smpsched
    root    (PRM_SYS)     27     0  0  Apr  9  ?         0:02 smpsched
    root    (PRM_SYS)     28     0  0  Apr  9  ?         0:02 smpsched
    root    (PRM_SYS)     29     0  0  Apr  9  ?         0:00 sblksched
    root    (PRM_SYS)     30     0  0  Apr  9  ?         0:00 sblksched
    root       OTHERS   1793     1  0  Apr  9  console   0:00 /usr/sbin/getty console
    root       OTHERS   1593  1248  0  Apr  9  ?         0:13 diaglogd
    root    (PRM_SYS)     36     0  0  Apr  9  ?        11:32 vxfsd
    root       OTHERS    377     1  0  Apr  9  ?         0:31 /usr/sbin/syncer
    root    (PRM_SYS)    570     0  0  Apr  9  ?         0:00 nfskd
    root       OTHERS    380     1  0  Apr  9  ?         0:00 vphbd -d 360 -p vph
bd.pid

                        -
                        -
                        -

    root       OTHERS   3693  3368  0 13:32:36 pts/tc    0:00 script
    root       OTHERS   4156  1521  0 14:14:08 ?         0:00 /usr/local/sbin/sshd
    root       OTHERS   4562     1  0 14:40:39 ?         0:05 /opt/prm/bin/prm2d
    root       OTHERS   3368  3367  0 12:51:57 pts/tc    0:00 -sh
    root       OTHERS  19232     1  0  Apr 10  ?         6:40 /usr/bin/dsmc sched
    root       OTHERS   5233  4651  3 15:44:32 pts/td    0:00 ps -efP
    root       OTHERS  19235     1  0  Apr 10  ?         0:00 /usr/bin/dsmcad
root@vaappd2[/etc] >
```

The **prmmonitor** can also be used at the command line to show the entitlements and amount of resources used as shown in the following listing:

```
root@vaappd2[/etc] >   prmmonitor

PRM configured from file:  /etc/prmconf.dev
File last modified:        Fri Apr 19 15:35:55 2002

HP-UX vaappd2 B.11.11 U 9000/800    04/19/02

Fri Apr 19 15:50:14 2002    Sample:  1 second
CPU scheduler state:  Enabled
                                                CPU      CPU
PRM Group                         PRMID   Entitlement     Used
_____________________________________________________________
OTHERS                                1         9.09%    5.45%
vality                                2        36.36%    0.00%
clearcase                             3        54.55%    0.00%

Fri Apr 19 15:50:14 2002    Sample:  1 second
Memory manager state:  Enabled    Paging:  No

                                       Memory  Upper
PRM Group                        PRMID  Entitlement  Bound    Usage   Procs  Stops
_____________________________________________________________________________
OTHERS                               1        9.09%          10.39%      62
vality                               2       36.36%           0.00%       0
clearcase                            3       54.55%           0.35%       5

PRM application manager state:  Disabled  (polling interval: 30 seconds)

root@vaappd2[/etc] >
```

Please keep in mind that the entitlements shown are enforced only at the time of peak system load so that PRM groups can exceed their entitlement, such as the *OTHERS* memory usage shown in the previous listing, if indeed the system is not at peak load and resources are available.

The procedure covered in this section is very similar to the "resource cap" procedure in the previous section, except that the **xprm** GUI was used for the configuration and that caps were not employed as part of this procedure. The process covered in this section is common on systems where peak load is often encountered and system administrators want to guarantee system resources for all users.

[Team LiB]

PRM and Oracle Setup

In this section we'll configure PRM on a system running five instances of Oracle. There are many processes running for each Oracle instance but there is a unique identifier for each instance that uses the *ORACLE_SID* environment variable. On the system used in this example the five Oracle instances are: *dwsbp1, dwarp1, dwmkp1, hrprd1* and *dwcrtp1*. We'll use these instance names during our configuration.

The first step we'll perform using **xprm**, the graphical interface of PRM, is to create PRM groups and assign shares. We have only four CPUs in this Superdome nPartition so we'll use only FSS and not PSET PRM groups as shown in Figure 18-14:

Figure 18-14. xprm Showing PRM Groups and FSS Shares

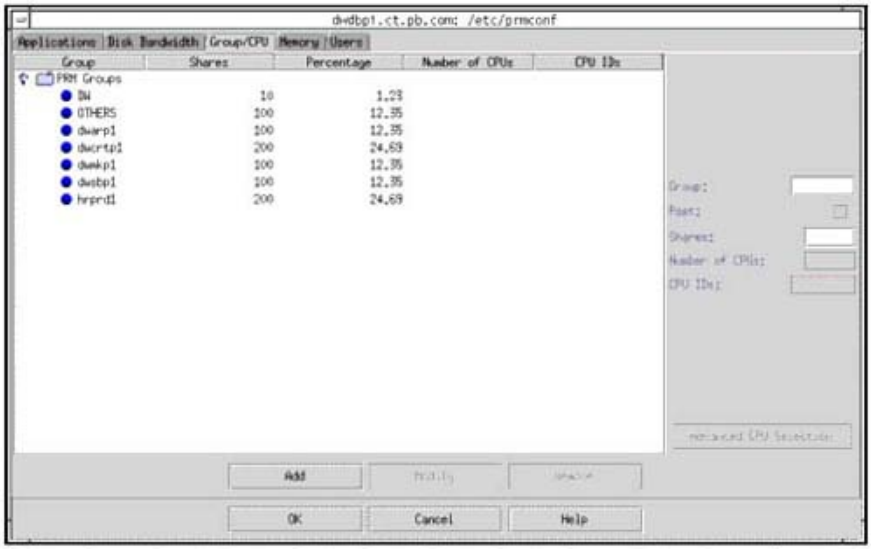
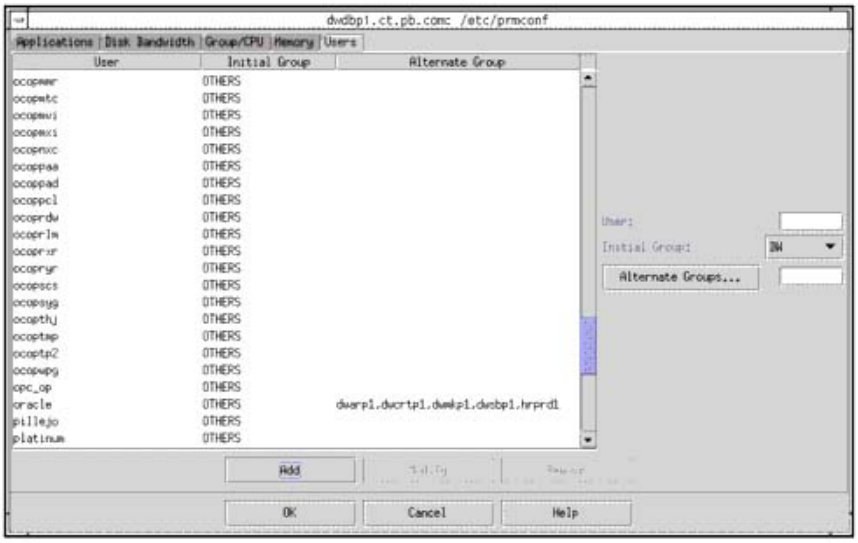


Figure 18-14 shows that we have a total of seven PRM groups. The bottom five shown in the figure are Oracle database instances. Two of the seven groups are more important than the others, so we've assigned higher shares to those two groups. The *DW* group is not important, so we've assigned a small share to it.

We haven't set up any *Memory* shares but that would be easy to do by selecting the *Memory* tab in the figure and assigned shares to the groups.

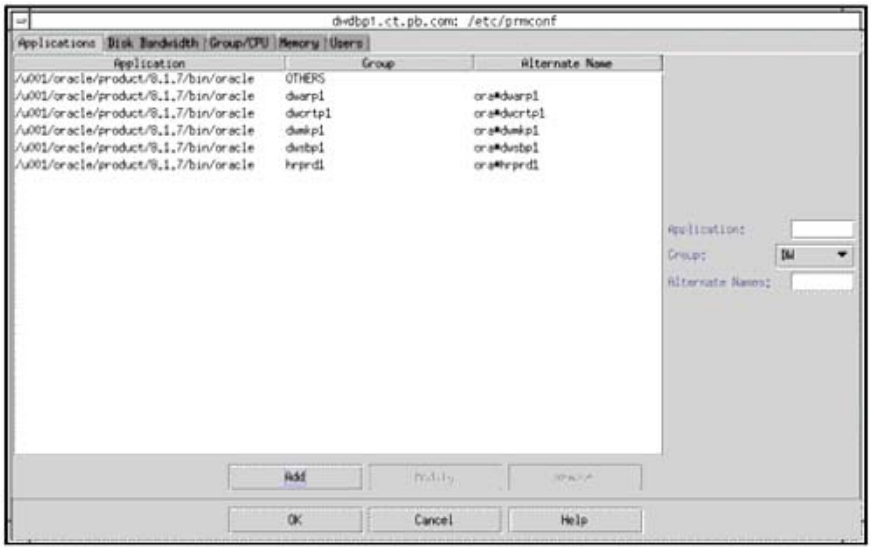
Next we'll set up the users. The user *oracle* is already in the *OTHERS* group and we specify the *Alternate Group* for *oracle* as the five database instances, as shown in Figure 18-15:

Figure 18-15. User *oracle* and Its *Alternate Groups*



Next we'll define the applications and the corresponding PRM groups to which they'll be assigned. Figure 18-16 shows the *Applications* tab:

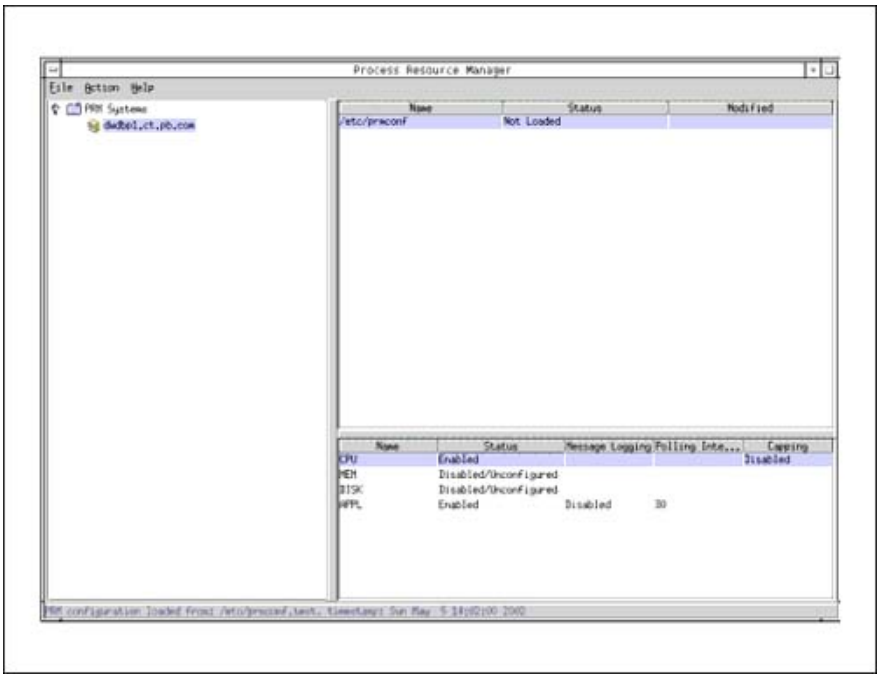
Figure 18-16. xprm and *Application*



We now have our five instances of Oracle set up such that the unique name for each instance will result in processes getting the assigned resources by the application manager.

Figure 18-17 shows a summary of what we've configured in **xprm**:

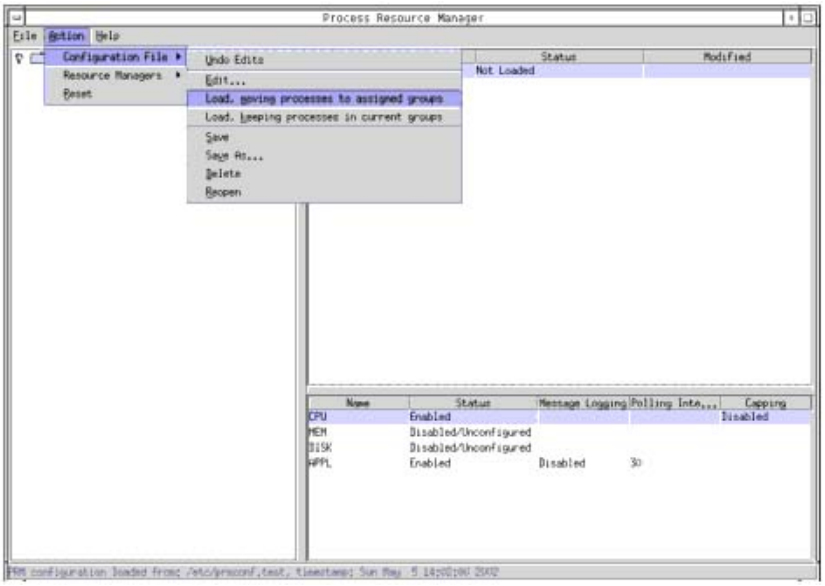
Figure 18-17. xprm for Oracle



The figure shows that *CPU* and *APPL* have been *Enabled*.

Now that we've performed the customization we want, we can apply the settings as shown Figure 18-18:

Figure 18-18. xprm Load Configuration



Now we can proceed to see that our configuration was loaded by running **prmlist**, as shown in the following abbreviated listing:

```
root@dwdbp1[/etc] > prmlist

PRM configured from file:  /etc/prmconf
File last modified:        Fri May 24 15:20:42 2002

PRM Group                       PRMID    CPU Entitlement
-----------------------------------------------------------
DW                                  2                 1%
OTHERS                              1             11.11%
dwarp1                              5             11.11%
dwcrtp1                             4             22.22%
dwmkp1                              6             11.11%
dwsbp1                              3             11.11%
hrprd1                              7             22.22%
```

```
PRM User                        Initial Group                  Alternate Group(s)
-------------------------------------------------------------------------------
adm                             OTHERS
bbuser                          OTHERS
best1                           OTHERS
bh001pa                         OTHERS
bin                             OTHERS
certegy                         OTHERS
cognos                          OTHERS
ctrapp                          OTHERS
daemon                          OTHERS
db_hp                           OTHERS
dbadict                         OTHERS
ddbadcb                         OTHERS
ddbajms                         OTHERS
ddbarcl                         OTHERS

                    -
                    .
                    .

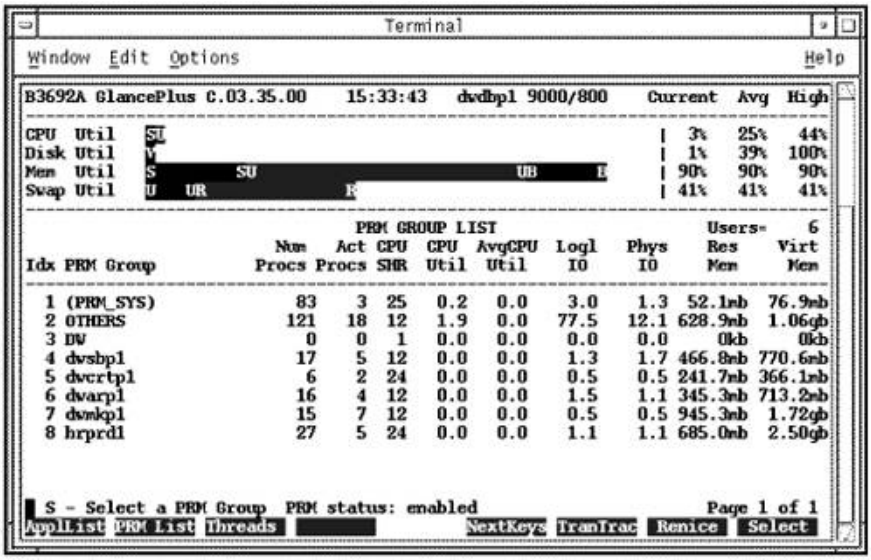
ocoptp2                         OTHERS
ocopwpg                         OTHERS
opc_op                          OTHERS
oracle                        OTHERS                       dwarp1,dwcrtp1,dwmkp1,dwsbp1
uucp                            OTHERS
webadmin                        OTHERS
www                             OTHERS
zeke                            OTHERS

PRM Application                 Assigned Group                 Alternate Name(s)
-------------------------------------------------------------------------------
/u001/oracle/product/8.1.7/bin OTHERS
/u001/oracle/product/8.1.7/bin dwarp1                          ora*dwarp1
/u001/oracle/product/8.1.7/bin dwcrtp1                         ora*dwcrtp1
/u001/oracle/product/8.1.7/bin dwmkp1                          ora*dwmkp1
/u001/oracle/product/8.1.7/bin dwsbp1                          ora*dwsbp1
/u001/oracle/product/8.1.7/bin hrprd1                          ora*hrprd1
root@dwdbp1[/etc] >
```

This listing shows the assigned groups and names at the end that we entered using **xprm**.

Next we'll run Glance and look at the PRM Groups to see if they appear as shown in Figure 18-19:

Figure 18-19. GlancePlus/UX Showing PRM Groups



This screen shot shows the PRM Groups we set up as well as their respective shares with *DW* having the lowest share value. The only activity now taking place on the system is in *OTHERS,* which is the group to which *root* was moved from *PRM_SYS*.

We can use **prmmonitor** to view PRM-related resources as well, as shown in the following abbreviated listing:

[View full width]

```
root@dwdbp1[/etc] > prmmonitor
```

```
PRM configured from file:  /etc/prmconf
File last modified:        Fri May 24 15:24:28 2002

HP-UX dwdbp1 B.11.11 U 9000/800    05/24/02

Fri May 24 15:37:24 2002    Sample:  1 second
CPU scheduler state:  Enabled
                                                CPU       CPU
PRM Group                        PRMID    Entitlement     Used
_____________________________________________________________
OTHERS                               1         12.34%    3.20%
DW                                   2          1.26%    0.00%
dwsbp1                               3         12.34%    0.00%
dwcrtp1                              4         24.69%    0.00%
dwarp1                               5         12.34%    0.00%
dwmkp1                               6         12.34%    0.00%
hrprd1                               7         24.69%    0.00%

PRM application manager state:  Enabled  (polling interval: 30 seconds)

root@dwdbp1[/etc] >
```

Now that we are satisfied with this configuration we can use the *Action->Save* menu pick in **xprm** to save our work **/etc/prmconf**. The following is an abbreviated listing of **prmconf** with our changes:

```
root@dwdbp1[/etc] > more /etc/prmconf
#
# Group/CPU records
#
DW:2:10::
OTHERS:1:100::
dwarp1:5:100::
dwcrtp1:4:200::
dwmkp1:6:100::
dwsbp1:3:100::
hrprd1:7:200::
#
# Memory records
#
#
# Application records
#
/u001/oracle/product/8.1.7/bin/oracle:::OTHERS
/u001/oracle/product/8.1.7/bin/oracle:::dwarp1,ora*dwarp1
/u001/oracle/product/8.1.7/bin/oracle:::dwcrtp1,ora*dwcrtp1
/u001/oracle/product/8.1.7/bin/oracle:::dwmkp1,ora*dwmkp1
/u001/oracle/product/8.1.7/bin/oracle:::dwsbp1,ora*dwsbp1
/u001/oracle/product/8.1.7/bin/oracle:::hrprd1,ora*hrprd1
#
# Disk bandwidth records
#
#
# User records
#
adm:::OTHERS
bbuser:::OTHERS
best1:::OTHERS
bh001pa:::OTHERS
bin:::OTHERS
certegy:::OTHERS
cognos:::OTHERS
ctrapp:::OTHERS
daemon:::OTHERS
db_hp:::OTHERS
dbadict:::OTHERS
ddbadcb:::OTHERS
ddbajms:::OTHERS
ddbarcl:::OTHERS
ddbarjc:::OTHERS
ddbatgf:::OTHERS
dfoareg:::OTHERS
dmisaxa:::OTHERS
                  -
                  .
                  .
ocopwpg:::OTHERS
opc_op:::OTHERS
oracle:::OTHERS,dwarp1,dwcrtp1,dwmkp1,dwsbp1,hrprd1
pillejo:::OTHERS
platinum:::OTHERS
precise:::OTHERS
root:::OTHERS
testing:::OTHERS
trnchess:::OTHERS
uucp:::OTHERS
webadmin:::OTHERS
www:::OTHERS
zeke:::OTHERS
root@dwdbp1[/etc] >
```

The system is now ready for users. After Oracle users are on the system we can monitor activity and make adjustments to the

shares as we see fit.

All of the tasks performed in this Oracle section could have been performed manually as shown in the first section. Either way the same goal is accomplished with PRM providing control over the resources given to the individual instances of Oracle.



Chapter 19. MC/ServiceGuard

About MC/ServiceGuard

MC/ServiceGuard Background

A Simple Example





About MC/ServiceGuard

MC/ServiceGuard is a High Availability (HA) application. It is used in conjunction with other HA techniques to produce a loosely coupled cluster. An application running on a primary system is run on a backup system should a failure or other switch over condition exist on the primary system. MC/ServiceGuard can detect failures of a CPU, LAN, application, and other components.

All of the systems as well as all of the application packages have IP addresses. When a failure occurs the backup system runs the high availability package and the IP address of the package moves with the package to the backup system. If only a LAN card fails in a primary system ServiceGuard will fail over to a standby LAN card rather than the backup system.

You can have many systems in a ServiceGuard cluster, all of which can be active. You don't need to have systems in a standby state waiting for a failure to occur.

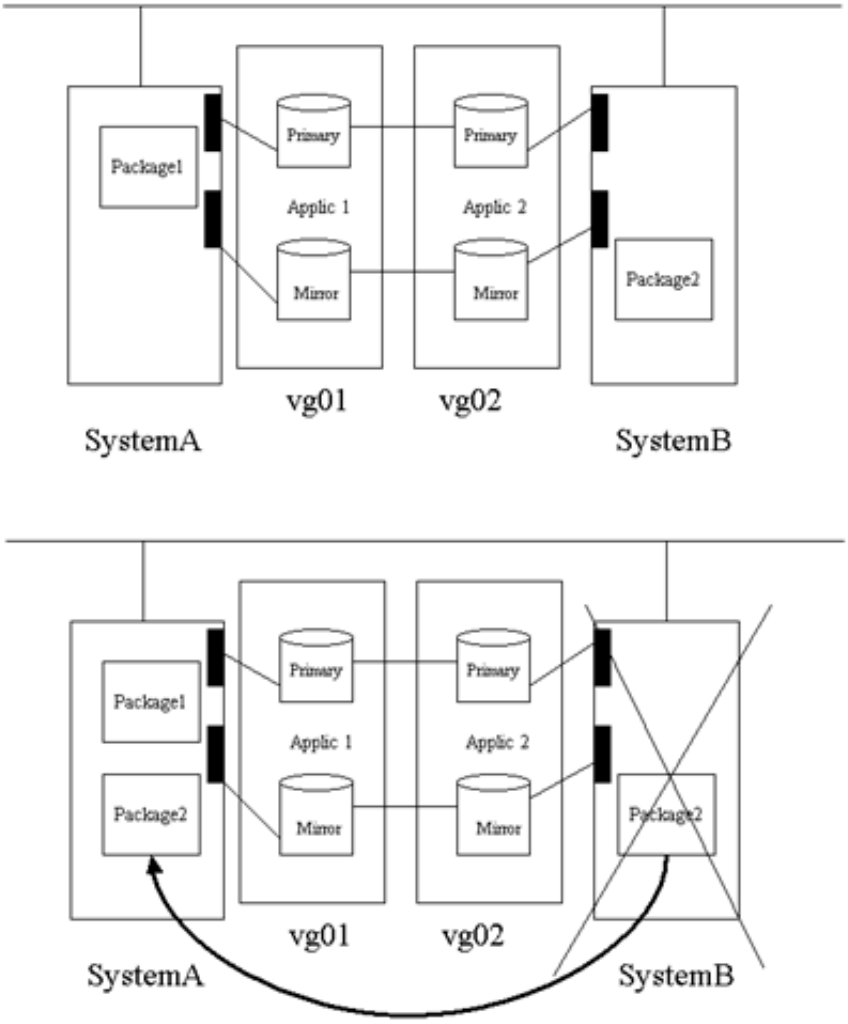
In the upcoming sections we'll cover ServiceGuard background and then setup a simple ServiceGuard cluster and application from one of the HP Education Center training courses.



MC/ServiceGuard Background

Figure 19-1 shows a basic ServiceGuard setup in which two systems both connect to *vg01* and *vg02*. There are package definitions on the respective systems for the applications they are running in the top of the diagram. The bottom of Figure 19-1 shows that *package2* is started on *systemA* when there is a failure on *systemB*. This is a cluster reformation when a failure occurs. This diagram depicts the fundamental operation of ServiceGuard; that is, an application on a failed system is started on a backup system.

Figure 19-1. *Top*: ServiceGuard Cluster; *Bottom*: Cluster Reformation

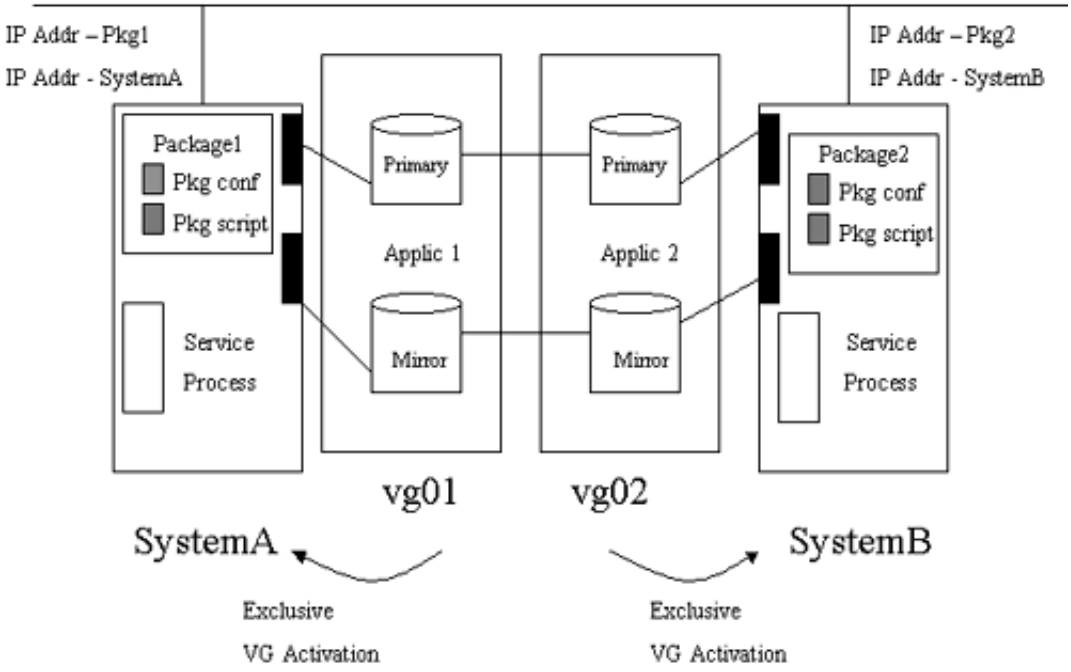


The application package contains all of the information related to applications in the ServiceGuard environment. For each package there is a *package control file* and *package control script*. The package configuration file contains all dependencies for the application such as the package name, service name, subnet, and so on. The package control script is run to start or stop the application package. These are ASCII files.

In addition to the ASCII files used to define the package and start and stop it, there is a *cluster configuration file*. This file has in it parameters such as the cluster name, maximum number of configured packages, and so on. This file, along with the *package configuration file*, are combined to produce a binary file that is distributed to all of the nodes. The **cmapplyconf** command is used to produce the binary file.

Figure 19-2 shows the relationship between some of the components and files in a ServiceGuard environment.

Figure 19-2. MC/ServiceGuard Components



This diagram is by no means complete, but it shows the relationship between some of the more important components in a ServiceGuard environment. The next section covers creating a simple ServiceGuard cluster including the steps required and the files related to the cluster.

A Simple Example

Let's go through a simple ServiceGuard example. Although this example is simple it demonstrates all of the fundamental steps to getting a ServiceGuard cluster set up and an application package running. This example is derived from some labs that are part of the HP ServiceGuard training. If you think ServiceGuard applies to your environment after reviewing this example I would recommend taking the ServiceGuard I and II courses to gain a thorough understanding of ServiceGuard. The example consists of the following three high-level steps:

1. Create a shared volume group **vg01**.
2. Create a ServiceGuard Cluster
3. Create an application package.

Let's now proceed to the first of the three steps and create a shared volume group.

Setup a Shared Volume Group

Our goal in this section is to have a shared volume group that both of the systems in our ServiceGuard cluster can access. Each of the two systems starts with HP-UX installed and a volume group **vg00** that is exclusive to each system; that is, it is not shared. We'll name our shared volume group **vg01**.

vg01 will consist of two disks with mirrored data to provide data protection. The disks are also on two different buses to provide protection against a bus failure. Background on the logical volume manager commands we issue to create the shared volume group are covered in Chapter 3, so I won't provide a lot of comments along with the following steps to create and share **vg01**. The two systems used throughout the upcoming examples have long system names, so I just use a prompt with *ny3c15* for the first system and *ny3c16* for the second system in the cluster.

The first step performed is to create **vg01**. Next we run **ioscan** to view all of the disks on the system. We then run **pvdisplay** to see which disk is used by **vg00** so that we don't use it for **vg01**. After selecting the two disks that we want to be in **vg00**, we run **pvcreate** on them and include them in **vg01** with **vgcreate** as shown in the following steps:

```
ny3c15# mkdir /dev/vg01

ny3c15# mknod /dev/vg01/group c 64 0x010000

ny3c15# ioscan -funC disk
Class     I  H/W Path    Driver      S/W State H/W Type  Description
===================================================================
disk      0  8/0.5.0    sdisk       CLAIMED   DEVICE    SEAGATE ST34572WC
                        /dev/dsk/c0t5d0   /dev/rdsk/c0t5d0
disk      1  8/0.8.0    sdisk       CLAIMED   DEVICE    SEAGATE ST34371W
                        /dev/dsk/c0t8d0   /dev/rdsk/c0t8d0
disk      2  8/4.13.0   sdisk       CLAIMED   DEVICE    SEAGATE ST32171W
                        /dev/dsk/c1t13d0   /dev/rdsk/c1t13d0
disk      3  8/4.14.0   sdisk       CLAIMED   DEVICE    SEAGATE ST32171W
                        /dev/dsk/c1t14d0   /dev/rdsk/c1t14d0
disk      4  8/4.15.0   sdisk       CLAIMED   DEVICE    SEAGATE ST32171W
                        /dev/dsk/c1t15d0   /dev/rdsk/c1t15d0
disk      5  8/8.1.0    sdisk       CLAIMED   DEVICE    SEAGATE ST32171W
                        /dev/dsk/c2t1d0   /dev/rdsk/c2t1d0
disk      6  8/8.10.0   sdisk       CLAIMED   DEVICE    SEAGATE ST32171W
                        /dev/dsk/c2t10d0   /dev/rdsk/c2t10d0
disk      7  8/8.11.0   sdisk       CLAIMED   DEVICE    SEAGATE ST32171W
                        /dev/dsk/c2t11d0   /dev/rdsk/c2t11d0
disk      8  8/16/5.2.0 sdisk       CLAIMED   DEVICE   CD-ROM XM-5701TA
                        /dev/dsk/c3t2d0   /dev/rdsk/c3t2d0

ny3c15# vgdisplay -v /dev/vg00 | tail -9

   --- Physical volumes ---
   PV Name                     /dev/dsk/c0t5d0
```

```
   PV Status                 available
   Total PE                  1023
   Free PE                   38
   Autoswitch                On

ny3c15# pvcreate -f /dev/rdsk/c1t13d0
Physical volume "/dev/rdsk/c1t13d0" has been successfully created.

ny3c15# pvcreate -f /dev/rdsk/c2t1d0
Physical volume "/dev/rdsk/c2t1d0" has been successfully created.

ny3c15# vgcreate /dev/vg01 /dev/dsk/c1t13d0 /dev/dsk/c2t1d0
Volume group "/dev/vg01" has been successfully created.
Volume Group configuration for /dev/vg01 has been saved in /etc/lvmconf/vg01.conf

ny3c15# strings /etc/lvmtab
/dev/vg00
-=8%#
/dev/dsk/c0t5d0
/dev/vg01
-==sW
/dev/dsk/c1t13d0
/dev/dsk/c2t1d0
```

Note that all of the steps performed in this procedure so far were run on system *ny3c15*. Now we make **vg01** inactive with **vgchange**, run **vgexport** to create the file **/tmp/lvmtab**, and copy it to the second system. We create **/dev/vg01** and run **vgimport** on the second system as shown in the following procedure:

```
ny3c15# vgchange -a n vg01
Volume group "vg01" has been successfully changed.

ny3c15# vgexport -p -s -m /tmp/lvm /dev/vg01

ny3c15# rcp /tmp/lvm_map ny3c16:/tmp/lvm_map

--------------------------------


ny3c16# mkdir /dev/vg01

ny3c16# mknod /dev/vg01/group c 64 0x010000

ny3c16# vgimport -s -m /tmp/lvm_map /dev/vg01
```

We now have **vg01** set up on both of the systems we'll use for our ServiceGuard cluster. The next section covers creating the ServiceGuard cluster.

Creating an MC/ServiceGuard Cluster

Now that we have our shared volume group **vg01** set up we can use it as a shared volume group in our ServiceGuard cluster. The following procedure includes several ServiceGuard commands. I'll include comments for some of the cluster-related commands in the upcoming procedure since they have not yet been covered. Assuming that the ServiceGuard application is loaded we can proceed to create the file **/etc/cmcluster/cmclnodelist** to include the two nodes in our cluster and then query these two nodes with **cmquerycl** to create a template cluster configuration file:

```
ny3c15# cat /etc/cmcluster/cmnodelist  ;also copy (rcp) this to ny3c16

ny3c15            root
ny3c16            root


ny3c15# cd /etc/cmcluster

ny3c15# cmquerycl -n ny3c15 -n ny3c16 -C conf.ascii


Warning: The disk at /dev/dsk/c3t2d0 on node ny3c15 does not have an ID.
Warning: The disk at /dev/dsk/c3t2d0 on node ny3c16 does not have an ID.
Warning: Disks which do not have IDs cannot be included in the topology description.
```

```
Use pvcreate(1m) to give a disk an ID.

Node Names:    ny3c15
               ny3c16

Bridged networks:

1       lan0           (ny3c15)
        lan1           (ny3c15)

2       lan0           (ny3c16)
        lan1           (ny3c16)

IP subnets:

156.153.202.0          lan0  (ny3c15)
                       lan0  (ny3c16)

Possible Heartbeat IPs:

156.153.202.0                     156.153.202.105     (ny3c15)
                                  156.153.202.106     (ny3c16)
Possible Cluster Lock Devices:

/dev/dsk/c1t13d0   /dev/vg01            30 seconds

/dev/dsk/c2t1d0    /dev/vg01            30 seconds

LVM volume groups:

/dev/vg00               ny3c15

/dev/vg01               ny3c15
                        ny3c16

/dev/vg00               ny3c16
```

You can see that **cmquerycl** provides a lot of information about the initial cluster components. The file **/etc/cmcluster/conf.ascii** is a template on which the cluster is based. You can edit this file to make any changes you like to the template, such as to change the name of the cluster, which we'll call *web_server* in our example. The maximum number of application packages that can be configured, and many other changes. The following listing shows the **conf.ascii** file for our example:

```
# cat conf.ascii
# **********************************************************************
# ********* HIGH AVAILABILITY CLUSTER CONFIGURATION FILE ***************
# ***** For complete details about cluster parameters and how to    ****
# ***** set them, consult the cmquerycl(1m) manpage or your manual. ****
# **********************************************************************

# Enter a name for this cluster.  This name will be used to identify the
# cluster when viewing or manipulating it.

CLUSTER_NAME       web_server


# Cluster Lock Device Parameters.  This is the volume group that
# holds the cluster lock which is used to break a cluster formation
# tie.  This volume group should not be used by any other cluster
# as cluster lock device.

FIRST_CLUSTER_LOCK_VG/dev/vg01


# Definition of nodes in the cluster.
# Repeat node definitions as necessary for additional nodes.

NODE_NAME          ny3c15
  NETWORK_INTERFACElan0
    HEARTBEAT_IP   156.153.202.105
  NETWORK_INTERFACElan1
  FIRST_CLUSTER_LOCK_PV/dev/dsk/c1t13d0
# List of serial device file names
# For example:
```

```
# SERIAL_DEVICE_FILE/dev/tty0p0

# Possible standby Network Interfaces for lan0: lan1.

NODE_NAME          ny3c16
  NETWORK_INTERFACElan0
    HEARTBEAT_IP   156.153.202.106
  NETWORK_INTERFACElan1
  FIRST_CLUSTER_LOCK_PV/dev/dsk/c1t13d0
# List of serial device file names
# For example:
# SERIAL_DEVICE_FILE/dev/tty0p0

# Possible standby Network Interfaces for lan0: lan1.


# Cluster Timing Parmeters (microseconds).

HEARTBEAT_INTERVAL 1000000
NODE_TIMEOUT       2000000


# Configuration/Reconfiguration Timing Parameters (microseconds).

AUTO_START_TIMEOUT 600000000
NETWORK_POLLING_INTERVAL2000000

# Package Configuration Parameters.
# Enter the maximum number of packages which will be configured in the cluster.
# You can not add packages beyond this limit.
# This parameter is required.
MAX_CONFIGURED_PACKAGES7



# List of cluster aware Volume Groups. These volume groups will
# be used by package applications via the vgchange -a e command.
# For example:
# VOLUME_GROUP     /dev/vgdatabase.
# VOLUME_GROUP     /dev/vg02.

VOLUME_GROUP       /dev/vg01

root@ny3c15   [/etc/cmcluster]
#
```

Now that we've made the changes to **conf.ascii**, we'll run **cmcheckconf**, which will perform a verification of our changes. With no errors found in our verification, we can apply our configuration with **cmapplyconf** after we enable **vg01**. We can then bring up the cluster with **cmruncl** as shown in the following listing:

```
ny3c15# cmcheckconf -C /etc/cmcluster/conf.ascii


Begin cluster verification...
Adding node ny3c15 to cluster web_server.
Adding node ny3c16 to cluster web_server.

Verification completed with no errors found.
Use the cmapplyconf command to apply the configuration.
ny3c15# vgchange -a y vg01
Activated volume group
Volume group "vg01" has been successfully changed.


ny3c15# cmapplyconf -C /etc/cmcluster/conf.ascii
#

Begin cluster verification...
Adding configuration to node ny3c15
Adding configuration to node ny3c16
Adding node ny3c15 to cluster web_server.
Adding node ny3c16 to cluster web_server.
Completed the cluster creation.

root@ny3c15   [/etc/cmcluster]
```

```
root@ny3c15   [/etc/cmcluster]

ny3c15# cmruncl
```

We received no errors and everything ran smoothly, so we can run **cmviewcl** to view our cluster and see if indeed it has formed:

```
ny3c15# cmviewcl -v

CLUSTER      STATUS
web_server   up

  NODE         STATUS       STATE
  ny3c15       up           running

    Network_Parameters:
    INTERFACE    STATUS       PATH         NAME
    PRIMARY      up           8/16/6       lan0
    STANDBY      up           8/20/5/1     lan1

  NODE         STATUS       STATE
  ny3c16       up           running

    Network_Parameters:
    INTERFACE    STATUS       PATH         NAME
    STANDBY      up           8/20/5/1     lan1
    PRIMARY      up           8/16/6       lan0
```

The **cmviewcl** output indicates that our two nodes are up and running smoothly. At this point we have a fully operational ServiceGuard cluster; however, there are no applications that are protected by the cluster. Now let's proceed to the third step, where we'll cover creating an application package.

Create a Highly Available Package

The procedure to set up a package is lengthy, so I'll provide some of the information for the Web server used in this example.

The first step for our particular example is to add an IP address for the Web server to the **/etc/hosts** file. After this is complete we need to perform some LVM-related work to create **vg02**, which will be used as a shared volume for our highly available package. I'll show only commands issued to create **vg02** since it is similar to the procedure used for **vg01** earlier. The main difference is that after **vg02** is created, a mirrored logical volume is put on it that will be used for our highly available application.

```
ny3c15# pvcreate -f /dev/rdsk/c1t14d0

Physical volume "/dev/rdsk/c1t14d0" has been successfully created.

root@ny3c15   [/root]

ny3c15# pvcreate -f /dev/rdsk/c2t10d0
Physical volume "/dev/rdsk/c2t10d0" has been successfully created.

ny3c15# mkdir /dev/vg02

ny3c15# mknod /dev/vg02/group c 64 0x020000

ny3c15# vgcreate /dev/vg02 /dev/dsk/c1t14d0 /dev/dsk/c2t10d0

Volume group "/dev/vg02" has been successfully created.
Volume Group configuration for /dev/vg02 has been saved in /etc/lvmconf/vg02.conf

ny3c15# lvcreate -L 200 -n stk /dev/vg02
Logical volume "/dev/vg02/stk" has been successfully created with
character device "/dev/vg02/rstk".
Logical volume "/dev/vg02/stk" has been successfully extended.
Volume Group configuration for /dev/vg02 has been saved in /etc/lvmconf/vg02.conf

ny3c15# lvextend -m 1 /dev/vg02/stk
The newly allocated mirrors are now being synchronized.This operation will
take some time. Please wait ....
Logical volume "/dev/vg02/stk" has been successfully extended.
```

```
Volume Group configuration for /dev/vg02 has been saved in /etc/lvmconf/vg02.conf

ny3c15# newfs -F vxfs /dev/vg02/rstk
    version 3 layout
    204800 sectors, 204800 blocks of size 1024, log size 1024 blocks
    unlimited inodes, 204800 data blocks, 203656 free data blocks
    7 allocation units of 32768 blocks, 32768 data blocks
    last allocation unit has 8192 data blocks
    first allocation unit starts at block 0
    overhead per allocation unit is 0 blocks

ny3c15# mkdir /opt/STK

ny3c15# mount /dev/vg02/stk /opt/STK

ny3c15# bdf
Filesystem          kbytes    used   avail %used Mounted on
/dev/vg00/lvol3      86016   42282   41006   51% /
/dev/vg00/lvol1      67733   17057   43902   28% /stand
/dev/vg00/lvol10    200704  189924   10780   95% /var
/dev/vg00/lvol9     471040  434239   34544   93% /usr
/dev/vg00/lvol8    1134592  924276  197172   82% /u01
/dev/vg00/lvol7      49152    1522   44657    3% /tmp
/dev/vg00/lvol6     577536  449719  119887   79% /opt
/dev/vg00/lvol5     172032  139466   30538   82% /home
/dev/vg00/lvol4     659456  595106   60350   91% /CD_OPS816
/dev/vg02/stk       204800    1157  190923    1% /opt/STK
```

The **/opt/STK** will be used for our highly available application. Both systems in the cluster will have access to **vg02**, since it is part of our high availability application.

Next, we install our application in **/opt/STK** and customize it. The Web server configuration file is customized to include some information peculiar to the application as it will run in our example. Then the **httpd.conf** file is updated to include the name of the package, which we'll call *web_stk*.

After our application is installed and customized on **/dev/vg02**, we can proceed with some additional LVM work to make **vg02** accessible on the second system in our cluster:

```
ny3c16# mkdir /dev/vg02

ny3c16# mknod /dev/vg02/group c 64 0x020000

ny3c15# umount /opt/STK

ny3c15# vgchange -a n vg02

ny3c15# vgexport -p -s -m /tmp/lvm /dev/vg02
#
Beginning the export process on Volume Group "/dev/vg02".
/dev/dsk/c1t14d0
/dev/dsk/c2t10d0

ny3c15# rcp /tmp/lvm_map ny3c16:/tmp/lvm_map
```

On the second system, we'll import the volume group and access it to ensure that it works:

[View full width]

```
ny3c16# vgimport -s -m /tmp/lvm /dev/vg02
vgimport: Warning:  Volume Group belongs to different CPU ID.
Can not determine if Volume Group is in use on another system. Continuing.
Warning: A backup of this volume group may not exist on this machine.
Please remember to take a backup using the vgcfgbackup command after activating the
➥ volume group.

root@ny3c16   [/root]

ny3c16# vgchange -a y vg02
Activated volume group
Volume group "vg02" has been successfully changed.

root@ny3c16   [/root]
ny3c16# mkdir /opt/STK
```

```
ny3c16# mount -o ro /dev/vg02/stk /opt/STK

ny3c16# bdf
Filesystem          kbytes    used   avail %used Mounted on
/dev/vg00/lvol3      86016   41978   41293   50% /
/dev/vg00/lvol1      67733   17057   43902   28% /stand
/dev/vg00/lvol10    200704  189896   10808   95% /var
/dev/vg00/lvol9     471040  434286   34500   93% /usr
/dev/vg00/lvol8    1134592  924276  197172   82% /u01
/dev/vg00/lvol7      49152    1522   44657    3% /tmp
/dev/vg00/lvol6     577536  449719  119887   79% /opt
/dev/vg00/lvol5     172032  139466   30538   82% /home
/dev/vg00/lvol4     659456  595106   60350   91% /CD_OPS816
/dev/vg02/stk       204800   93528  104318   47% /opt/STK

ny3c16# umount /opt/STK

root@ny3c16   [/root]
# vgchange -a n vg02
Volume group "vg02" has been successfully changed.
```

Note that some of the previous commands were performed on *ny3c15* and others on *ny3c16*. The **mount** at the end is a test to ensure that **vg03** is accessible on system *ny3c16*.

Now we're ready to perform some package-related work. First we'll create and edit the Web server package, which we'll call STK:

```
ny3c15# mkdir /etc/cmcluster/web_stk

ny3c15# cd /etc/cmcluster/web_stk

ny3c15# cmmakepkg -p web_stk.conf

Package control script is created.
This file must be edited before it can be used.
```

The package we've created is a template that we can edit to include all of the characteristics of our package. The *-p* specifies that a *package* template is to be created. Shorty, we'll use the **cmmakepkg** command with the *-s* option to create a *script*.

The **web_stk.conf** file needs to include all of the information related to our package. We provide a lot of information related to the package, such as the names of the two nodes, the run and halt scripts for the package, the service name, and so on. The following is the full listing of the file, with arrows appearing next to the lines that were modified to include information specific to our cluster:

```
# **********************************************************************
# ****** HIGH AVAILABILITY PACKAGE CONFIGURATION FILE (template) *******
# **********************************************************************
# ******* Note: This file MUST be edited before it can be used. ********
# * For complete details about package parameters and how to set them, *
# * consult the MC/ServiceGuard or MC/LockManager manpages or manuals. *
# **********************************************************************

# Enter a name for this package.  This name will be used to identify the
# package when viewing or manipulating it.  It must be different from
# the other configured package names.

PACKAGE_NAME       web_stk                                          <--


# Enter the failover policy for this package. This policy will be used
# to select an adoptive node whenever the package needs to be started.
# The default policy unless otherwise specified is CONFIGURED_NODE.
# This policy will select nodes in priority order from the list of
# NODE_NAME entries specified below.
#
# The alternative policy is MIN_PACKAGE_NODE. This policy will select
# the node, from the list of NODE_NAME entries below, which is
# running the least number of packages at the time this package needs
# to start.

FAILOVER_POLICY    CONFIGURED_NODE
```

```
# Enter the failback policy for this package. This policy will be used
# to determine what action to take when a package is not running on
# its primary node and its primary node is capable of running the
# package. The default policy unless otherwise specified is MANUAL.
# The MANUAL policy means no attempt will be made to move the package
# back to its primary node when it is running on an adoptive node.
#
# The alternative policy is AUTOMATIC. This policy will attempt to
# move the package back to its primary node whenever the primary node
# is capable of running the package.

FAILBACK_POLICY    MANUAL


# Enter the names of the nodes configured for this package.  Repeat
# this line as necessary for additional adoptive nodes.
# Order IS relevant.  Put the second Adoptive Node AFTER the first
# one.
# Example : NODE_NAME  original_node
#           NODE_NAME  adoptive_node

NODE_NAME          ny3c15                                          <--
NODE_NAME          ny3c16                                          <--


# Enter the complete path for the run and halt scripts.  In most cases
# the run script and halt script specified here will be the same script,
# the package control script generated by the cmmakepkg command.  This
# control script handles the run(ning) and halt(ing) of the package.
# If the script has not completed by the specified timeout value,
# it will be terminated.  The default for each script timeout is
# NO_TIMEOUT.  Adjust the timeouts as necessary to permit full
# execution of each script.
# Note: The HALT_SCRIPT_TIMEOUT should be greater than the sum of
# all SERVICE_HALT_TIMEOUT specified for all services.

RUN_SCRIPT         /etc/cmcluster/web_stk/web_stk.cntl                        <--
RUN_SCRIPT_TIMEOUT NO_TIMEOUT
HALT_SCRIPT        /etc/cmcluster/web_stk/web_stk.cntl                        <--
HALT_SCRIPT_TIMEOUTNO_TIMEOUT


# Enter the SERVICE_NAME, the SERVICE_FAIL_FAST_ENABLED and the
# SERVICE_HALT_TIMEOUT values for this package.  Repeat these
# three lines as necessary for additional service names.  All
# service names MUST correspond to the service names used by
# cmrunserv and cmhaltserv commands in the run and halt scripts.
#
# The value for SERVICE_FAIL_FAST_ENABLED can be either YES or
# NO.  If set to YES, in the event of a service failure, the
# cluster software will halt the node on which the service is
# running.  If SERVICE_FAIL_FAST_ENABLED is not specified, the
# default will be NO.
#
# SERVICE_HALT_TIMEOUT is represented in the number of seconds.
# This timeout is used to determine the length of time (in
# seconds) the cluster software will wait for the service to
# halt before a SIGKILL signal is sent to force the termination
# of the service.  In the event of a service halt, the cluster
# software will first send a SIGTERM signal to terminate the
# service.  If the service does not halt, after waiting for the
# specified SERVICE_HALT_TIMEOUT, the cluster software will send
# out the SIGKILL signal to the service to force its termination.
# This timeout value should be large enough to allow all cleanup
# processes associated with the service to complete.  If the
# SERVICE_HALT_TIMEOUT is not specified, a zero timeout will be
# assumed, meaning the cluster software will not wait at all
# before sending the SIGKILL signal to halt the service.
#
# Example: SERVICE_NAME                  DB_SERVICE
#          SERVICE_FAIL_FAST_ENABLED     NO
#          SERVICE_HALT_TIMEOUT          300
#
# To configure a service, uncomment the following lines and
# fill in the values for all of the keywords.
```

```
#
SERVICE_NAME                    web_stk.mon                                          <--
SERVICE_FAIL_FAST_ENABLED        NO
SERVICE_HALT_TIMEOUT             60


# Enter the network subnet name that is to be monitored for this package.
# Repeat this line as necessary for additional subnet names.  If any of
# the subnets defined goes down, the package will be switched to another
# node that is configured for this package and has all the defined subnets
# available.

SUBNET             156.153.202.0                                                  <--


# The following keywords (RESOURCE_NAME, RESOURCE_POLLING_INTERVAL, and
# RESOURCE_UP_VALUE) are used to specify Package Resource Dependencies.  To
# define a Package Resource Dependency, a RESOURCE_NAME line with a fully
# qualified resource path name, and one or more RESOURCE_UP_VALUE lines are
# required.  A RESOURCE_POLLING_INTERVAL line (how often in seconds the resource
# is to be monitored) is optional and defaults to 60 seconds.  An operator and
# a value are used with RESOURCE_UP_VALUE to define when the resource is to be
# considered up.  The operators are =, !=, >, <, >=, and <=, depending on the
# type of value.  Values can be string or numeric.  If the type is string, then
# only = and != are valid operators.  If the string contains whitespace, it
# must be enclosed in quotes.  String values are case sensitive.  For example,
#
#                  Resource is up when its value is
#                  --------------------------------
#                  RESOURCE_UP_VALUE= UP"UP"
#                  RESOURCE_UP_VALUE!= DOWNAny value except "DOWN"
#                  RESOURCE_UP_VALUE= "On Course""On Course"
#
# If the type is numeric, then it can specify a threshold, or a range to
# define a resource up condition.  If it is a threshold, then any operator
# may be used.  If a range is to be specified, then only > or >= may be used
# for the first operator, and only < or <= may be used for the second operator.
# For example,
#                  Resource is up when its value is
#                  --------------------------------
#                  RESOURCE_UP_VALUE     = 55    (threshold)
#                  RESOURCE_UP_VALUE     > 5.1greater than 5.1    (threshold)
#                  RESOURCE_UP_VALUE     > -5 and < 10between -5 and 10   (range)
#
# Note that "and" is required between the lower limit and upper limit
# when specifying a range.  The upper limit must be greater than the lower
# limit.  If RESOURCE_UP_VALUE is repeated within a RESOURCE_NAME block, then
# they are inclusively OR'd together.  Package Resource Dependencies may be
# defined by repeating the entire RESOURCE_NAME block.
#
# Example : RESOURCE_NAME              /net/interfaces/lan/status/lan0
#                      RESOURCE_POLLING_INTERVAL120
#                      RESOURCE_UP_VALUE= RUNNING
#                      RESOURCE_UP_VALUE= ONLINE
#
#           Means that the value of resource /net/interfaces/lan/status/lan0
#           will be checked every 120 seconds, and is considered to
#           be 'up' when its value is "RUNNING" or "ONLINE".
#
# Uncomment the following lines to specify Package Resource Dependencies.
#
#RESOURCE_NAME         <Full_path_name>
#RESOURCE_POLLING_INTERVAL  <numeric_seconds>
#RESOURCE_UP_VALUE     <op> <string_or_numeric> [and <op> <numeric>]


# The default for PKG_SWITCHING_ENABLED is YES. In the event of a
# failure, this permits the cluster software to transfer the package
# to an adoptive node.  Adjust as necessary.

PKG_SWITCHING_ENABLEDYES


# The default for NET_SWITCHING_ENABLED is YES.  In the event of a
# failure, this permits the cluster software to switch LANs locally
```

```
# (transfer to a standby LAN card).  Adjust as necessary.

NET_SWITCHING_ENABLEDYES

# The default for NODE_FAIL_FAST_ENABLED is NO.  If set to YES,
# in the event of a failure, the cluster software will halt the node
# on which the package is running.  Adjust as necessary.

NODE_FAIL_FAST_ENABLEDNO
```

The **web_stk.conf** file calls a control file, which we'll name **web_stk.cntl**, that contains the functions for the run and halt. This file is produced with the following command:

```
ny3c15# cmmakepkg -s web_stk.cntl
```

The *-s* option creates a template package control script. In this file we'll add all of the information important to the package, such as volume group, service name, and package start and stop routines. Again, since this is only a template, we'll have to include important information in it. Because this is a long file, I've included only the customer-defined portion of the file. This does not imply that the other lines are not important, because the script won't work without them:

```
#"(#) A.11.05    $Revision: 82.3 $ $Date: 98/11/03 10:56:49 $"
# ************************************************************************
# *                                                                      *
# *        HIGH AVAILABILITY PACKAGE CONTROL SCRIPT (template)           *
# *                                                                      *
# *       Note: This file MUST be edited before it can be used.          *
# *                                                                      *
# ************************************************************************

# UNCOMMENT the variables as you set them.

# Set PATH to reference the appropriate directories.
PATH=/sbin:/usr/bin:/usr/sbin:/etc:/bin

# VOLUME GROUP ACTIVATION:
# Specify the method of activation for volume groups.
# Leave the default ("VGCHANGE="vgchange -a e") if you want volume
# groups activated in exclusive mode. This assumes the volume groups have
# been initialized with 'vgchange -c y' at the time of creation.
#
# Uncomment the first line (VGCHANGE="vgchange -a e -q n"), and comment
# out the default, if your disks are mirrored on separate physical paths,
#
# Uncomment the second line (VGCHANGE="vgchange -a e -q n -s"), and comment
# out the default, if your disks are mirrored on separate physical paths,
# and you want the mirror resynchronization to ocurr in parallel with
# the package startup.
#
# Uncomment the third line (VGCHANGE="vgchange -a y") if you wish to
# use non-exclusive activation mode. Single node cluster configurations
# must use non-exclusive activation.
#
# VGCHANGE="vgchange -a e -q n"
# VGCHANGE="vgchange -a e -q n -s"
# VGCHANGE="vgchange -a y"
VGCHANGE="vgchange -a e"# Default

# VOLUME GROUPS
# Specify which volume groups are used by this package. Uncomment VG[0]=""
# and fill in the name of your first volume group. You must begin with
# VG[0], and increment the list in sequence.
#
# For example, if this package uses your volume groups vg01 and vg02, enter:
#         VG[0]=vg01
#         VG[1]=vg02
#
# The volume group activation method is defined above. The filesystems
# associated with these volume groups are specified below.
#
VG[0]=vg02

# FILESYSTEMS
# Specify the filesystems which are used by this package. Uncomment
```

```
# LV[0]=""; FS[0]=""; FS_MOUNT_OPT[0]="" and fill in the name of your first
# logical volume, filesystem and mount option for the file system. You must
# begin with LV[0], FS[0] and FS_MOUNT_OPT[0] and increment the list in
# sequence.
#
# For example, if this package uses the file systems pkg1a and pkg1b,
# which are mounted on the logical volumes lvol1 and lvol2 with read and
# write options enter:
#         LV[0]=/dev/vg01/lvol1; FS[0]=/pkg1a; FS_MOUNT_OPT[0]="-o rw"
#         LV[1]=/dev/vg01/lvol2; FS[1]=/pkg1b; FS_MOUNT_OPT[1]="-o rw"
#
# The filesystems are defined as triplets of entries specifying the logical
# volume, the mount point and the mount options for the file system. Each
# filesystem will be fsck'd prior to being mounted. The filesystems will be
# mounted in the order specified during package startup and will be unmounted
# in reverse order during package shutdown. Ensure that volume groups
# referenced by the logical volume definitions below are included in
# volume group definitions above.
#
LV[0]=/dev/vg02/stk; FS[0]=/opt/STK; FS_MOUNT_OPT[0]=""

# FILESYSTEM UNMOUNT COUNT
# Specify the number of unmount attempts for each filesystem during package
# shutdown.  The default is set to 1.
LV_UMOUNT_COUNT=1

# IP ADDRESSES
# Specify the IP and Subnet address pairs which are used by this package.
# Uncomment IP[0]="" and SUBNET[0]="" and fill in the name of your first
# IP and subnet address. You must begin with IP[0] and SUBNET[0] and
# increment the list in sequence.
#
# For example, if this package uses an IP of 192.10.25.12 and a subnet of
# 192.10.25.0 enter:
#         IP[0]=192.10.25.12
#         SUBNET[0]=192.10.25.0 # (netmask=255.255.255.0)
#
# Hint: Run "netstat -i" to see the available subnets in the Network field.
#
# IP/Subnet address pairs for each IP address you want to add to a subnet
# interface card.  Must be set in pairs, even for IP addresses on the same
# subnet.
#
IP[0]=156.153.202.93                            # package IP address
SUBNET[0]=156.153.202.0

# SERVICE NAMES AND COMMANDS.
# Specify the service name, command, and restart parameters which are
# used by this package. Uncomment SERVICE_NAME[0]="", SERVICE_CMD[0]="",
# SERVICE_RESTART[0]="" and fill in the name of the first service, command,
# and restart parameters. You must begin with SERVICE_NAME[0], SERVICE_CMD[0],
# and SERVICE_RESTART[0] and increment the list in sequence.
#
# For example:
#         SERVICE_NAME[0]=pkg1a
#         SERVICE_CMD[0]="/usr/bin/X11/xclock -display 192.10.25.54:0"
#         SERVICE_RESTART[0]=""  # Will not restart the service.
#
#         SERVICE_NAME[1]=pkg1b
#         SERVICE_CMD[1]="/usr/bin/X11/xload -display 192.10.25.54:0"
#         SERVICE_RESTART[1]="-r 2"   # Will restart the service twice.
#
#         SERVICE_NAME[2]=pkg1c
#         SERVICE_CMD[2]="/usr/sbin/ping"
#         SERVICE_RESTART[2]="-R" # Will restart the service an infinite
#                                 number of times.
#
# Note: No environmental variables will be passed to the command, this
# includes the PATH variable. Absolute path names are required for the
# service command definition.  Default shell is /usr/bin/sh.
#
SERVICE_NAME[0]=web_stk.mon
SERVICE_CMD[0]=/etc/cmcluster/web_stk/web_stk.mon
SERVICE_RESTART[0]=""
```

```
# DTC manager information for each DTC.
# Example: DTC[0]=dtc_20
#DTC_NAME[0]=


# START OF CUSTOMER DEFINED FUNCTIONS

# This function is a place holder for customer define functions.
# You should define all actions you want to happen here, before the service is
# started.  You can create as many functions as you need.

function customer_defined_run_cmds
{
# ADD customer defined run commands.
#: # do nothing instruction, because a function must contain some command.
                  /usr/local/bin/httpd
                  test_return 51
}

# This function is a place holder for customer define functions.
# You should define all actions you want to happen here, before the service is
# halted.

function customer_defined_halt_cmds
{
# ADD customer defined halt commands.
#: # do nothing instruction, because a function must contain some command.
                  if
                  [ -f /usr/local/etc/httpd/logs/httpd.pid ]
                  then
                  pid=$(cat /usr/local/etc/httpd/logs/httpd.pid)
                  if ps -p $pid > /dev/null 2>&1
                  then
                  kill $pid
                  fi
                  fi
                  test_return 52
}

# END OF CUSTOMER DEFINED FUNCTIONS


# START OF RUN FUNCTIONS
```

The **web_stk**, **srm.conf**, and **httpd.conf** files all have to be copied to the second system in the cluster and have to be made executable.

There is also a monitoring script that we produced to print messages regarding the system on which the package is running.

After all of these scripts have been produced, they have to be copied to the second system. All of the files have to be executable.

Next we mark the volume group **vg02** as a cluster volume group using the *-c y* option as shown below:

```
ny3c15# vgchange -a n vg02
vgchange: Volume group "vg02" has been successfully changed.

#

ny3c15# vgchange -c y vg02
Performed Configuration change.
Volume group "vg02" has been successfully changed.
```

The *-c* specifies that a cluster-related option will be specified, and the *y* indicates that **vg02** is a member of the cluster.

Now that we've made all of the necessary modifications to files, we can run the package commands shown below:

```
ny3c15# cmcheckconf -P web_stk.conf

Begin package verification...
Verification completed with no errors found.
```

```
Use the cmapplyconf command to apply the configuration.

ny3c15# cmapplyconf -P web_stk.conf

Begin package verification...
Modify the package configuration ([y]/n)? y
Completed the cluster update.

ny3c15# cmmodpkg -e web_stk

root@ny3c15   [/etc/cmcluster/web_stk]
```

In the previous three commands we checked our configuration with **cmcheckconf**, applied our configuration with **cmapplyconf**, and then modified our package to enable it with the *-e* option. You can also specify specific systems on which to enable a package with:

```
cmmodpkg -e -n nodename -n nodename package_name
```

You may have to issue this form of the command after your package has automatically switched to the backup node and then the backup node fails. At this point the package is running on the first system and we can issue **cmviewcl -v** to see the details concerning our cluster and the web server package:

```
ny3c15# cmviewcl -v

CLUSTER      STATUS
web_server   up

  NODE         STATUS       STATE
  ny3c15       up           running

    Network_Parameters:
    INTERFACE    STATUS       PATH         NAME
    PRIMARY      up           8/16/6       lan0
    STANDBY      up           8/20/5/1     lan1

    PACKAGE      STATUS       STATE        PKG_SWITCH   NODE
    web_stk      up           running      enabled      ny3c15

      Policy_Parameters:
      POLICY_NAME     CONFIGURED_VALUE
      Failover        configured_node
      Failback        manual

      Script_Parameters:
      ITEM       STATUS   MAX_RESTARTS  RESTARTS   NAME
      Service    up                 0         0   web_stk.mon
      Subnet     up                               156.153.202.0

      Node_Switching_Parameters:
      NODE_TYPE    STATUS       SWITCHING    NAME
      Primary      up           enabled      ny3c15       (current)
      Alternate    up           enabled      ny3c16

  NODE         STATUS       STATE
  ny3c16       up           running

    Network_Parameters:
    INTERFACE    STATUS       PATH         NAME
    STANDBY      up           8/20/5/1     lan1
    PRIMARY      up           8/16/6       lan0

root@ny3c15   [/etc/cmcluster/web_stk]
```

web_stk is running on *ny3c15* at this time. Next we'll halt the package on *ny3c15* with **cmhaltpkg**, and manually run it on *ny3c16* with **cmrunpkg** and run **cmviewcl** to see the package moved to *ny3c16*:

```
ny3c15# cmhaltpkg web_stk
Halting package web_stk.

cmhaltpkg  : Successfully halted package web_stk.
cmhaltpkg  : Completed successfully on all packages specified.

root@ny3c15   [/etc/cmcluster/web_stk]
ny3c15# cmviewcl -v
```

```
CLUSTER      STATUS
web_server   up

  NODE         STATUS       STATE
  ny3c15       up           running

    Network_Parameters:
    INTERFACE    STATUS       PATH         NAME
    PRIMARY      up           8/16/6       lan0
    STANDBY      up           8/20/5/1     lan1

  NODE         STATUS       STATE
  ny3c16       up           running

    Network_Parameters:
    INTERFACE    STATUS       PATH         NAME
    STANDBY      up           8/20/5/1     lan1
    PRIMARY      up           8/16/6       lan0

UNOWNED_PACKAGES

    PACKAGE      STATUS       STATE        PKG_SWITCH   NODE
    web_stk      down         halted       disabled     unowned

      Policy_Parameters:
      POLICY_NAME     CONFIGURED_VALUE
      Failover        configured_node
      Failback        manual

      Script_Parameters:
      ITEM       STATUS   NODE_NAME    NAME
      Subnet     up       ny3c15       156.153.202.0
      Subnet     up       ny3c16       156.153.202.0

      Node_Switching_Parameters:
      NODE_TYPE    STATUS       SWITCHING    NAME
      Primary      up           enabled      ny3c15
      Alternate    up           enabled      ny3c16

root@ny3c15   [/etc/cmcluster/web_stk]
#
ny3c15# cmrunpkg -n ny3c16 -v web_stk
ny3c15# cmviewcl -v
#

CLUSTER      STATUS
web_server   up

  NODE         STATUS       STATE
  ny3c15       up           running

    Network_Parameters:
    INTERFACE    STATUS       PATH         NAME
    PRIMARY      up           8/16/6       lan0
    STANDBY      up           8/20/5/1     lan1

  NODE         STATUS       STATE
  ny3c16       up           running

    Network_Parameters:
    INTERFACE    STATUS       PATH         NAME
    STANDBY      up           8/20/5/1     lan1
    PRIMARY      up           8/16/6       lan0

    PACKAGE      STATUS       STATE        PKG_SWITCH   NODE
    web_stk      up           running      disabled     ny3c16

      Policy_Parameters:
      POLICY_NAME     CONFIGURED_VALUE
      Failover        configured_node
      Failback        manual

      Script_Parameters:
      ITEM       STATUS   MAX_RESTARTS  RESTARTS   NAME
      Service    up                 0         0   web_stk.mon
```

```
      Subnet     up                                  156.153.202.0

      Node_Switching_Parameters:
      NODE_TYPE    STATUS       SWITCHING    NAME
      Primary      up           enabled      ny3c15
      Alternate    up           enabled      ny3c16       (current)

root@ny3c15   [/etc/cmcluster/web_stk]
```

In this example we were able to manually halt the package on *ny3c15*, run it on *ny3c16*, and view it in the halted state in the first **cmviewcl** output and then view it in the running state with the second **cmviewcl** output.

The automatic failover from *ny3c15* to *ny3c16* also worked with this package. When *ny3c15* failed, the package automatically moved to *ny3c16* as we had designed it.

Should you encounter problems, there are log files in the package directory that provide useful troubleshooting information. One of the log files will have the path **/etc/cmcluster/***appname/appname.log.* Depending on the error encountered, you may get a line number in your script where the error was found. The following messages were in the log file for our specific application and clearly pointed to a problem with our IP address that we were able to fix:

[View full width]

```
########### Node "ny3c15": Starting package at Tue Jul 23 14:18:51 EDT 2002 ###########
Jul 23 14:18:51 - "ny3c15": Activating volume group vg02 with exclusive option.
Activated volume group in Exclusive Mode.
Volume group "vg02" has been successfully changed.
Jul 23 14:18:52 - Node "ny3c15": Checking filesystems:
   /dev/vg02/stk
file system is clean - log replay is not required
Jul 23 14:18:54 - Node "ny3c15": Mounting /dev/vg02/stk at /opt/STK
Jul 23 14:18:54 - Node "ny3c15": Adding IP address 156.153.202.105 to subnet 156.153.202.0
cmmodnet  : Unable to verify IP address 156.153.202.105.
cmmodnet  : 156.153.202.105 might already be configured as a heartbeat IP address.
                  ERROR:  Function add_ip_address
                  ERROR:  Failed to add IP address to subnet
Jul  23  14:18:54 - Node "ny3c15": Remove IP address 156.153.202.105 from subnet 156.153.
➥ 202.0
cmmodnet  : Unable to complete command : Device busy
                 ERROR:  Function remove_ip_address
                 ERROR:  Failed to remove 156.153.202.105
Jul 23 14:18:55 - Node "ny3c15": Unmounting filesystem on /dev/vg02/stk
                  WARNING:   Running fuser to remove anyone using the file system
➥ directly.
/dev/vg02/stk:

Jul 23 14:18:56 - Node "ny3c15": Deactivating volume group vg02
Deactivated volume group in Exclusive Mode.
Volume group "vg02" has been successfully changed.
```

This example covered all of the fundamentals for working with ServiceGuard. To fully understand the intricacies of ServiceGuard, I recommend taking the HP Education Center courses on this and related topics. When working with high availability solutions, you have to be sure of your implementation and not take any chances.

Section 3: UNIX Background

Chapter 20. UNIX File System Introduction - File System Layout, file and ls Commands

Chapter 21. Working with Files and Directories - Permissions, Commands, File Name Expansion, and Wild Cards

Chapter 22. Viewing Files - Redirection, cat, more, pg, head, and tail Commands

Chapter 23. UNIX Tools - split, wc, sort, cmp, diff, comm, dircmp, cut, paste, join, and tr

Chapter 24. Advanced UNIX Tools - Regular Expressions, sed, awk, and grep

Chapter 25. find Command

Chapter 26. The vi Editor

Chapter 27. Introduction to the KornShell, C Shell, and Bash

Chapter 28. Introduction to Shell Programming

Appendix A. Select HP-UX Online Manual Pages

Appendix B. Virtual Partitions Online Manual Pages



Chapter 20. UNIX File System Introduction - File System Layout, file and ls Commands

The Basis of UNIX

File Types

The file Command

The ls Command

File System Layout

Linux File System Layout



The Basis of UNIX

Most everything in UNIX comes down to files and directories. If you understand the hierarchical UNIX file system layout and how to manipulate files and directories, then you're a long way toward UNIX understanding. In this and upcoming chapters you'll see examples from several UNIX variants as well as Linux.

A lot goes into working with UNIX files and directories. This chapter covers the basics and background of files and directories in UNIX, including:

- The **file** command
- The ls command
- The file system layout

There are references to a number of UNIX operating systems in this chapter, including Linux, because the commands covered apply to most all UNIX variants.

The next chapter, Chapter 21, covers commands related to files and directories. The only commands covered in this chapter are the **file** and **ls** commands. **file** is used to determine the type of file and **ls** is used to list files and directories.



File Types

A file is a means by which information is stored on a UNIX system. The commands you issue, the applications you use, the data you store, and the devices you access, such as printers and keyboard, are all contained in files. One aspect of UNIX that makes it both simple and complex: simple because you know that everything out there is a file, complex because the contents of a file could be anything ranging from ASCII text files to executable programs.

Every file on the system has a filename. The operating system takes care of all file-system-related tasks; you just need to know the name of the file and how to use it. Many types of files are on UNIX systems. Some file types are peculiar to the UNIX variant you are using. Device files, for instance, contain information about the specific hardware platform on which you are running your UNIX variant. In general, however, most file types are similar going from system to system. The file types we will look at are:

- Text Files
- Data Files
- Source Code Files
- Executable Files
- Shell Programs
- Links
- Device Files

Many times when dealing with various types of files, there are extensions associated with the files. Many applications, for instance, associate a specific extension with that application. Text files, for instance, sometimes have a **.txt** extension. Most C source code programs have a **.c** extension, and most C++ programs have a **.cc** extension. Although the extension can be useful for determining the type of a file, no written requirements state that a particular type of file must contain an assigned extension or that you must have any extension at all.

The following table shows some commonly used extensions for various types of files:

Extension	File Type
.1 to .*n*	Online manual source.
.a	An archive or library.
.c	C program source.
.cc	C++ program source.
.csh	C shell script.
.f	FORTRAN program source.
.ksh	Korn shell script.
.o	Object file of compiled source.
.ps	Postscript source.
.shar	Shell archive.
.sh	Bourne shell script.
.tar	**tar** archive.

.txt	ASCII text file.
.Z	Compressed file.

A useful technique for determining the file type is to use the **file** command. Most UNIX variants support the **file** command, although the outputs differ somewhat, so I suggest that you use the **file** command to determine file type. The best way to learn about file types and the **file** command is through example, which we'll do later in this chapter after discussing some common file types in more detail.

Text Files

What could be simpler than a file that contains characters, just like the ones you're now reading in this chapter? These ASCII characters are letters and numerals that represent the work you perform. If, for instance, you use a UNIX editor to create an electronic mail message or a letter, you are creating a text file in most cases. Here is an example of part of an ASCII text file from a Linux system:

[View full width]

```
Thu Nov  5 07:29:53 1999 debug: FTS: appending '/dev/hda1'
Thu Nov  5 07:29:53 1999 debug: FTS: appending '/tmp/LST/swap.sel'
Thu Nov  5 07:29:53 1999 debug: Building partition-list with argument <Linux>
Thu Nov  5 07:29:53 1999 debug: Respecting exclude file /tmp/LST/targets.sel:
/dev/hdd
/dev/hda1
/dev/hda2
Thu Nov  5 07:29:53 1999 debug: No ADDITIONAL partitions available
Thu Nov  5 07:29:59 1999 debug: added '8390' to '/root/tmp/modules.handled'
Thu Nov  5 07:29:59 1999 debug: added '8390' to '/root/etc/modules/2.0.29/#1 Tue Feb 11
➥ 20:36:48 MET 1997.default'
Thu Nov  5 07:29:59 1999 debug: added 'scsi_mod' to '/root/tmp/modules.handled'
Thu Nov  5 07:29:59 1999 debug: added 'scsi_mod' to '/root/etc/modules/2.0.29/#1 Tue Feb
➥ 11 20:36:48 MET 1997.default'
Thu Nov  5 07:29:59 1999 debug: added 'sd_mod' to '/root/tmp/modules.handled'
Thu Nov  5 07:29:59 1999 debug: added 'sd_mod' to '/root/etc/modules/2.0.29/#1 Tue Feb 11
➥ 20:36:48 MET 1997.default'
Thu Nov  5 07:29:59 1999 debug: added 'sr_mod' to '/root/tmp/modules.handled'
Thu Nov  5 07:29:59 1999 debug: added 'sr_mod' to '/root/etc/modules/2.0.29/#1 Tue Feb 11
➥ 20:36:48 MET 1997.default'
Thu Nov  5 07:29:59 1999 debug: added 'st' to '/root/tmp/modules.handled'
Thu Nov  5 07:29:59 1999 debug: added 'st' to '/root/etc/modules/2.0.29/#1 Tue Feb 11 20:
➥ 36:48 MET 1997.default'
Thu Nov  5 07:29:59 1999 debug: added 'sg' to '/root/tmp/modules.handled'
Thu Nov  5 07:29:59 1999 debug: added 'sg' to '/root/etc/modules/2.0.29/#1 Tue Feb 11 20:
➥ 36:48 MET 1997.default'
Thu Nov  5 07:29:59 1999 debug: added 'nfs' to '/root/tmp/modules.handled'
Thu Nov  5 07:29:59 1999 debug: added 'nfs' to '/root/etc/modules/2.0.29/#1 Tue Feb 11 20:
➥ 36:48 MET 1997.default'
Thu Nov  5 07:29:59 1999 debug: added 'isofs' to '/root/tmp/modules.handled'
Thu Nov  5 07:29:59 1999 debug: added 'isofs' to '/root/etc/modules/2.0.29/#1 Tue Feb 11
➥ 20:36:48 MET 1997.default'
Thu Nov  5 07:29:59 1999 debug: 'isofs' is already handled
Thu Nov  5 07:29:59 1999 debug: 'nfs' is already handled
Thu Nov  5 07:29:59 1999 debug: 'sg' is already handled
Thu Nov  5 07:29:59 1999 debug: 'st' is already handled
```

Data Files

A file that contains data used by one of your applications is a data file. If you use a sophisticated desktop publishing tool such as FrameMaker® to write a book, you create data files that FrameMaker uses. These data files contain data, which you can usually read, and formatting information, which you can sometimes read but is usually hidden from you. If your UNIX installation uses a database program, then you may have data files that you can partially read.

Source Code Files

A source code file is a text file that contains information related to a programming language such as C, C++, Pascal, FORTRAN, and so on. These files are readable - at least to the programmer who wrote them. When a programmer develops a source code file, they create a file that conforms to the naming convention of the program language being used, such as adding a ".c" to the end of the file if creating a C program.

The following is an example of a C source code file:

```
/* this is K & R sort program */

# include <stdio.h>
# include <stdlib.h>

       int N;
       int v[1000000];        /* v is array to be sorted */
       int left = 0;          /* left pointer */
       int right;
       int swapcount, comparecount = 0;
                              /* count swaps and compares*/
        int i, j, t;
       char print;
       char pr_incr_sorts;
main()


{
   printf("Enter number of numbers to sort : ");
   scanf("%10d", &N);                    /* 10d used for a BIG input */
   printf ("\n");                            /* select type of input to sort */

   printf("Enter rand(1), in-order(2), or reverse order (3)    sort : ");
   scanf("%2d", &type);
   printf ("\n");                        /* select type of input to sort */

   if (type == 3)
                for (i=0; i<N; ++i)      /* random        */
                      v[i] = (N - i);

   else if (type == 2)
                for (i=0; i<N; ++i)
                      v[i]= (i + 1);        /* in order      */

   else if (type == 1)
                for (i=0; i<N; ++i)
                      v[i]=rand();          /* reverse order */
   fflush(stdin);
   printf("Do you want to see the numbers before sorting (y or n)? : ");
   scanf("%c", &print);
   printf ("\n");                        /* View unsorted numbers?  */
   if (print == 'y')
      {
            printf ("\n");
         for (i=0; i<N; ++i)
            printf("a[%2d]= %2d\n", i, v[i]);
            printf ("\n");
      }

      fflush(stdin);
     printf("Do you want to see the array at each step as it sorts? (y or n)? : ");
      scanf("%c", &pr_incr_sorts);
      printf ("\n");                     /* View incremental sorts?   */

         right = N-1;                    /* right pointer            */

                    qsort(v, left, right);



      {

         fflush(stdin);
         printf ("Here is the sorted list of %2d items\n", N);
             printf ("\n");
         for (i=0; i<N; ++i)
```

```
            printf ("%2d\n ", v[i]);
            printf ("\n");
            printf ("\n");                /* print sorted list        */
                  }
                       printf ("number of swaps = %2d\n ", swapcount);
                       printf ("number of compares = %2d\n ", comparecount);
      }

   /* qsort function */

            void qsort( v, left, right)
                         int v[], left, right;
      {
                  int i, last;
                  if (left > right)
                   return;

                  swap(v, left, (left + right)/2);
                  last = left;
                  for (i=left+1; i <= right; i++)
                  {
                     comparecount = ++comparecount;
                       if (v[i] < v[left])
                  swap(v, ++last, i);
                  }
                  swap(v, left, last);
                  qsort(v, left, last-1);
                  qsort(v, last+1, right);
                 }

                 /* swap function  */


                  swap(v, i, j)
                     int v[], i, j;
                     {int temp;
                        swapcount = swapcount++;
                        temp = v[i];
                        v[i] = v[j];
                        v[j] = temp;


  if (pr_incr_sorts == 'y')
     {
                  printf("Incremental sort of array = ");
        printf ("\n");
        for (i=0; i<N; ++i)
           printf("a[%2d]= %2d\n", i, v[i]);
           printf ("\n");
     }
}
```

Executable Files

Executable files are compiled or interpreted programs that can be run. You can't read executable files, and you'll typically get a bunch of errors, unreadable characters, and beeps from your UNIX system when you try to look at one of these. You may also lose your screen settings and cause other problems.

You don't have to go far in UNIX to find executable files, they are everywhere. Many of the UNIX commands you issue are executable files that you can't read. In addition, if you are developing programs on your system, you are creating your own executables.

Here is an example of what you see if you attempt to send an executable to the screen:

```
unknown/etc/ttytyperunknown<@=>|<@=>|:unknown<@=>
callocLINESCOLUMNSunknownPackaged for
argbad aftger%3
parmnumber missing <@=>|<@=>|:
@ @ 3### @@@A:2TTO|>@#<|2X00R
EraseKillOOPS<@=>|<@=>|:
<@=>|<@=>|:
<@=>|<@=>|:<@=>|ATOO<@=>|:<@=>|<@=>|:<@=>|<@=>|:<@=>|<@=>|:
```

Shell Programs

A shell program is both a file that you can run to perform a task and a file that you can read. So yes, even though you can run this file because it is executable, you can also read it. I'm going to describe shell programming in more detail in Chapter 28.

I consider shell programming to be an important skill for every user to have. I'll spend some time going over the basics of shell programming. Some of the background I'm about to cover relating to file types and permissions is important when it comes to shell programming, so this is important information for you to understand.

Here is an example of part of a shell program that performs an audit of a system:

```
#!/bin/sh
{

echo "PROG>>>>> determining if Ignite-UX loaded on system."


igtest='swlist  | grep Ignite'
echo $igtest
if [ -n "$igtest" ]; then
echo "Ignite-UX installed"
else
echo "Ignite-UX is not installed"
fi

echo "PROG>>>>> determining if make_recovery has been run by
             checking for /var/opt/ignite/arch.include."

if [ -f /var/opt/ignite/recovery/arch.include ]; then
echo "make_recovery has been run."
else
echo "make_recovery has not been run."
fi

echo "PROG>>>>> determining if make_recovery was run with the
             -C (for check_recovery) option."

if [ -f /var/opt/ignite/recovery/makrec.last ]; then
echo "make_recovery with -C has been run. We'll now
             run check_recovery."
/opt/ignite/bin/check_recovery 2>&1
else
echo "make_recovery with -C has not been run. We can't
             run check_recovery."
fi

}  | tee -a /tmp/IMPORTANT/igtest.out
```

The shell program is text you can read and modify if indeed you have permission to do so. In addition to programming information, shell programs contain comments indicated by lines beginning with a #.

Links

A link is a pointer to a file stored elsewhere on the system. Instead of having two or more copies of a file on your system, you can link to a file that already exists on your system.

One particularly useful way that links have been used in UNIX is related to new releases of the operating system. The locations of files sometimes change going from one release to another, and rather than learn all the new locations, links are produced from the old location to the new one. When you run a command using the old location, the link points to the new location.

Links are also useful for centralizing files. If a set of identical files has to be updated often, it is easier to link to a central file and update it, rather than having to update several copies of the file in several different locations.

Device Files

Device files, sometimes called *device special files*, contain information about the hardware connected to your system. Because device special files are associated with system administration functions, they are usually not covered much in user, as opposed to system administration, material. I think the situation should be otherwise. It is very frustrating for a user to want to write a file to a floppy disk or a tape and not have any idea how to access a device file. I'll cover some use of device files so you are not completely in the dark in this area.

Devices on your system can often be accessed with different device files. A disk, for instance, can be accessed with either a block device file or a character device file. Most of this access is the responsibility of your system administrator; however, when you attempt to determine the file type, you may encounter special files of different types such as character and block.

Other types of files are on your system as well, but for the purposes of getting started with UNIX, the file types I will describe supply sufficient background to get you started.

[Team LiB]

PREVIOUS NEXT

The file Command

The **file** command is used to determine the file type. This command is useful because the name of a file does not always indicate its file type. The following examples perform a long listing of a file to provide some background information on the file, and then the **file** command is run to show the file type. I don't cover the command used to list files until the next chapter, but I include it in these examples. The **ls** command provides a listing of files. **ls** is covered in detail in the next chapter. We need to use it in this chapter in only its basic form.. Combined with the "-l" option, you can produce a long listing that provides a lot of information about files. Using **ls -l** in the following examples, you will see the name of each file, file type, permissions, number of hard links, owner name, group name, size in bytes, and time stamp. You may not know what much of this information is for now; however, some of this information may be useful when viewing the output of the **file** command. Examples in this chapter for the **file** command show that different UNIX variants may produce somewhat different outputs of the **file** command. The following examples show an HP-UX output for the "UNIX example" and then a Linux output where available.

Text File (UNIX example)

(Described by the **file** command as *ascii text*.)

```
# ls -l .mosaic-global-history
-rw-r--r--   1 201      users         587 Dec 22  1999 .mosaic-global-history
# file  .mosaic-global-history
.mosaic-global-history: ascii text
#
```

Text File (Linux example)

(Described by the **file** command as *ASCII text*.)

```
# ls -l *
-rw-r--r--   1 root     root       251367 Nov  5 07:11 debug
-rw-r--r--   1 root     root         2020 Nov  5 07:11 history
# file *
debug:      ASCII text
history:    ASCII text
```

Data File (UNIX example)

(Described by the **file** command as *data*.)

```
# ls -l Static.dat
-rw-r--r--   1 201      users      235874 Aug 26  1999 Static.dat
# file Static.dat
Static.dat:     data
#
```

Source Code File (UNIX example)

(Described by the **file** command as *c program text*.)

```
# ls -l krsort.c
-rwxrwxrwx   1 201      users        3234 Nov 16  1999 krsort.c
# file krsort.c
krsort.c:       c program text
#
```

Source Code File (Linux example)

(Described by the **file** command as *C program text*.)

```
# ls -l *.c
-rw-r--r--   1 root     root         4521 Jul 12  1999 intl-bindtextdom.c
-rw-r--r--   1 root     root         6234 Jul 12  1999 intl-cat-compat.c
-rw-r--r--   1 root     root        14128 Jul 12  1999 intl-dcgettext.c
-rw-r--r--   1 root     root         1750 Jul 12  1999 intl-dgettext.c
-rw-r--r--   1 root     root        12759 Jul 12  1999 intl-finddomain.c
-rw-r--r--   1 root     root         1907 Jul 12  1999 intl-gettext.c
-rw-r--r--   1 root     root         1646 Jul 12  1999 intl-intl-compat.c
-rw-r--r--   1 root     root         5361 Jul 12  1999 intl-loadmsgcat.c
-rw-r--r--   1 root     root         7271 Jul 12  1999 intl-localealias.c
-rw-r--r--   1 root     root         2914 Jul 12  1999 intl-textdomain.c
# file *.c
intl-bindtextdom.c: C program text
intl-cat-compat.c:  C program text
intl-dcgettext.c:   C program text
intl-dgettext.c:    C program text
intl-finddomain.c:  C program text
intl-gettext.c:     C program text
intl-intl-compat.c: C program text
intl-loadmsgcat.c:  C program text
intl-localealias.c: C program text
intl-textdomain.c:  C program text
#
```

Executable File (UNIX example)

(Described by the **file** command as *shared executable.*)

```
# ls -l krsort
-rwxr-xr-x   1 201      users      34592 Nov 16  1999 krsort
# file krsort
krsort:         PA-RISC1.1 shared executable dynamically linked -not stripped
#
```

Executable File (LInux example)

(Described by the **file** command as *executable.*)

```
# ls -l a*
-rwxr-xr-x   1 root     root         3888 Jul 24  1999 activate
-rwxr-xr-x   1 root     root         4452 Feb 25  1999 adjtimex
# file a*
activate: ELF 32-bit LSB executable,   Intel 80386, version 1, stripped
adjtimex: ELF 32-bit LSB executable,   Intel 80386, version 1, stripped
#
```

Shell Program (UNIX example)

(Described by the **file** command as *commands text*.)

```
# ls -l llsum
-rwxrwxrwx   1 root     sys         1267 Feb 23  1999 llsum
# file llsum
llsum:          commands text
#
```

Shell Program (Linux example)

(Described by the **file** command as *Bourne shell script text*.)

```
# ls -l request-route
-rwx------  1 root     root      1046 Sep 19  1999 request-route
# file request-route
request-route: Bourne shell script text
#
```

Link (UNIX example)

(The link is not referenced by the **file** command; this is shown as a *shared executable dynamically linked*. The reference to *dynamically linked* does not mean that this is a link.)

```
# ls -l /usr/bin/ar
lr-xr-xr-t   1 root     sys           15 Mar 23  1999 ar -> /usr/ccs/bin/ar
# file /usr/bin/ar
/usr/bin/ar:          s800 shared executable dynamically linked
#
```

Link (Linux example)

(The link shown is a *symbolic link*.)

```
# ls -l reboot
lrwxrwxrwx   1 root   root         4 Nov  5 01:31 reboot -> halt
# file * | grep link
depmod:            symbolic link to modprobe
ksyms:             symbolic link to insmod
pidof:             symbolic link to killall5
reboot:            symbolic link to halt
rmmod:             symbolic link to insmod
swapoff:           symbolic link to swapon
telinit:           symbolic link to init
udosctl:           symbolic link to /sbin/umssync
umssetup:          symbolic link to /sbin/umssync
#
```

Block Device File (UNIX example)

(Described by the **file** command as *block special*.)

```
# ls -l /dev/dsk/c0t1d0
brw-r--r--   1 bin      sys       31 0x001000 Apr 17  1999 /dev/dsk/c0t1d0
# file /dev/dsk/c0t1d0
/dev/dsk/c0t1d0:        block special (31/4096)
#
```

Block Device File (Linux example)

(Described by the **file** command as *block special*.)

```
# ls -l loop*
brw-rw----   1 root     disk       7,   0 Sep 23  1999 loop0
brw-rw----   1 root     disk       7,   1 Sep 23  1999 loop1
brw-rw----   1 root     disk       7,   2 Sep 23  1999 loop2
brw-rw----   1 root     disk       7,   3 Sep 23  1999 loop3
brw-rw----   1 root     disk       7,   4 Sep 23  1999 loop4
brw-rw----   1 root     disk       7,   5 Sep 23  1999 loop5
brw-rw----   1 root     disk       7,   6 Sep 23  1999 loop6
brw-rw----   1 root     disk       7,   7 Sep 23  1999 loop7
# file loop*
loop0: block special (7/0)
loop1: block special (7/1)
loop2: block special (7/2)
loop3: block special (7/3)
loop4: block special (7/4)
loop5: block special (7/5)
```

```
loop6: block special (7/6)
loop7: block special (7/7)
#
```

Character Device File (UNIX example)

(Described by the **file** command as *character special*.)

```
# ls -l /dev/rdsk/c0t1d0
crw-r-----   1 root     sys     188 0x001000 Mar 23  1999 /dev/rdsk/c0t1d0
# file /dev/rdsk/c0t1d0
/dev/rdsk/c0t1d0:       character special (188/4096)
#
```

Character Device File (Linux example)

(Described by the **file** command as *character special*.)

```
# ls -l mi*
crw-rw-rw-   1 root     sys       14,   2 Sep 23  1999 midi00
crw-rw-rw-   1 root     sys       14,  18 Sep 23  1999 midi01
crw-rw-rw-   1 root     sys       14,  34 Sep 23  1999 midi02
crw-rw-rw-   1 root     sys       14,  50 Sep 23  1999 midi03
crw-rw-rw-   1 root     sys       14,   0 Sep 23  1999 mixer
crw-rw-rw-   1 root     sys       14,  16 Sep 23  1999 mixer1
# file mi*
midi00: character special (14/2)
midi01: character special (14/18)
midi02: character special (14/34)
midi03: character special (14/50)
mixer:  character special (14/0)
mixer1: character special (14/16)
#
```

The ls Command

The **ls** command brings with it a lot to discuss. I haven't yet described the options to **ls,** yet we have already used this command and the -/ option as part of the **file** command discussion. You can't do much on a UNIX system without **ls**, so I'll cover it now. The best way to cover the most important options to **ls** is to show examples. I do just that in the upcoming description of the **ls** command.

ls

The following is an example of **ls** without any options other than the directory to list:

```
# ls /home/denise

27247b.exe
410pt1.exe
410pt2.exe
41ndir.exe
41nds1.exe
41nds4.exe
41nwad.exe
41rtr2.exe
HPDA1.EXE
Mail
N3212B6.EXE
SCSI4S.EXE
clean
clean2
clean3
content.exe
dsenh.exe
eg1
eg2
en0316bz.exe
en0316tb.exe
explore.exe
flexi_cd.exe
fred.h
hal.c
hpdl0117.exe
hpdlinst.txt
hpux.patches
j2577a.exe
ja95up.exe
msie10.exe
n32e12n.exe
nfs197.exe
pass.sb
plusdemo.exe
ps4x03.exe
psg
quik_res.exe
rclock.exe
rkhelp.exe
roni.mak
sb.txt
smsup2.exe
softinit.remotesoftcm
srvpr.exe
steve.h
target.exe
tcp41a.exe
tnds2.exe
upgrade.exe
whoon
win95app.exe
total 46718
```

Which of these are files? Which are directories? Have all of the entries been listed? There are many options to **ls** that will answer these questions.

There is not a lot of information reported as a result of having issued this command. **ls** lists the contents of the directory specified, or the current working directory if no directory is specified.

ls -a

To list all the entries of a directory, you would use the *-a* option. Files that begin with a "." are called *hidden files* and are not usually listed with **ls**. The following example shows the output of **ls -a**:

```
$ ls -a /home/denise

.Xauthority
.cshrc
.dt
.dtprofile
.elm
.exrc
.fmrc.orig
.glancerc
.gpmhp
.history
.login
.lrom
.mailrc
.netscape-bookmarks.html
.netscape-cache
.netscape-cookies
.netscape-history
.netscape-newsgroups-news.spry.com
.netscape-newsgroups-newsserv.hp.com
.netscape-preferences
.newsrc-news.spry.com
.newsrc-newsserv.hp.com
.profile
.rhosts
.sh_history
.softbuildrc
.softinit.orig
.sw
.xinitrc
.xsession
27247b.exe
410pt1.exe
410pt2.exe
41ndir.exe
41nds1.exe
41nds4.exe
41nwad.exe
41rtr2.exe
HPDA1.EXE
Mail
N3212B6.EXE
SCSI4S.EXE
clean
clean2
clean3
content.exe
dsenh.exe
eg1
eg2
en0316bz.exe
en0316tb.exe
explore.exe
flexi_cd.exe
fred.h
hal.c
hpdl0117.exe
hpdlinst.txt
hpux.patches
j2577a.exe
```

```
ja95up.exe
msie10.exe
n32e12n.exe
nfs197.exe
pass.sb
plusdemo.exe
ps4x03.exe
psg
quik_res.exe
rclock.exe
rkhelp.exe
roni.mak
sb.txt
smsup2.exe
softinit.remotesoftcm
srvpr.exe
steve.h
target.exe
tcp41a.exe
tnds2.exe
upgrade.exe
whoon
win95app.exe
total 46718
```

Notice that this output includes hidden files, those that begin with a ".", as well as all other files listed with just **ls**. These did not appear when **ls** was issued without the *-a* option. All subsequent examples include the *-a* option.

ls -l

To list all information about the contents of directory, you use the *-l* option to **ls**, as shown in the following example (some of these file names were shortened to fit on the page):

```
$ ls -al /home/denise

-rw-------   1 denise   users         98 Oct  6 09:19 .Xauthority
-r--r--r--   1 denise   users        814 May 19 10:10 .cshrc
drwxr-xr-x   7 denise   users       1024 Sep 26 11:14 .dt
-rwxr-xr-x   1 denise   users       8705 Jul  7 12:04 .dtprofile
drwx------   2 denise   users       1024 Jul 31 18:48 .elm
-r--r--r--   1 denise   users        347 May 19 10:10 .exrc
-rwxrwxrwx   1 denise   users        170 Jun  6 14:20 .fmrc.orig
-rw-------   1 denise   users         97 Jun 12 18:59 .glancerc
-rw-------   1 denise   users      17620 Sep 21 16:11 .gpmhp
-rwxr-xr-x   1 denise   users        391 Sep 19 09:55 .history
-r--r--r--   1 denise   users        341 May 19 10:10 .login
drwx--x--x   2 denise   users       1024 Jul 31 18:48 .lrom
-rw-r--r--   1 denise   users        768 Jul 28 12:54 .mailrc
-rw-------   1 denise   users       1450 Oct  6 13:58 .netscape-bookmarks.html
drwx------   2 denise   users      10240 Oct 10 15:24 .netscape-cache
-rw-------   1 denise   users         91 Sep 18 14:16 .netscape-cookies
-rw-------   1 denise   users      43906 Oct 10 15:32 .netscape-history
-rw-r--r--   1 denise   users        566 Aug 25 14:36 .netscape--news.spry.com
-rw-------   1 denise   users      46514 Jun 28 12:35 .netscape-.hp.com
-rw-------   1 denise   users       1556 Sep 28 15:02 .netscape-preferences
-rw-------   1 denise   users        104 Jul 11 11:01 .newsrc-news.spry.com
-rw-r--r--   1 denise   users        223 Sep 26 13:26 .newsrc-newv.hp.com
-r--r--r--   1 denise   users        446 May 19 10:10 .profile
-rw-------   1 denise   users         21 Jul  6 13:21 .rhosts
-rw-------   1 denise   users       2328 Oct 10 15:22 .sh_history
-rw-r--r--   1 denise   users       1052 Sep 22 15:00 .softbuildrc
-rwxrwxrwx   1 denise   users        161 Jul 11 12:19 .softinit.orig
drwxr-xr-x   3 denise   users       1024 Aug 31 15:44 .sw
-rw-------   1 denise   users         23 Jun  2 15:01 .xinitrc
-rwxr-xr-x   1 denise   users      11251 May 19 10:41 .xsession
-rw-r--r--   1 denise   users     611488 Oct  3 12:00 27247b.exe
-rw-r--r--   1 denise   users     114119 Sep 29 12:49 410pt1.exe
-rw-r--r--   1 denise   users     136979 Sep 29 12:53 410pt2.exe
-rw-r--r--   1 denise   users     173978 Sep 29 12:40 41ndir.exe
-rw-r--r--   1 denise   users     363315 Sep 29 12:52 41nds1.exe
-rw-r--r--   1 denise   users     527524 Sep 29 12:57 41nds4.exe
```

```
-rw-r--r--   1 denise   users    1552513 Sep 29 12:50 41nwad.exe
-rw-r--r--   1 denise   users     853424 Sep 29 12:24 41rtr2.exe
-rw-r--r--   1 denise   users    1363011 Sep 20 12:20 HPDA1.EXE
drwx------   2 denise   users         24 Jul 31 18:48 Mail
-rw-r--r--   1 denise   users    1787840 Aug 31 09:35 N3212B6.EXE
-rw-r--r--   1 denise   users      13543 Sep 23 09:46 SCSI4S.EXE
-rw-r--r--   1 denise   users      28395 Aug 30 15:07 cabview.exe
-rwx--x--x   1 denise   users         66 Jun  8 17:40 clean
-rwx--x--x   1 denise   users         99 Jun 20 17:44 clean2
-rwx--x--x   1 denise   users         66 Jun 20 17:51 clean3
-rw-r--r--   1 denise   users      15365 Aug 30 15:07 content.exe
-rw-r--r--   1 denise   users     713313 Sep 29 12:56 dsenh.exe
-rwx------   1 denise   users        144 Aug 14 17:10 eg1
-rwx------   1 denise   users        192 Aug 15 12:13 eg2
-rw-r--r--   1 denise   users     667890 Sep 20 12:41 en0316bz.exe
-rw-r--r--   1 denise   users     641923 Sep 20 12:42 en0316tb.exe
-rw-r--r--   1 denise   users       6251 Aug 30 15:07 explore.exe
-rw-r--r--   1 denise   users      23542 Aug 30 15:08 flexi_cd.exe
-rw-r--r--   1 denise   users         30 Aug 14 17:02 fred.h
-rw-r--r--   1 denise   users          0 Aug 14 17:24 hal.c
-rw-r--r--   1 denise   users     895399 Sep 20 12:32 hpdl0117.exe
-rw-r--r--   1 denise   users      14135 Sep 20 12:39 hpdlinst.txt
-rw-------   1 denise   users       2943 Jun 19 14:42 hpux.patches
-rw-r--r--   1 denise   users     680279 Sep 20 12:26 j2577a.exe
-rw-r--r--   1 denise   users     930728 Sep 20 15:16 ja95up.exe
-rw-r--r--   1 denise   users      53575 Oct 10 10:37 mbox
-rw-r--r--   1 denise   users    1097728 Aug 30 15:03 msie10.exe
-rw-r--r--   1 denise   users    1790376 Sep 18 14:32 n32e12n.exe
-rw-r--r--   1 denise   users    1393835 Sep 29 12:59 nfs197.exe
-rw-------   1 denise   users        977 Jul  3 14:25 pass.sb
-rw-r--r--   1 denise   users    1004544 Aug 30 15:00 plusdemo.exe
-rw-r--r--   1 denise   users     229547 Sep 29 12:27 ps4x03.exe
-rwxr--r--   1 denise   users        171 Aug  9 13:43 psg
-rw-r--r--   1 denise   users      16645 Aug 30 15:08 quik_res.exe
-rw-r--r--   1 denise   users      14544 Aug 30 15:08 rclock.exe
-rw-r--r--   1 denise   users    2287498 Aug 30 15:12 rkhelp.exe
-rw-r--r--   1 denise   users          0 Aug 15 12:10 roni.mak
-rw-r--r--   1 denise   users       1139 Sep 28 10:35 sb.txt
-rw-r--r--   1 denise   users     569855 Sep 29 12:55 smsup2.exe
-rw-------   1 root     sys          161 Jul 11 12:18 softinit.remotesoftcm
-rw-r--r--   1 denise   users         39 Sep 29 12:48 srvpr.exe
-rw-r--r--   1 denise   users         38 Aug 15 12:14 steve.h
-rw-r--r--   1 denise   users      14675 Aug 30 15:08 target.exe
-rw-r--r--   1 denise   users     229630 Sep 29 12:54 tcp41a.exe
-rw-r--r--   1 denise   users    1954453 Sep 29 12:26 tnds2.exe
-rw-r--r--   1 denise   users     364270 Sep 23 09:50 upgrade.exe
-rwx-----x   1 denise   users         88 Aug  9 13:43 whoon
-rw-r--r--   1 denise   users     191495 Aug 30 15:00 win95app.exe
total 46718
```

Because I find this to be the most commonly used option with the **ls** command, I describe each of the fields produced by **ls -l**. I'll use the earlier example of the **ls -l** command, which showed only one file when describing the fields:

```
$ ls -l sort
-rwxr-x--x   1 marty     users     120 Jul 26 10:20 sort
```

The first field defines the access rights of the file, which I cover in "Permissions" in the next chapter. The *owner* has read, write, and execute permissions on the file. The *group* has read and execute permissions on the file. *other* has execute permissions.

The second field is the link count. This lists how many files are symbolically linked to the file. We will get into the details of the **ln** command used to link files later. In this case, the link count is *1*, which means that this file is linked only to itself. For directories such as **.dt** shown below, the number of subdirectories is shown rather than the link count. This number includes one for the directory itself as well as one for the parent directory. This means that a total of five directories are below **.dt**.

```
drwxr-x--x   7 denise      users    1024 Jul 26 10:20 .dt
```

The subdirectories below **/home/denise/.dt** are:

```
/home/denise/.dt/Desktop
```

/home/denise/.dt/appmanager

/home/denise/.dt/palettes

/home/denise/.dt/sessions

/home/denise/.dt/types

These five subdirectories plus the directory itself and the parent directory make a total of seven.

The third field lists the owner of the file. Your login name, such as *denise*, is listed here. When you create a file, your login name is listed by default as the owner of the file.

The fourth field lists the group to which the file belongs. Groups are covered in the next chapter in "Permissions."

The fifth field shows the size of the file. The file **sort** is *120* bytes in size.

The sixth field (which includes a date and time such as *Jul 26 10:20*) lists the date and time the file was created or last changed.

The seventh field lists the files and directories in alphabetical order. You first see the files that begin with a ".", then the files that begin with numbers, then the files that begin with uppercase letters, and finally the files that begin with lower case letters. There are a lot of characters that a file can begin with in UNIX, so if you perform an **ls -l** and don't see the file you are looking for, it may appear at a different spot in the listing from what you expected.

ls -i

To get information about the inode of a file, you use the **-i** option to **ls**. The following example includes both the *-i* and *-l* options to **ls**:

```
$ls -ail /home/denise

137717 -rw-------   1 denise   users         98 Oct  6 09:19 .Xauthority
137623 -r--r--r--   1 denise   users        814 May 19 10:10 .cshrc
140820 drwxr-xr-x   7 denise   users       1024 Sep 26 11:14 .dt
137629 -rwxr-xr-x   1 denise   users       8705 Jul  7 12:04 .dtprofile
180815 drwx------   2 denise   users       1024 Jul 31 18:48 .elm
137624 -r--r--r--   1 denise   users        347 May 19 10:10 .exrc
137652 -rwxrwxrwx   1 denise   users        170 Jun  6 14:20 .fmrc.orig
137650 -rw-------   1 denise   users         97 Jun 12 18:59 .glancerc
137699 -rw-------   1 denise   users      17620 Sep 21 16:11 .gpmhp
137640 -rwxr-xr-x   1 denise   users        391 Sep 19 09:55 .history
137625 -r--r--r--   1 denise   users        341 May 19 10:10 .login
185607 drwx--x--x   2 denise   users       1024 Jul 31 18:48 .lrom
137642 -rw-r--r--   1 denise   users        768 Jul 28 12:54 .mailrc
137641 -rw-------   1 denise   users       1450 Oct  6 13:58 .netscaperks.html
179207 drwx------   2 denise   users      10240 Oct 10 15:24 .netscape-cache
137656 -rw-------   1 denise   users         91 Sep 18 14:16 .netscape-cookies
137635 -rw-------   1 denise   users      43906 Oct 10 15:32 .netscape-history
137645 -rw-r--r--   1 denise   users        566 Aug 25 14:36 .netscapes.spry.com
137646 -rw-------   1 denise   users      46514 Jun 28 12:35 .netsca
137634 -rw-------   1 denise   users       1556 Sep 28 15:02 .netscaperences
137637 -rw-------   1 denise   users        104 Jul 11 11:01 .newsrcws.spry.com
137633 -rw-r--r--   1 denise   users        223 Sep 26 13:26 .newsrc-hp.com
137626 -r--r--r--   1 denise   users        446 May 19 10:10 .profile
137649 -rw-------   1 denise   users         21 Jul  6 13:21 .rhosts
137694 -rw-------   1 denise   users       2328 Oct 10 15:22 .sh_history
137698 -rw-r--r--   1 denise   users       1052 Sep 22 15:00 .softbuildrc
137636 -rwxrwxrwx   1 denise   users        161 Jul 11 12:19 .softinit.orig
 33600 drwxr-xr-x   3 denise   users       1024 Aug 31 15:44 .sw
137648 -rw-------   1 denise   users         23 Jun  2 15:01 .xinitrc
137628 -rwxr-xr-x   1 denise   users      11251 May 19 10:41 .xsession
137715 -rw-r--r--   1 denise   users     611488 Oct  3 12:00 27247b.exe
137707 -rw-r--r--   1 denise   users     114119 Sep 29 12:49 410pt1.exe
137710 -rw-r--r--   1 denise   users     136979 Sep 29 12:53 410pt2.exe
137705 -rw-r--r--   1 denise   users     173978 Sep 29 12:40 41ndir.exe
137709 -rw-r--r--   1 denise   users     363315 Sep 29 12:52 41nds1.exe
```

```
137714 -rw-r--r--   1 denise   users     527524 Sep 29 12:57 41nds4.exe
137708 -rw-r--r--   1 denise   users    1552513 Sep 29 12:50 41nwad.exe
137696 -rw-r--r--   1 denise   users     853424 Sep 29 12:24 41rtr2.exe
137654 -rw-r--r--   1 denise   users    1363011 Sep 20 12:20 HPDA1.EXE
182429 drwx------   2 denise   users         24 Jul 31 18:48 Mail
137683 -rw-r--r--   1 denise   users    1787840 Aug 31 09:35 N3212B6.EXE
137702 -rw-r--r--   1 denise   users      13543 Sep 23 09:46 SCSI4S.EXE
137638 -rwx--x--x   1 denise   users         66 Jun  8 17:40 clean
137651 -rwx--x--x   1 denise   users         99 Jun 20 17:44 clean2
137632 -rwx--x--x   1 denise   users         66 Jun 20 17:51 clean3
137688 -rw-r--r--   1 denise   users      15365 Aug 30 15:07 content.exe
137713 -rw-r--r--   1 denise   users     713313 Sep 29 12:56 dsenh.exe
137667 -rwx------   1 denise   users        144 Aug 14 17:10 eg1
137671 -rwx------   1 denise   users        192 Aug 15 12:13 eg2
137662 -rw-r--r--   1 denise   users     667890 Sep 20 12:41 en0316bz.exe
137665 -rw-r--r--   1 denise   users     641923 Sep 20 12:42 en0316tb.exe
137689 -rw-r--r--   1 denise   users       6251 Aug 30 15:07 explore.exe
137690 -rw-r--r--   1 denise   users      23542 Aug 30 15:08 flexi_cd.exe
137670 -rw-r--r--   1 denise   users         30 Aug 14 17:02 fred.h
137673 -rw-r--r--   1 denise   users          0 Aug 14 17:24 hal.c
137660 -rw-r--r--   1 denise   users     895399 Sep 20 12:32 hpdl0117.exe
137661 -rw-r--r--   1 denise   users      14135 Sep 20 12:39 hpdlinst.txt
137647 -rw-------   1 denise   users       2943 Jun 19 14:42 hpux.patches
137659 -rw-r--r--   1 denise   users     680279 Sep 20 12:26 j2577a.exe
137697 -rw-r--r--   1 denise   users     930728 Sep 20 15:16 ja95up.exe
137684 -rw-r--r--   1 denise   users    1097728 Aug 30 15:03 msie10.exe
137658 -rw-r--r--   1 denise   users    1790376 Sep 18 14:32 n32e12n.exe
137643 -rw-r--r--   1 denise   users    1393835 Sep 29 12:59 nfs197.exe
137639 -rw-------   1 denise   users        977 Jul  3 14:25 pass.sb
137664 -rw-r--r--   1 denise   users    1004544 Aug 30 15:00 plusdemo.exe
137704 -rw-r--r--   1 denise   users     229547 Sep 29 12:27 ps4x03.exe
137666 -rwxr--r--   1 denise   users        171 Aug  9 13:43 psg
137691 -rw-r--r--   1 denise   users      16645 Aug 30 15:08 quik_res.exe
137692 -rw-r--r--   1 denise   users      14544 Aug 30 15:08 rclock.exe
137693 -rw-r--r--   1 denise   users    2287498 Aug 30 15:12 rkhelp.exe
137669 -rw-r--r--   1 denise   users          0 Aug 15 12:10 roni.mak
137657 -rw-r--r--   1 denise   users       1139 Sep 28 10:35 sb.txt
137712 -rw-r--r--   1 denise   users     569855 Sep 29 12:55 smsup2.exe
137644 -rw-------   1 root     sys          161 Jul 11 12:18 softinittesoftcm
137706 -rw-r--r--   1 denise   users         39 Sep 29 12:48 srvpr.exe
137682 -rw-r--r--   1 denise   users         38 Aug 15 12:14 steve.h
137685 -rw-r--r--   1 denise   users      14675 Aug 30 15:08 target.exe
137711 -rw-r--r--   1 denise   users     229630 Sep 29 12:54 tcp41a.exe
137700 -rw-r--r--   1 denise   users    1954453 Sep 29 12:26 tnds2.exe
137703 -rw-r--r--   1 denise   users     364270 Sep 23 09:50 upgrade.exe
137663 -rwx-----x   1 denise   users         88 Aug  9 13:43 whoon
137678 -rw-r--r--   1 denise   users     191495 Aug 30 15:00 win95app.exe
```

The inode number contains the following: the location of files and directories on the disk; access permissions; owner and group IDs; file link count; time of last modification; time of last access; device identification number for special files; and a variety of other information. inode numbers are used extensively by the system as you change directories and perform various tasks.

ls -p

Because you may have subdirectories within the directory you are listing, you may want to use the *-p* option to **ls,** which puts a "/" (slash) in after directory names, as shown in the following example:

```
$ ls -ap /home/denise

.Xauthority
.cshrc
.dt/
.dtprofile
.elm/
.exrc
.fmrc.orig
.glancerc
.gpmhp
.history
.login
.lrom/
```

```
.mailrc
.netscape-bookmarks.html
.netscape-cache/
.netscape-cookies
.netscape-history
.netscape-newsgroups-news.spry.com
.netscape-newsgroups-newsserv.hp.com
.netscape-preferences
.newsrc-news.spry.com
.newsrc-newsserv.hp.com
.profile
.rhosts
.sh_history
.softbuildrc
.softinit.orig
.sw/
.xinitrc
.xsession
27247b.exe
410pt1.exe
410pt2.exe
41ndir.exe
41nds1.exe
41nds4.exe
41nwad.exe
41rtr2.exe
HPDA1.EXE
Mail/
N3212B6.EXE
SCSI4S.EXE
clean
clean2
clean3
content.exe
dsenh.exe
eg1
eg2
en0316bz.exe
en0316tb.exe
explore.exe
flexi_cd.exe
fred.h
hal.c
hpdl0117.exe
hpdlinst.txt
hpux.patches
j2577a.exe
ja95up.exe
msie10.exe
n32e12n.exe
nfs197.exe
pass.sb
plusdemo.exe
ps4x03.exe
psg
quik_res.exe
rclock.exe
rkhelp.exe
roni.mak
sb.txt
smsup2.exe
softinit.remotesoftcm
srvpr.exe
steve.h
target.exe
tcp41a.exe
tnds2.exe
upgrade.exe
whoon
win95app.exe
```

ls -R

Because the subdirectories you are listing probably have files and subdirectories beneath them, you may want to recursively list these. The *-R* option to **ls** shown in the following example performs this recursive listing. This listing is truncated because it would be too long if it included all the subdirectories under **/home/denise**:

```
$ ls -aR /home/denise

.Xauthority
.cshrc
.dt
.dtprofile
.elm
.exrc
.fmrc.orig
.glancerc
.gpmhp
.history
.login
.lrom
.mailrc
.netscape-bookmarks.html
.netscape-cache
.netscape-cookies
.netscape-history
.netscape-newsgroups-news.spry.com
.netscape-newsgroups-newsserv.hp.com
.netscape-preferences
.newsrc-news.spry.com
.newsrc-newsserv.hp.com
.profile
.rhosts
.sh_history
.softbuildrc
.softinit.orig
.sw
.xinitrc
.xsession
27247b.exe
410pt1.exe
410pt2.exe
41ndir.exe
41nds1.exe
41nds4.exe
41nwad.exe
41rtr2.exe
HPDA1.EXE
Mail
N3212B6.EXE
SCSI4S.EXE
clean
clean2
clean3
content.exe
dsenh.exe
eg1
eg2
en0316bz.exe
en0316tb.exe
explore.exe
flexi_cd.exe
fred.h
                .
                .                                    (skip some of the listing)
                .


/home/denise/.lrom:
LRAAAa27637.CC
LRBAAa27637.CC
LROM.AB
LROM.AB.OLD
LROM.AB.lk1
LROM.SET
```

```
/home/denise/.netscape-cache:
cache306C05510015292.gif
cache306C05560025292.gif
cache306C05560035292.gif

/home/denise/.sw:
sessions

                    .
                    .                                   (skip remainder of the listing)
                    .
```

ls Summary

I have shown you what I believe to be the most important, and most often used, **ls** options. Because you may have future needs to list files and directories based on other criteria, I provide you with a list of most **ls** options. There is no substitute, however, for issuing the **man ls** command. Whatever I provide is only a summary. Viewing the man pages for **ls** gives you much more information. The following is a summary of the more commonly used **ls** options:

ls - List the contents of a directory

Options

-a	List all entries.
-b	Print non-graphic characters.
-c	Use the time the file was last modified for producing order in which files are listed.
-d	List only the directory name, not its contents.
-f	Assume that each argument is a directory.
-g	Only the group is printed and not the owner.
-i	Print the inode number in the first column of the report.
-m	List the contents across the screen, separated by commas.
-n	Numbers for UID and GID are printed instead of names.
-o	List the information in long form (-l), except that group is omitted.
-p	Put a slash (/) at the end of directory names.
-q	Non-printing characters are represented by a "?".
-r	Reverse the order in which files are printed.
-s	Show the size in blocks instead of bytes.
-t	List in order of time saved, with most recent first.
-u	Use the time of last access instead of last modification for determining order in which files are printed.
-x	List files in multi-column format as shown in examples.
-A	Same as -a, except that current and parent directories aren't listed (only used on some UNIX variants).
-C	Multicolumn output produced.

-F Directory followed by a "/", executable by an "*", symbolic link by an "@", and FIFO by a "|".

-L List file or directory to which link points.

-R Recursively list subdirectories.

-1 Output will be listed in single-column format.

Also, some shorthand command names are available for issuing **ls** with options. For instance, **ll** is equivalent to **ls -l**, and **lsr** is equivalent to **ls -R** in some UNIX variants. We will also cover creating an "alias," whereby you can define your own shorthand for any command in an upcoming chapter.

[Team LiB]

PREVIOUS NEXT

File System Layout

Before I begin describing the file system layout, you need to know that you do not necessarily have a single file system. Your system administrator may have set up a variety of file system types on your system. As a beginning user, you don't care too much about the different types of file systems; however, before proceeding to the file system layout, let me briefly cover some of the different file system types. Your system administrator cares a lot about the different file system types because the commands that he or she issues allow them to specify an option such as "-F" followed by the file system type. One good place to start when looking for the different file system types supported by your UNIX variant is the manual page for the **mount** command. The "-F" option, or another option such as "-t" for type in Linux, is usually followed by a list of file system types supported. The different file system types and command options to file system types depend on the UNIX variant that you are using. Some commands that commonly support a file system type option are **dcopy, fsck, mksf, mount, newfs**, and others, some of which you may need to know as an advanced user if you are going to perform system administration in the future. The following is a description of some of the commonly supported file systems on UNIX variants:

- Personal Computer File System (PCFS) is a file system type that allows direct access to PC formatted disks.
- UNIX File System (UFS) is the standard or basic default UNIX file system on some UNIX variants.
- CD-ROM File System (CDFS) is used when you mount a CD-ROM. Most CD-ROMs are read-only, so you can't write to them. This is called High Sierra File System (HSFS) on some UNIX variants.
- Network File System (NFS) is a way of accessing files on other systems on the network from your local system. An NFS mounted file system looks as though it is local to your system even though it is located on another system.
- Loopback File System (LOFS) allows you to have the same file system in multiple places using alternate path names.
- VxFS is an extent-based Journal File System that supports fast file system recovery and on-line features such as backup on HP-UX.
- High Performance File System (HFS) is HP's version of the UNIX File System. This is used in most of the examples.
- TMPFS is a memory-based file system.
- CacheFS is a file system present in cache.

File system types is an area of high customization on UNIX variants. You will find some of the file systems listed on some UNIX variants and others on other UNIX variants. Although UNIX and its associated commands are very similar, going from one UNIX variant to another, file system types are generally peculiar to a specific UNIX variant. A good place to start when determining what file system types are included with your UNIX variant is to view the file system types listed in the manual page for **mount**.

The manual page for **mount** on the Caldera Linux system we are using lists many file system types. With the Linux **mount** command, you would use the *-t vfstype* option to specify one of the file system types to mount:

- minix
- ext
- ext2

- xiafs
- hpfs
- fat
- msdos
- umsdos
- vfat
- proc
- nfs
- iso9660
- smb
- ncp
- affs
- ufs
- sysv
- xenix
- coherent

One common file system I often mount on Linux systems is a DOS floppy disk. For example, we would go through the following sequence of events to mount a DOS floppy, copy a file to it, and unmount it:

```
# mount -t msdos /dev/fd0 /mnt/floppy
# cp * /mnt/floppy
# ls /mnt/floppy
file1           file2     file3     file4
# umount /dev/fd0
```

This sequence of commands first mounts **/dev/fd0**, which is the floppy disk device file, under the mount point **/mnt/floppy. /dev/fd0** is mounted as type *msdos,* as specified by the *-t msdos*. I next copy all files in the current directory to the floppy. All the files on the floppy are then listed with **ls,** producing the list of four files shown. I then unmount the floppy disk with the **umount** command so that I can take the floppy to a DOS system and read the files. Because the floppy was mounted as type *msdos,* the files were written to the floppy in DOS format.

Similarly, to mount a CD-ROM on a Linux system you would issue the following **mount** command, unless it's mounted at boot:

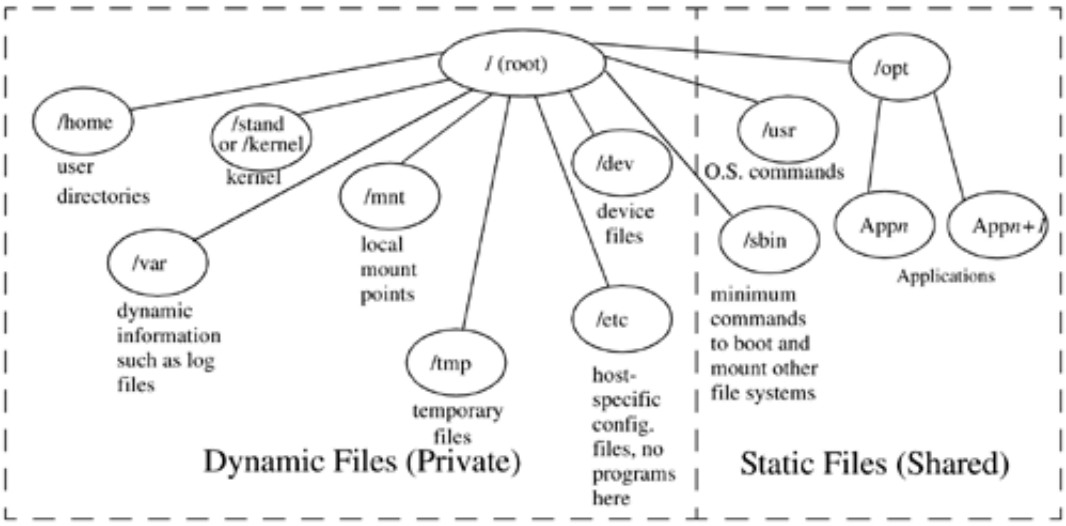
```
# mount /dev/hdd /cdrom
mount: block device /dev/hdd is write-protected, mounting read-only
#
```

You would substitute your CD-ROM device file for **/dev/hdd**. This CD-ROM is, of course, a read-only device, as the message from **mount** indicates.

The file system layout of most UNIX variants is based on the AT&T SVR4 layout. This means that going from one UNIX variant to the next, you see pretty much the same names used in the file system layouts.

Figure 20-1 is a high-level depiction of the file system.

Figure 20-1. General UNIX File System Layout (differs somewhat among UNIX variants)



Here are some of the more important features of the UNIX file system layout:

- Files and directories are organized by category. The two most obvious categories that appear in Figure 20-1 are static vs. dynamic files. Also, other categories are executable, configuration, data files, and so on. The static files are also labeled "shared," because other hosts on the network may share these. The directories **/usr, /sbin**, and **/opt** are shared directories.

- The operating system (OS) and applications are kept separate from one another. Application vendors don't care where their applications are loaded; that is up to you. But to a system administrator, it is highly desirable to keep applications separate from the operating system so you don't inadvertently have application files overwriting operating system files. In addition, if applications are loaded in a separate area, they are "modular," meaning that a system administrator can add, remove, and modify them without affecting the operating system or other applications. Applications are kept in the **/opt** directory.

- Intra-system files are kept in a separate area from inter-system, or network-accessible files. **/usr** and **/sbin** are shared operating system directories. No host-specific information is in these two directories. **/etc** is used to hold the host-specific configuration files.

- Executable files are kept separate from system configuration files so that the executables may be shared among hosts. Having the configuration files separate from the programs that use them also means that updates to the operating system won't affect the configuration files.

I provide descriptions of some of the most important directories of the file system layout and their contents:

/	This is the root directory, which is the base of the file system's hierarchical tree structure. A directory is logically viewed as being part of **/**. Regardless of the disk on which a directory or logical volume is stored, it is logically viewed as a part of the root hierarchy.
/dev	Contains host-specific device files.
/etc	Contains host-specific system and application configuration files. The information in this directory is important to the operation of the system and is of a permanent nature. There are also additional configuration directories below **/etc**. There are two **/etc** subdirectories of particular interest:
/export	Servers export root directories for networked clients. For instance, a server might export user directories such as **/export/home/***username*.
/home	Users' home directories are located here. Because the data stored in users' home

directories will be modified often, you can expect this directory to grow in size.

/kernel Contains kernel configuration and binary files that are required to bring up a system such as genunix.

/lost+found This is the lost files directory. Here you find files that are in use but are not associated with a directory. These files typically become "lost" as a result of a system crash that caused the link between the physical information on the disk and the logical directory to be severed. The program **fsck**, which is run at the time of boot, finds these files and places them in the **lost+found** directory.

/mnt This directory is reserved as a mount point for local file systems. You can either mount directly to **/mnt** or have **/mnt** subdirectories as mount points such as **/mnt1, /mnt2, /mnt3**, etc.

/net Name reserved as mount points for remote file systems.

/opt The directory under which applications are installed. As a rule, application vendors never specify a particular location for their applications to be installed. Now, with **/opt**, there is a standard directory under which applications should be installed. This is an organizational improvement for system administrators because they can now expect applications to be loaded under **/opt** and the application name.

/sbin Contains commands and scripts used to boot, shut down, and fix file system mounting problems. /sbin is available when a system boots, because it contains commands required to bring up a system.

/stand Contains kernel configuration and binary files that are required to bring up a system. Two significant files contained in this directory are the system and vmunix (kernel) files.

/tmp This is a free-for-all directory, where any user can *temporarily store* files. Because of the loose nature of this directory, it should not be used to store anything important, and users should know that whatever they have stored in **/tmp** can be deleted without notice. Application working files should go in **/var/tmp** or **/var/opt/appname**, not in **/tmp**.

/usr Most of the UNIX operating system is contained in **/usr**. Included in this directory are commands, libraries, and documentation. A limited number of subdirectories can appear in **/usr**.

/var Holds files that are primarily temporary. Files such as log files, which are frequently deleted and modified, are stored here. Think of this as a directory of "variable" size. Files that an application or command create at runtime should be placed in this directory, including logfile and spool files. However, some applications may store state information in **/var**.

[Team LiB]

PREVIOUS NEXT



Linux File System Layout

The Linux file system layout is very similar to the other UNIX variants described earlier, both in concept and in implementation. Files are contained in directories, and directories can have any number of subdirectories. Most operating systems are arranged this way, including UNIX.

The following is a long listing at the root level of a Caldera Linux system:

```
# ls -l /
total 439
-rw-rw-rw-   1 root     root        23 Nov  5 07:10 .lgdb
dr-xr-xr-x   3 root     root      1024 Nov  5 07:10 amd
dr-xr-xr-x   2 root     root       512 Nov  5 07:10 auto
drwxr-xr-x   2 root     root      2048 Nov  5 07:12 bin
drwxr-xr-x   2 root     root      1024 Nov  5 01:44 boot
drwxr-xr-x   2 root     root     10240 Nov  6 22:27 dev
drwxr-xr-x  15 root     root      2048 Nov  7 00:15 etc
drwxr-xr-x   5 root     root      1024 Nov  5 01:43 home
drwxr-xr-x   2 root     root      1024 Nov  5 01:29 initrd
lrwxrwxrwx   1 root    root        11 Nov  5 01:29 install ->
                                                  /var/lib/LST
drwxr-xr-x   4 root     root      1024 Nov  5 01:44 lib
drwxr-xr-x   2 root     root     12288 Nov  5 01:29 lost+found
drwxr-xr-x   4 root     root      1024 Nov  5 01:44 mnt
drwxr-xr-x   4 root     root      1024 Sep 10  1996 opt
dr-xr-xr-x   5 root     root         0 Nov  5 01:59 proc
drwxr-xr-x   5 root     root      1024 Nov  6 20:56 root
drwxr-xr-x   2 root     root      2048 Nov  7 00:03 sbin
drwxrwxrwt   6 root     root      1024 Nov  7 00:10 tmp
drwxr-xr-x  21 root     root      1024 Nov  5 01:40 usr
drwxr-xr-x  14 root     root      1024 Nov  5 01:32 var
-rw-r--r--   1 root     root    404158 Nov  5 01:29 vmlinuz
```

Most of the directories are the same as those described earlier. A couple of directory names did not appear earlier that are significant, however. The Linux kernel, **vmlinuz**, appears at the root level. The directory **/root** is the home directory of the user *root.*

The **/proc** directory contains information about the Linux system. The **/proc** directory is really a set of data structure information that looks like a directory. The following is a long listing of the **/proc** directory:

```
# ls -l /proc
total 0
dr-xr-xr-x   3 root     root            0 Nov  7 00:13 1
dr-xr-xr-x   3 root     root            0 Nov  7 00:16 1011
dr-xr-xr-x   3 root     root            0 Nov  7 00:13 104
dr-xr-xr-x   3 bin      root            0 Nov  7 00:13 106
dr-xr-xr-x   3 root     root            0 Nov  7 00:13 116
dr-xr-xr-x   3 root     root            0 Nov  7 00:13 118
dr-xr-xr-x   3 root     root            0 Nov  7 00:13 146
dr-xr-xr-x   3 root     root            0 Nov  7 00:13 155
dr-xr-xr-x   3 root     root            0 Nov  7 00:13 156
dr-xr-xr-x   3 daemon   root            0 Nov  7 00:13 162
dr-xr-xr-x   3 root     root            0 Nov  7 00:13 167
dr-xr-xr-x   3 root     root            0 Nov  7 00:13 170
dr-xr-xr-x   3 root     65535           0 Nov  7 00:13 178
dr-xr-xr-x   3 root     root            0 Nov  7 00:13 18
dr-xr-xr-x   3 root     root            0 Nov  7 00:13 19
dr-xr-xr-x   3 nobody   65535           0 Nov  7 00:13 193
dr-xr-xr-x   3 root     root            0 Nov  7 00:13 199
dr-xr-xr-x   3 root     root            0 Nov  7 00:13 2
dr-xr-xr-x   3 root     root            0 Nov  7 00:13 20
dr-xr-xr-x   3 root     root            0 Nov  7 00:13 200
dr-xr-xr-x   3 root     root            0 Nov  7 00:13 201
dr-xr-xr-x   3 root     root            0 Nov  7 00:13 202
dr-xr-xr-x   3 root     root            0 Nov  7 00:13 203
dr-xr-xr-x   3 root     root            0 Nov  7 00:13 204
dr-xr-xr-x   3 root     root            0 Nov  7 00:13 205
dr-xr-xr-x   3 root     root            0 Nov  7 00:13 21
dr-xr-xr-x   3 root     root            0 Nov  7 00:13 273
```

```
dr-xr-xr-x   3 root     root            0 Nov  7 00:13 274
dr-xr-xr-x   3 root     root            0 Nov  7 00:13 276
dr-xr-xr-x   3 root     root            0 Nov  7 00:13 3
dr-xr-xr-x   3 root     root            0 Nov  7 00:13 307
dr-xr-xr-x   3 root     root            0 Nov  7 00:13 327
dr-xr-xr-x   3 root     root            0 Nov  7 00:13 358
dr-xr-xr-x   3 root     root            0 Nov  7 00:13 375
dr-xr-xr-x   3 root     root            0 Nov  7 00:13 443
dr-xr-xr-x   3 root     root            0 Nov  7 00:13 444
dr-xr-xr-x   3 root     root            0 Nov  7 00:13 445
dr-xr-xr-x   3 root     root            0 Nov  7 00:13 45
dr-xr-xr-x   3 root     root            0 Nov  7 00:13 482
dr-xr-xr-x   3 root     root            0 Nov  7 00:13 483
dr-xr-xr-x   3 root     root            0 Nov  7 00:16 996
dr-xr-xr-x   3 root     root            0 Nov  7 00:16 997
dr-xr-xr-x   3 root     root            0 Nov  7 00:16 998
-r--r--r--   1 root     root            0 Nov  7 00:13 cmdline
-r--r--r--   1 root     root            0 Nov  7 00:13 cpuinfo
-r--r--r--   1 root     root            0 Nov  7 00:13 devices
-r--r--r--   1 root     root            0 Nov  7 00:13 dma
-r--r--r--   1 root     root            0 Nov  7 00:13 filesys
-r--r--r--  1 root     root           0 Nov  7 00:13 interrupt
-r--r--r--   1 root     root            0 Nov  7 00:13 ioports
-r--------   1 root     root     33558528 Nov  7 00:13 kcore
-r--------   1 root     root            0 Nov  5 07:00 kmsg
-r--r--r--   1 root     root            0 Nov  7 00:13 ksyms
-r--r--r--   1 root     root            0 Nov  7 00:05 loadavg
-r--r--r--   1 root     root            0 Nov  7 00:13 locks
-r--r--r--   1 root     root            0 Nov  7 00:13 mdstat
-r--r--r--   1 root     root            0 Nov  7 00:13 meminfo
-r--r--r--   1 root     root            0 Nov  7 00:13 modules
-r--r--r--   1 root     root            0 Nov  7 00:13 mounts
dr-xr-xr-x   2 root     root            0 Nov  7 00:13 net
-r--r--r--   1 root     root            0 Nov  7 00:13 pci
dr-xr-xr-x   2 root     root            0 Nov  7 00:13 scsi
lrwxrwxrwx   1 root     root           64 Nov  7 00:13 self->110
-r--r--r--   1 root     root            0 Nov  7 00:13 stat
dr-xr-xr-x   5 root     root            0 Nov  7 00:13 sys
-r--r--r--   1 root     root            0 Nov  7 00:13 uptime
-r--r--r--   1 root     root            0 Nov  7 00:13 version
```

The directories in the long listing, which have a "*d*" at the beginning of the line, are processes. The file **/proc/kcore** represents the physical memory of your Linux system. Files in **/proc** contain some interesting information about your system. One file you'll want to take a look at is the **/proc/cpuinfo** file. An example of this file is shown in the following listing:

```
# more /proc/cpuinfo
processor      : 0
cpu            : 686
model          : 3
vendor_id      : GenuineIntel
stepping       : 4
fdiv_bug       : no
hlt_bug        : no
fpu            : yes
fpu_exception  : yes
cpuid          : yes
wp             : yes
flags          : fpu vme de pse tsc msr pae mce cx8 11 mtrr pge mca cmov mmx
bogomips       : 266.24
```

more is covered in the next chapter. For now, you just need to know that **more** is used for viewing files.

The **/boot** directory contains the Linux kernel and other files used by the LILO boot manager. The following is a listing of **/boot**:

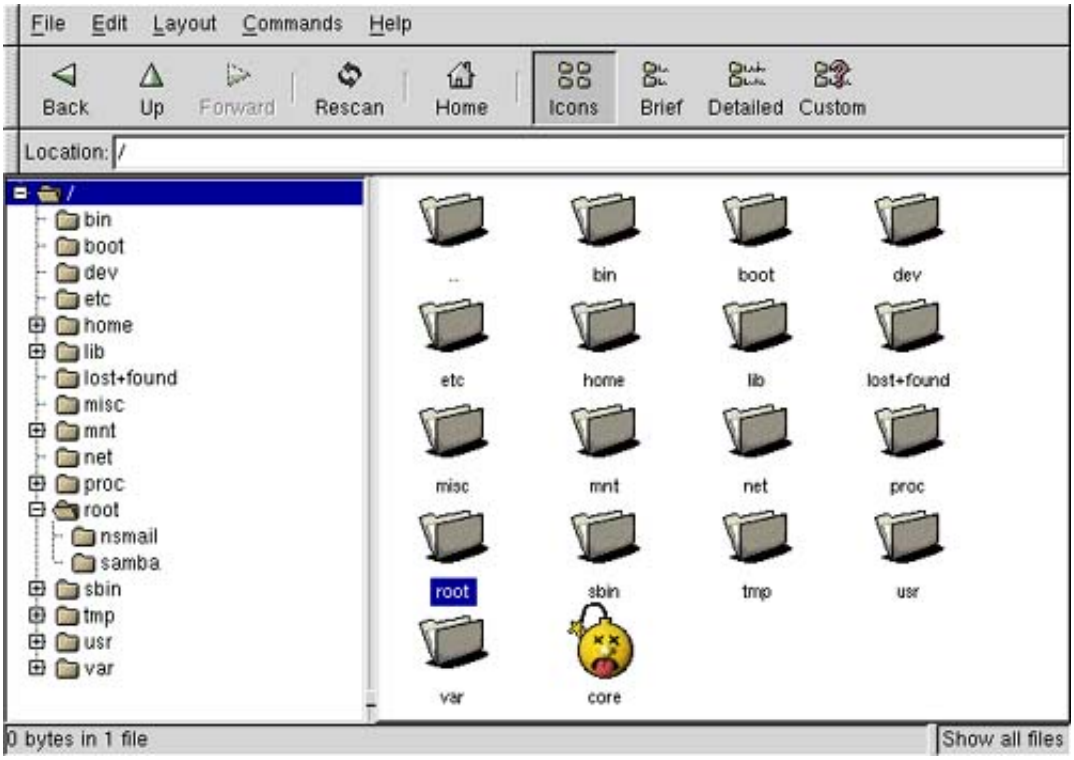
```
# ls -l /boot
total 452
-rw-r--r--   1 root     root     15954 Feb 11  1997 WHATSIN
-rw-r--r--   1 root     root     15954 Feb 11  1997
                                      WHATSIN-2.0.29-modular
-rw-r--r--   1 root     root       204 Jul 24  1996 any_b.b
-rw-r--r--   1 root     root       204 Jul 24  1996 any_d.b
-rw-r--r--   1 root     root       512 Nov  5 01:44 boot.0300
-rw-r--r--   1 root     root      4416 Jul 24  1996 boot.b
```

```
-rw-r--r--   1 root     root          88 Jul 24  1996 chain.b
-rw-------   1 root     root        7680 Nov  5 01:44 map
-r--r--r--   1 root     root        1565 Mar  7  1997 message
-rw-r--r--   1 root     root         192 Jul 24  1996 os2_d.b
-rw-r--r--   1 root     root      404158 Feb 11  1997 vmlinuz-
                                                     2.0.29-modular
```

As a user, you are probably most concerned with your home directory, assigned to you by your system administrator, which is most likely in the **/home** directory. I think, however, that as a UNIX user you need to understand the overall UNIX file system layout. Keep in mind that you may find minor differences in the file system layout going from one UNIX variant to another, but in general the layout is similar.

If you're running a graphical user interface on your Linux system, you can view the file system graphically. Figure 20-2 shows the Red Hat Linux *File Manager* which is part of the *Gnome* user interface.

Figure 20-2. *File Manager* in *Gnome* Showing the Red Hat Linux File System



You can see that many of the directories shown on the left-hand side of Figure 20-2 are the same as those described earlier for UNIX systems in general. Although you may be working with any number of different UNIX variants, you'll find most of the system structures to be similar.



Chapter 21. Working with Files and Directories - Permissions, Commands, File Name Expansion, and Wild Cards

Introduction

Permissions

Absolute and Relative Path Names

File Name Expansion and Wild Cards

pwd and cd

chmod

cp

mv

mkdir

rm

rmdir

Using Commands





Introduction

A variety of topics related to UNIX file system strucutre, navigation, and daily use are covered in this chapter, including:

- Permissions
- Absolute and relative path names
- File name expansion and wildcards
- Several commands, including: **pwd**, **cd**, **chmod**, **cp**, **mv**, **mkdir**, **rm**, and **rmdir**
- Examples of using many commands



Permissions

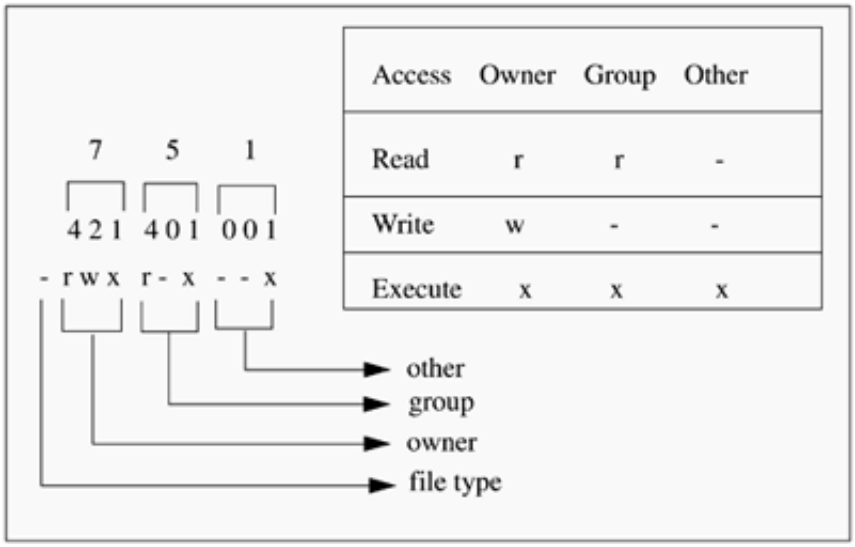
The best place to begin discussing permissions is by issuing the **ls** command, which lists the contents of directories. Permissions are the means by which files and directories are made secure on your UNIX system. Because UNIX is multi-user, potentially thousands of users could be accessing the files on a system. Permissions controls who has access to what files.

Here is an example **ls -l** command and output:

```
$ ls -l sort
-rwxr-x--x   1 marty     users     120 Jul 26 10:20 sort
```

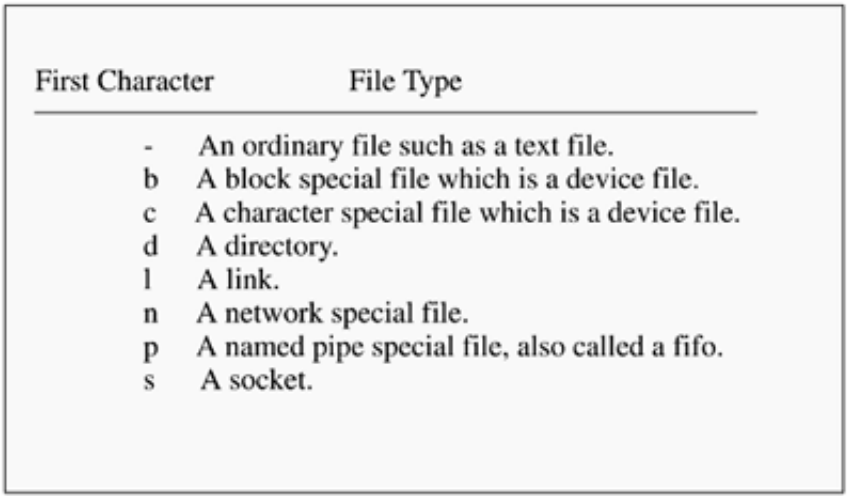
Issuing this command has produced a lot of information relating to a file called **sort**. Let's begin to understand what this listing has produced by analyzing the first set of characters (-rwxr-x--x). This set of characters is made up of four distinct fields, as shown in Figure 21-1.

Figure 21-1. Permissions for File sort



The first character in this group is related to the file type. I covered some file types earlier, but the **ls -l** command does not analyze files to the same level of detail. Among the types of files that **ls -l** will list are shown in Figure 21-2:

Figure 21-2. File Types of the ls Command



First Character	File Type
-	An ordinary file such as a text file.
b	A block special file which is a device file.
c	A character special file which is a device file.
d	A directory.
l	A link.
n	A network special file.
p	A named pipe special file, also called a fifo.
s	A socket.

Keep in mind that file types can vary slightly from one UNIX variant to another. The file types listed in Figure 21-2 are common to most UNIX variants. For every file on the system, UNIX supports three classes of access:

- User access (u). Access granted to the owner of the file.
- Group access (g). Access granted to members of the same group as the owner of the file.
- Other access (o). Access granted to everyone else.

These access rights are defined by the position of read (r), write (w), and execute (x) when the long listing command is issued. For the long listing (**ls -l**) issued earlier, you see the permissions in Table 21-1.

Table 21-1. Long Listing Permissions for the File sort

Access	User Access	Group Access	Other
Read	r	r	-
Write	w	-	-
Execute	x	x	x

Permissions are not granted where a "-" (hyphen) appears. In addition, there are other permissions, such as s, S, t, and T, which I don't cover at this time.

You can see that access rights are arranged in groups of three: three groups of permissions with three access levels each. The owner, in this case *marty*, has read, write, and execute permissions on the file. Anyone in the group *users* is permitted read and execute access to the file. *other* is permitted only execute access of the file.

The definitions of read, write, and execute differ somewhat for files and directories. Here is what you can do if you have read, write, and execute permissions for files:

read	You have permission to read the file.
write	You have permission to change and to write to the file.
execute	You can run, or execute, the program.

Here is what you can do if you have read, write, and execute permissions for directories:

Read	You can list the contents of the directory.
Write	You can create files in the directory, delete files in the directory, and create subdirectories in the directory.
execute	You can change to this directory using the **cd** command, which we'll cover shortly.

We will cover permissions again when the **chmod** command is described.

[Team LiB]

PREVIOUS NEXT

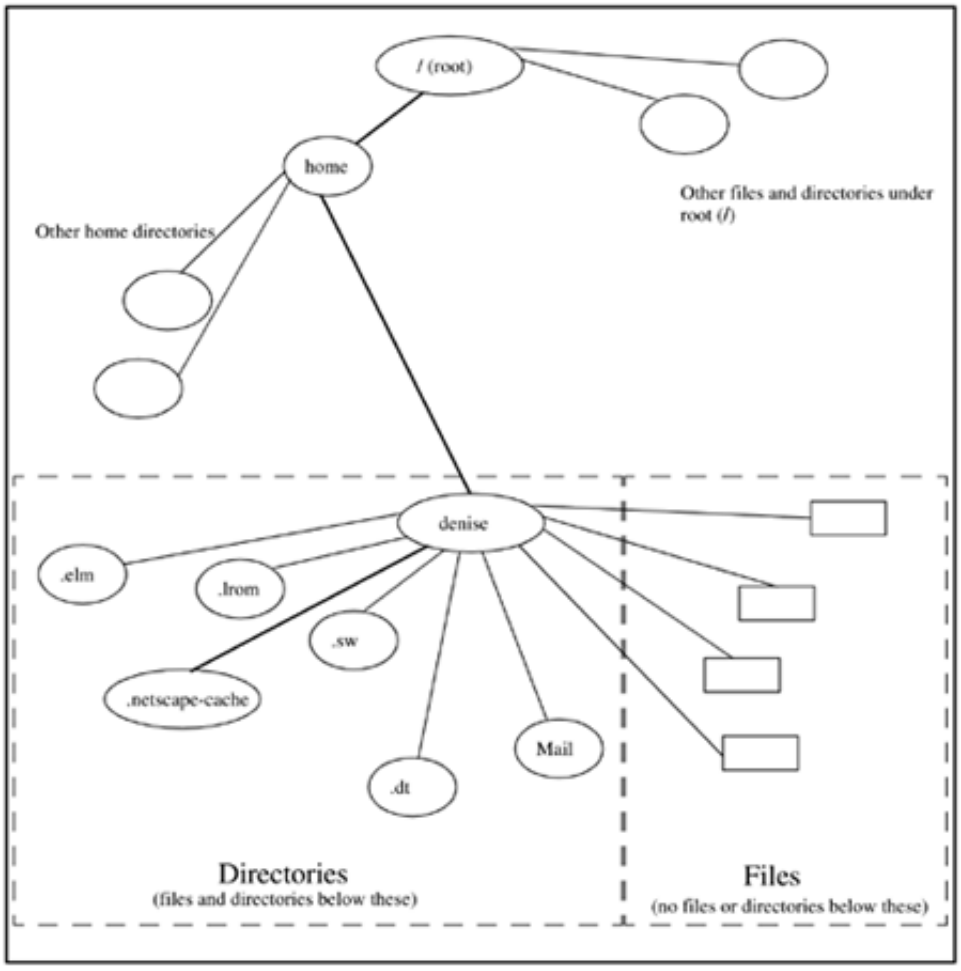
Absolute and Relative Path Names

We have already covered two topics that can serve as the basis for a discussion of absolute and relative path names: some important directories on the system and user login. If you take a look at the user *denise* and the way some of her files can be organized, we can get to the bottom of relative and absolute path names quickly.

The UNIX file system covered in Chapter 2 showed a hierarchy. In this hierarchy, there was the root (**/**) directory at the top, and files and directories were below root. The two means by which you can traverse this hierarchy to get to a "lower" point are with absolute path names and relative path names. Let's take a closer look at the files and directories that *denise* may have in her user area.

First of all we'll assume that *denise* has many files. This activity is one of the things users do - create files. In addition, your system administrator has provided several default files for purposes such as defining your user environment after login (we'll get into this in a lot more detail in upcoming chapters). *denise* probably has many files under her user area, and subdirectories as well. Her user area may look something like that shown in Figure 21-3:

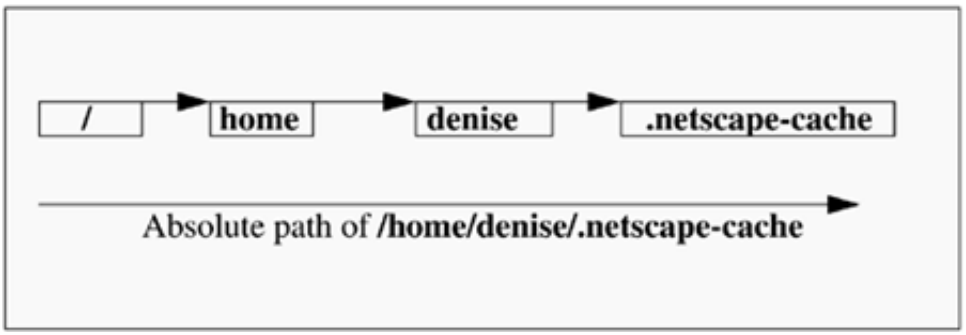
Figure 21-3. Home Directory for *denise*



Most users will have their home directory under **/home** (and who said UNIX doesn't make any sense?). If you want to get to a subdirectory of *denise* using an absolute path name, you traverse the hierarchy using the complete path. To get to the directory **.netscape-cache** under *denise*, you could view the absolute path

as shown in Figure 21-4:

Figure 21-4. Absolute Path



You are progressing from root (**/**), to **home**, to **denise**, and finally to the directory **.netscape-cache**. The change directory (**cd**) command you issue looks like the following:

```
$ cd /home/denise/.netscape-cache
```

This is an absolute path because it starts at root and progresses through the hierarchy. No matter what directory you are currently working in, even if it is **.netscape-cache**, you could use the absolute path name. In many cases, however, you may not need to issue an absolute path name. You may be so close to the file or directory you wish to access that the absolute path would be a waste of time for you. If, for instance, you are already in **/home/denise**, then you could change to **.netscape-cache** using a relative path name easily:

```
$ cd .netscape-cache
```

The relative path name is shorter because you don't begin with a slash (**/**) that brings you back up to the top of the hierarchy to work your way down. Instead, you are starting at some point in the file system hierarchy, such as **/home/denise**, and entering a path relative to that directory, such as **.netscape-cache**.

[Team LiB]

PREVIOUS NEXT



File Name Expansion and Wild Cards

Before we cover additional file system related commands, it is worth taking a look at file name expansion. An overview of file name expansion is useful to ensure that you're comfortable with this topic before we cover additional commands.

Table 21-2 lists some common file name expansion and pattern matching.

Table 21-2. File Name Expansion and Pattern Matching

Character(s)	Example	Description
*	1) **ls *.c**	Match zero or more characters
?	2) **ls conf.?**	Match any single character
[list]	3) **ls conf.[co]**	Match any character in list
[lower-upper]	4) **ls libdd.9873[5-6].sl**	Match any character in range
str{str1,str2,str3,...}	5) **ls ux*.{700,300}**	Expand str with contents of {}
~	6) **ls -a ~**	Home directory
~username	7) **ls -a ~gene**	Home directory of username

The following descriptions of the examples shown in Table 21-2 are more detailed:

1. To list all files in a directory that end in ".c", you could do the following:

   $ **ls *.c**
   conf. SAM.c conf.c

2. To find all the files in a directory named "conf" with an extension of one character, you could do the following:

   $ **ls conf.?**
   conf.c conf.o conf.1

3. To list all the files in a directory named "conf" with only the extension "c" or "o," you could do the following:

   $ **ls conf.{co}**
   conf.c conf.o

4. To list files with similar names but a field that covers a range, you could do the following:

   $ **ls libdd9873[5-6].sl**
   libdd98735.sl libdd98736.sl

5. To list files that start with "ux" and have the extension "300" or "700," you could do the following:

   $ **ls ux*.{700,300}**
   uxbootlf.700 uxinstfs.300

6. To list the files in your home directory, you could use ~ (tilde):

```
$ ls -a ~
    .     .cshrc.org .login    .shrc.org
    ..    .exrc      .login.org .cshrc
          .history   .profile
```

7. To list the files in the home directory of a user, you could do the following:

```
$  ls -a ~gene
   .           .history    splinedat   under.des
   ..          .login      trail.txt   xtra.part
   .chsrc      .login.org  ESP-File
   .cshrc.org  .profile    Mail
   .exrc       .shrc.org   opt
```

[Team LiB]

PREVIOUS NEXT

pwd and cd

When we covered absolute and relative path names, we used the **cd** command to **c**hange **d**irectory. You can be at any point in the file system hierarchy and use **cd** to change to the desired directory, provided that you have the necessary permissions to change to that directory.

We can change a directory using an absolute path name, as shown in the following example:

```
$ cd /home/denise/.netscape-cache
```

Regardless of your current location in the file system hierarchy, this changes you to the directory **/home/denise/.netscape-cache**. If, however, your current location in the file system hierarchy is **/home/denise**, then you could use a relative pathname to change to **.netscape-cache**, as shown in the following example:

```
$ cd .netscape-cache
```

In order to change directory to a place in the file system relative to your current location, you need a way to determine your current location. The **pwd** command, for **p**resent **w**orking **d**irectory, can do this for you. Going back to the previous example in which we changed directories using the relative path, we could have first issued the **pwd** command to see that our location was **/home/denise**, as shown in the following example:

```
$ pwd
/home/denise
$ cd .netscape-cache
$ pwd
$ /home/denise/.netscape-cache
$
```

pwd takes some of the mystery out of determining your current directory.

Let's now take a look at moving up a level in the directory tree using two dots:

```
$ pwd
/home/denise/.netscape-cache
$ cd ..
$ pwd
/home/denise
$
```

The two-dot notation moves you to the parent directory of your current directory.

To return to your home directory, you could issue the **cd** command with no arguments, as shown in the following example:

```
$ pwd
/tmp
$ cd
$ pwd
/home/denise
$
```

This shows that no matter what your current location in the file system hierarchy, you can always get back quickly to your home directory. I don't get into shell parameters for some time, but there is a shell parameter that defines your *home* location, as shown in the following example:

```
$ pwd
/tmp
$ cd $HOME
$ pwd
/home/denise
$
```

Using the **pwd** and **cd** commands, you can always obtain your current directory and change to any directory.

cd - Change to a new current directory.

Arguments

none	Change to home directory. This is defined by the HOME environment variable
..	The two-dot notation moves you to the parent directory of your current directory.
path	You can specify either an absolute or a relative path to change to.

pwd - Present Working Directory, so you know your current location.

Examples

```
$ pwd
/home/denise/.netscape-cache
$ cd ..
$ pwd
/home/denise
$
```

[Team LiB]

PREVIOUS NEXT

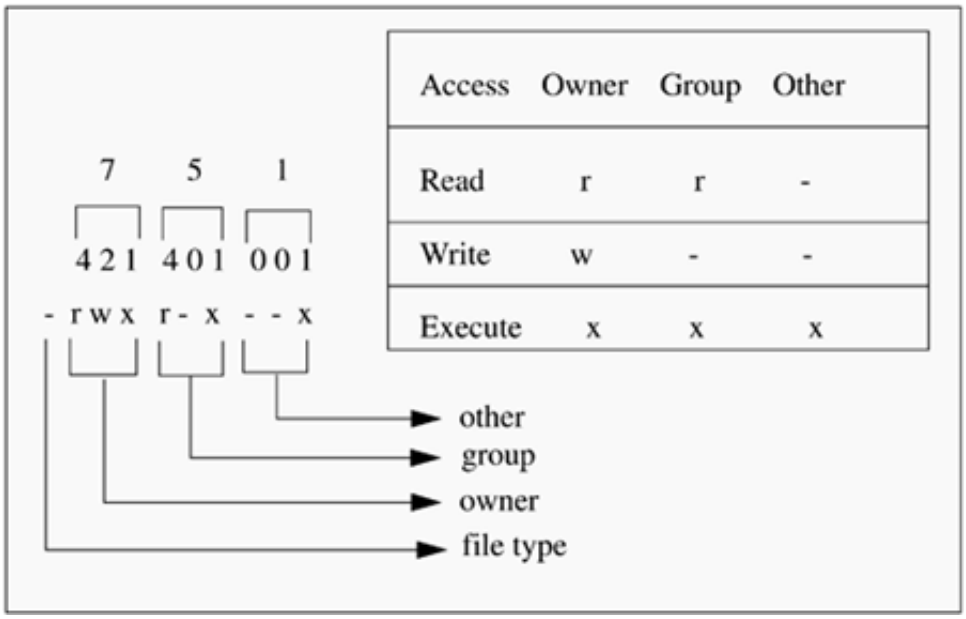
chmod

The **chmod** command is used to change the permissions on a file. Let's start our discussion of **chmod** with the listing of the file **sort** shown earlier:

```
$ ls -l sort
-rwxr-x--x   1 marty     users     120 Jul 26 10:20 sort
```

Figure 21-5 shows a breakdown of the permissions on **sort**.

Figure 21-5. Permissions of File sort

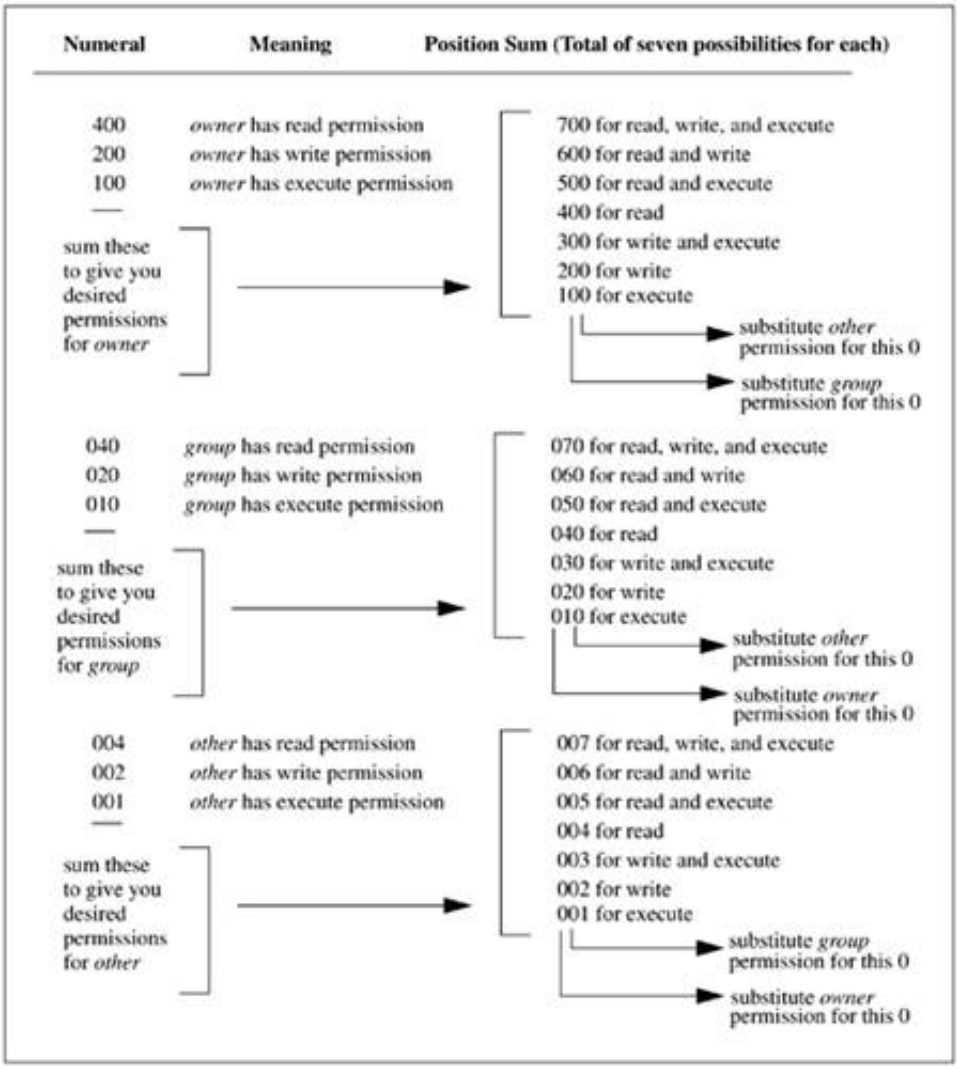


You have very little control over the type of file defined. You do, however, have a great deal of control over the permissions of this file if it belongs to you. The **chmod** command is used to change the permissions on a file or directory. If you are the owner of the file, you can have a field day changing the permissions on the file.

There are two means by which you can change the permissions: symbolic or numeric. I focus first on the numeric mode, because the numbers involved are easy to manage and I sometimes find that new UNIX users get hung up on the meaning of some of the symbols. I'll then cover the symbols and include the symbol meanings in the **chmod** summary. I decided to use the symbols in the summary because the numeric mode, which I much prefer, is becoming obsolete. In some UNIX variants, the **chmod** manual page is strewn with references to "obsolescent form" whenever the numeric mode is covered.

First of all, what do I mean by numbers? Looking at the numbers for **sort**, we see permissions of 751: 7 for *owner* (hundreds position), 5 for *group* (tens position), and 1 for *other* (ones position). Figure 21-6 helps with the meanings of the positions:

Figure 21-6. Numerical Permissions Summary



Selecting the desired permissions for *owner*, *group*, and *other,* you use the **chmod** command to assign those permissions to a file or directory. Some of these permission possibilities are infrequently used, such as execute only, because you usually need to have read access to a file in order to execute it; however, I included all possibilities in Figure 21-6 for completeness. In addition to the permission mode bits shown in figure 21-6, some miscellaneous mode bits also exist that you don't need to be concerned with at this time.

If you decided that you would like to add write permission of the file **sort** for *group* and remove all permissions for *other*, you would simply execute the **chmod** command with the appropriate numeric value. The following set of commands first lists the existing permissions for **sort**, then it changes the permissions on **sort**, and finally it lists the new permissions on **sort**:

```
$ ls -l sort
-rwxr-x--x   1 marty     users     120 Jul 26 10:20 sort

$ chmod 770 sort

$ ls -l sort
-rwxrwx---   1 marty     users     120 Jul 26 10:20 sort
```

The same set of commands to change the permissions using the symbolic mode would be:

```
$ ls -l sort
-rwxr-x--x   1 marty     users     120 Jul 26 10:20 sort

$ chmod g+w,o-x sort

$ ls -l sort
-rwxrwx---   1 marty     users     120 Jul 26 10:20 sort
```

In symbolic mode, you issue the **chmod** command and specify who will be affected by the change [user (u), group (g), other (o), or all (a)], the operation you wish to perform [add (+), delete (-), or replace (=)] on permissions, and the permission you wish to specify [read (r), write (w), or execute (x)]. In the previous example using symbolic mode, write (w) permission is being added (+) for *group* (g), and execute (x) permission is being removed (-) for *other* (o).

The following is a summary of some of the more commonly used symbols of **chmod**:

chmod - Change permissions of specified files using the following symbolic mode list.

Symbol of who is affected:

u	User is affected.
g	Group is affected.
o	Other is affected.
a	All users are affected.

Operation to perform:

+	Add permission.
-	Remove permission.
=	Replace permission.

Permission specified:

r	Read permission.
w	Write permission.
x	Execute permission.
u	Copy user permissions.
g	Copy group permissions.
o	Copy other permissions.



cp

The **cp** command is used to copy a file from one location to another location. You do not alter the original file when you perform a copy. Provided that you have access to the destination directory to which you wish to copy a file, you can place as many copies of a file in that destination as you wish.

The following bullet list describes some of the types of copies you can perform with **cp**. Because there are many file types in UNIX (as covered in Chapter 20) and you have many ways to specify path names for the source and destination files being copied, this list might help you understand the many ways that **cp** can be used:

- Copy a source file to a new file name.
- Copy several source files to a different directory.
- Copy several source files to the same directory.
- Copy an entire directory to a different directory.
- Copy several directories to different directories.

The following is an example of copying a file to a new file name within the same directory:

```
$ cp krsort krsort.sav
```

What if the file **krsort.sav** already exists? The answer is that it is replaced by the new **krsort.sav** being copied. To prevent such mishaps (officially called an overwrite) from occurring, you use the *-i* option to **cp**. *-i* asks you whether you wish to overwrite the file before the copy takes place. If your response is affirmative, then the old file is overwritten with the new file.

The following example first shows a listing of the contents of a directory. Using **cp** with the *-i* option, we copy the file **krsort.c** to **krsortorig.c**, a file that already exists. By responding *n* when asked whether we want to over write **krsortorig.c,** the file is not overwritten and no copy takes place:

```
$ ls -l
total 168
-rwxr-xr-x   1 denise   users      34592 Oct 31 11:27 krsort
-rwxr-xr-x   1 denise   users       3234 Oct 31 11:27 krsort.c
-rwxr-xr-x  1 denise   users      32756 Oct 31 11:27 krsort.dos
-rw-r--r--   1 denise   users       9922 Oct 31 11:27 krsort.q
-rwxr-xr-x  1 denise   users     3085 Oct 31 11:27 krsortorig.c
$ cp -i krsort.c krsortorig.c
overwrite krsortorig.c? (y/n) n
$
```

cp - Copy files and directories.

Options

-i Interactive copy whereby you are prompted to confirm that you wish to overwrite an existing file.

-f Force existing files to be overwritten by files being copied if there is a conflict in file names occurs.

-p Preserve permissions when copying.

-r Copy recursively.

-R Copy recursively, except when permissions are different.





mv

The **mv** command is used to move a file or directory from one location to another location. You can also move multiple files.

The following example shows a listing of a directory, the move of file **krsort.c** to **krsort.test.c** within this directory, and a listing of the directory showing that the file has been moved:

```
$ ls -l
total 168
-rwxr-xr-x   1 denise   users    34592 Oct 31 15:17 krsort
-rwxr-xr-x   1 denise   users     3234 Oct 31 15:17 krsort.c
-rwxr-xr-x   1 denise   users     2756 Oct 31 15:17 krsort.dos
-rw-r--r--   1 denise   users     9922 Oct 31 15:17 krsort.q
-rwxr-xr-x   1 denise   users    3085 Oct 31 15:17 krsortorig.c
$ mv krsort.c krsort.test.c
$ ls -l
total 168
-rwxr-xr-x   1 denise   users  34592 Oct 31 15:17 krsort
-rwxr-xr-x   1 denise   users  32756 Oct 31 15:17 krsort.dos
-rw-r--r--   1 denise   users   9922 Oct 31 15:17 krsort.q
-rwxr-xr-x   1 denise   users   3234 Oct 31 15:17 krsort.test.c
-rwxr-xr-x   1 denise   users   3085 Oct 31 15:17 krsortorig.c
$
```

What if the destination file already exists? You guessed it, UNIX is more than happy to write over the destination file. Using the *-i* option, **mv** asks you to confirm overwriting a file before it does so. The following example shows an attempt to move **krsort.test.c** to **krsortorig.c.** The user is alerted to the fact that **krsortorig.c** already exists and chooses not to let the move take place:

```
$ ls -l
total 168
-rwxr-xr-x   1 denise   users      34592 Oct 31 15:17 krsort
-rwxr-xr-x   1 denise   users      32756 Oct 31 15:17 krsort.dos
-rw-r--r--   1 denise   users       9922 Oct 31 15:17 krsort.q
-rwxr-xr-x   1 denise   users       3234 Oct 31 15:17 krsort.test.c
-rwxr-xr-x   1 denise   users       3085 Oct 31 15:17 krsortorig.c
$ mv -i krsort.test.c krsortorig.c
remove krsortorig.c? (y/n) n
$ ls -l
total 168
-rwxr-xr-x   1 denise   users      34592 Oct 31 15:17 krsort
-rwxr-xr-x   1 denise   users      32756 Oct 31 15:17 krsort.dos
-rw-r--r--   1 denise   users       9922 Oct 31 15:17 krsort.q
-rwxr-xr-x   1 denise   users       3234 Oct 31 15:17 krsort.test.c
-rwxr-xr-x   1 denise   users       3085 Oct 31 15:17 krsortorig.c
$
```

Because the response was not in the affirmative, the move does not take place, and the original **krsortorig.c** remains intact.

mv - Move files and directories.

Options

-i Interactive move whereby you are prompted to confirm that you wish to overwrite an existing file.

-f Force existing files to be overwritten by files being moved if a conflict in file names occurs.

-p Preserve permissions when moving.



mkdir

How nice it would be to have a command to make a directory any time you wish. You could then use this command to create directories and thereby organize your files in multiple directories in a way similar to organizing files in a filing cabinet. The **mkdir** command allows you to do just that - make a directory.

This is an incredibly simple command. You specify the name of the directory to create. In the following example, we'll look at the contents of a directory with the **ls** command, use **mkdir** to make the directory named **default.permissions**, and then perform another **ls** to see the new directory:

```
$ ls -l
total 2
drwxr-xr-x   2 denise   users       1024 Oct 31 11:27 krsort.dir.old
$ mkdir default.permissions
$ ls -l
total 4
drwxr-xr-x   2 denise   users       1024 Oct 31 11:27 krsort.dir.old
drwxr-xr-x   2 denise   users         24 Oct 31 11:32 default.permissions
$
```

The new directory has been produced with default permissions for the user *denise*. *group* and *other* have both read and execute permissions for this directory.

What if you wanted to create a directory with specific permissions on it instead of default permissions? You could use the *-m* option to **mkdir** and specify the mode or permissions you wanted. To give all users read permission on the **krsort.dir.new** directory, you issue the following:

```
$ mkdir -m "a=r" read.permissions
$ ls -l
total 6
drwxr-xr-x   2 denise   users       1024 Oct 31 11:27 krsort.dir.old
drwxr-xr-x   2 denise   users         24 Oct 31 11:32 default.permissions
dr--r--r--   2 denise   users         24 Oct 31 11:33 read.permissions
$
```

Remember the symbolic versus numeric mode of permissions? This **mkdir** command shows the symbolic mode, which although I do not like as much as the numeric mode that should be used because the numeric mode is becoming obsolete.

You don't have to stop at creating a directory with only one level of depth. With the *-p* option, you can create a new directory with any number of subdirectories in it. Intermediate directories are created with the *-p* option. Let's now create a directory named **level1**, with the directory **level2** beneath it, and the directory **level3** below **level2** in the following example. The **ls** command with the *-R* option recursively lists the directories below **level1**:

```
$ mkdir -p level1/level2/level3
$ ls -R level1
level2

level1/level2:
level3

level1/level2/level3:
$
```

After creating the directory **level1** and issuing the **ls** command, we can see that **level2** is indeed beneath **level1**, and **level3** is beneath **level2**.

mkdir - Create specified directories.

Options

- -m — Specify the mode (permissions) of the directory.
- -p — Create intermediate directories to achieve the full path. If you want to create several layers of directories down, you use *-p*.

rm

The **rm** command removes one or more files from a directory and can also be used to remove the directory itself. Going back to our earlier discussion on permissions, you must have both *write* and *execute* permissions on a directory in order to remove a file from it. If you own the directory from which you are going to remove files, then you can probably remove files from it. If, however, you don't have the appropriate permissions on a directory, then the **rm** fails.

As with some of the other commands we have covered, you can use the *-i* option, which asks you to confirm each file as it is removed. This means that if you are asked whether you really wish to remove a file and you respond *n*, then the file is not removed. If you respond *y*, the file is removed.

You can also use the *-r* (or *-R*) option to recursively delete the contents of directories and then delete the directories. This means you can recursively delete the files and directories specified. If there is any question in your mind about whether or not you wish to recursively delete files and directories, then use the *-i* option along with *-r*.

You can use the *-f* option to remove files and directories, which performs removes *without* asking you to confirm them.

The following example performs a long listing of the directory **krsort.dir.new**, interactively prompts the user to see whether he or she wants to delete the files in this directory, and then lists the contents of this directory again, showing that all files have *not* been deleted because the user responded *n*:

```
$ ls -l krsort.dir.new
total 168
-rwxr-xr-x   1 denise   users    34592 Oct 27 18:44 krsort
-rwxr-xr-x   1 denise   users     3234 Oct 27 18:46 krsort.c
-rwxr-xr-x   1 denise   users    32756 Oct 27 18:46 krsort.dos
-rw-r--r--   1 denise   users     9922 Oct 27 18:46 krsort.q
-rwxr-xr-x   1 denise   users     3085 Oct 27 18:46 krsortorig.c
$ rm -i krsort.dir.new/*
../krsort.dir.new/krsort: ? (y/n) n
../krsort.dir.new/krsort.c: ? (y/n) n
../krsort.dir.new/krsort.dos: ? (y/n) n
../krsort.dir.new/krsort.q: ? (y/n) n
../krsort.dir.new/krsortorig.c: ? (y/n) n
$ ls -l krsort.dir.new
total 168
-rwxr-xr-x   1 denise   users    34592 Oct 27 18:44 krsort
-rwxr-xr-x   1 denise   users     3234 Oct 27 18:46 krsort.c
-rwxr-xr-x   1 denise   users    32756 Oct 27 18:46 krsort.dos
-rw-r--r--   1 denise   users     9922 Oct 27 18:46 krsort.q
-rwxr-xr-x   1 denise   users     3085 Oct 27 18:46 krsortorig.c
$
```

Note that the response to being asked whether the file should be deleted was *n* in all cases. This means that none of the files have been removed. A *y* response to any question results in that file being removed from the directory. To interactively delete a directory, you combine the options *-i* and *-r* of the **rm** command. If, however, you do not delete every file in a directory, then the directory is not removed if the *-i* option is used. The first part of the following example shows all but the file **krsort** being removed from the directory **krsort.dir.new**. The directory is not deleted because this file still exists. When the file is removed in the second part of this example, the directory itself is then deleted:

```
$ rm -ir krsort.dir.new
directory krsort.dir.new: ? (y/n) y
krsort.dir.new/krsort: ? (y/n) n
krsort.dir.new/krsort.c: ? (y/n) y
krsort.dir.new/krsort.dos: ? (y/n) y
krsort.dir.new/krsort.q: ? (y/n) y
krsort.dir.new/krsortorig.c: ? (y/n) y
krsort.dir.new: ? (y/n) y
rm: directory krsort.dir.new not removed.  Directory not empty

$ rm -ir krsort.dir.new
directory krsort.dir.new: ? (y/n) y
krsort.dir.new/krsort: ? (y/n) y
krsort.dir.new: ? (y/n) y
```

$

rm - Remove files and directories.

Options

-i	Interactive remove whereby you are prompted to confirm that you wish to remove an existing file.
-f	Force files to be removed.
-r (-R)	Recursively remove the contents of the directory and then the directory itself.

[Team LiB]

rmdir

The **rmdir** command removes one or more directories. The directory must be empty in order to be removed or you must use the *-f* option. You can also specify more than one directory to be removed. Going back to our earlier discussion on permissions, you must have both *write* and *execute* permissions on the parent of a directory to be removed in order to remove it.

As with some of the other commands we have covered, you can use the *-i* option, which asks you to confirm each directory as it is removed. This means that if you are asked whether you really wish to remove a directory and you respond *n,* then it is not removed. If you respond *y,* it is removed.

The order in which you specify that directories are to be removed is significant. If you want to remove both a directory and its subdirectory, you must specify the subdirectory to be removed first. If you specify the parent directory rather than its subdirectory to be removed first, the removal of the parent directory fails because it is not empty.

You can use the *-f* option to force the removal of directories, an action that performs removal *without* asking you to confirm them.

The following example performs a long listing of the directory **krsort.dir.new**, showing that this directory has in it a file called **.dotfile**. When we attempt to remove this directory with **rmdir,** a message is displayed informing us that this directory is not empty. The file .**dotfile** in this directory prevents the **rmdir** command from removing **krsort.dir.new**. After removing **.dotfile,** we are able to remove the directory with **rmdir**:

```
$ ls -al ../krsort.dir.new
total 4
drwxr-xr-x   2 denise   users       1024 Oct 27 18:57 .
drwxrwxr-x   4 denise   users       1024 Oct 27 18:40 ..
-rw-r--r--   1 denise   users          0 Oct 27 18:56 .dotfile
$ rmdir -i ../krsort.dir.new
../krsort.dir.new: ? (y/n) y
rmdir: ../krsort.dir.new: Directory not empty
$ rm ../krsort.dir.new/.dotfile
$ rmdir -i ../krsort.dir.new
../krsort.dir.new: ? (y/n) y
$
```

rmdir has now successfully removed **krsort.dir.new** because **.dotfile** is gone:

rmdir - Remove directories.

Options

- i — Interactive remove whereby you are prompted to confirm that you wish to remove a directory.
- f — Force directories to be removed.
- p — If, after removing a directory, the parent directory is empty, then remove it also. This goes on until a parent directory is encountered that is not empty.

Using Commands

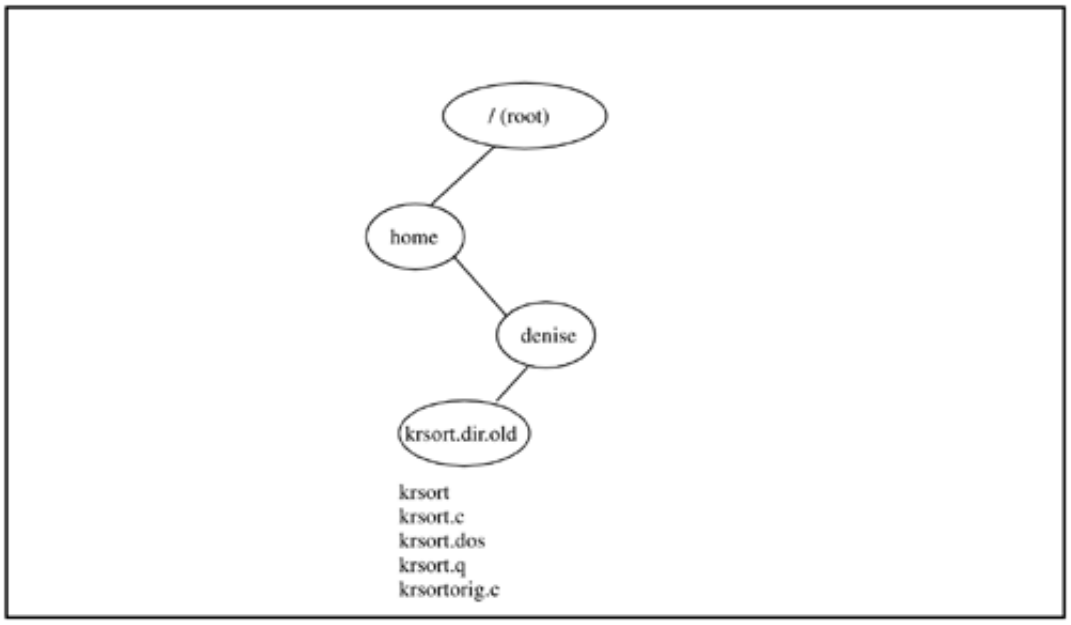
Using the cd, pwd, ls, mkdir, and cp Commands

Now that we have covered some of these commands in an "isolated" fashion, let's put some of the commands together.

Let's start by viewing the hierarchy of a directory under *denise's* home directory, called **krsort.dir.old,** as shown in Figure 21-7:

```
$ cd /home/denise/krsort.dir.old
$ pwd
/home/denise/krsort.dir.old
$ ls -l
total 168
-rwxr-xr-x   1 denise   users    34592 Oct 27 18:20 krsort
-rwxr-xr-x   1 denise   users     3234 Oct 27 17:30 krsort.c
-rwxr-xr-x   1 denise   users    32756 Oct 27 17:30 krsort.dos
-rw-r--r--   1 denise   users     9922 Oct 27 17:30 krsort.q
-rwxr-xr-x  1 denise   users     3085 Oct 27 17:30 krsortorig.c
$
```

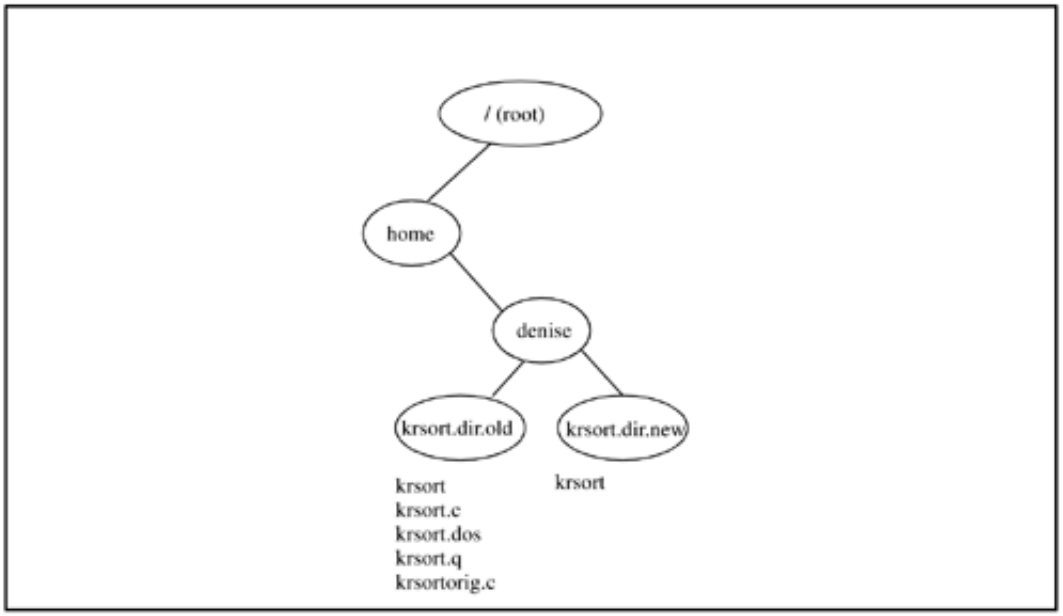
Figure 21-7. /home/denise/krsort.dir.old



We can then make a new directory called **krsort.dir.new** and copy a file to it as shown in Figure 21-8:

```
$ mkdir ../krsort.dir.new
$ cp krsort ../krsort.dir.new
$ ls -l ../krsort.dir.new
total 68
-rwxr-xr-x   1 denise   users      34592 Oct 27 18:27 krsort
$
```

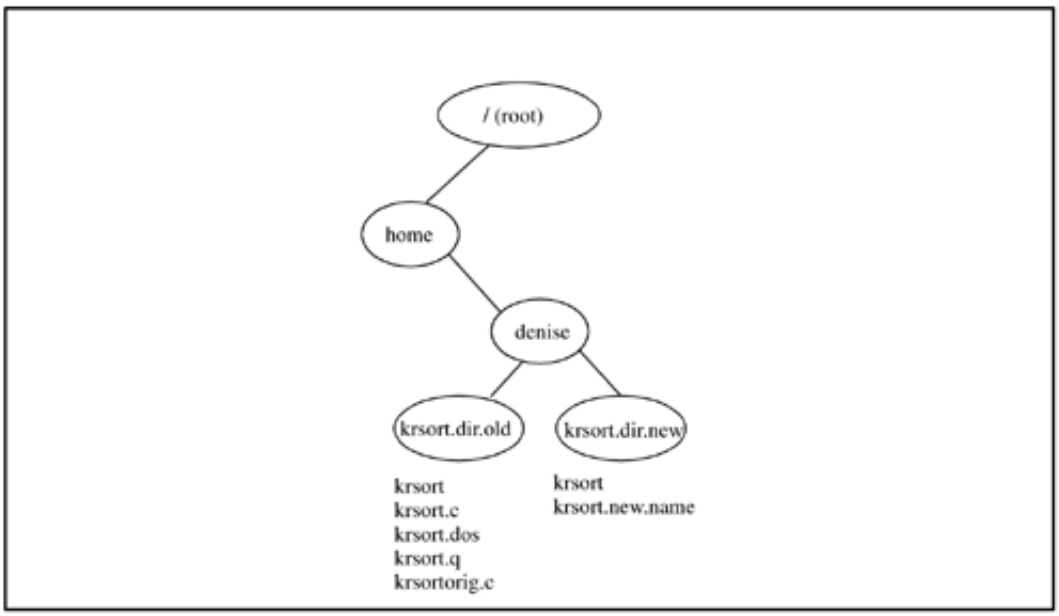
Figure 21-8. /home/denise/krsort.dir.new



Now let's try the *-i* option to **cp**. If we attempt to copy a file to an existing file name, we'll be asked if we wish to overwrite the destination file. We are alerted to the fact that the destination file already exists and we can then select a new name for the file we wish to copy, as shown in Figure 21-9:

```
$ pwd
/users/denise/krsort.dir.old
$ cp -i krsort ../krsort.dir.new
overwrite ../krsort.dir.new/krsort? (y/n) n
$ cp krsort ../krsort.dir.new/krsort.new.name
$ ls -l ../krsort.dir.new
total 136
-rwxr-xr-x   1 denise  users    34592 Oct 27 18:27 krsort
-rwxr-xr-x     1 denise      users       34592  Oct 27 18:29 krsort.new.name
$
```

Figure 21-9. /home/denise/krsort.dir.new/krsort.new.name Added

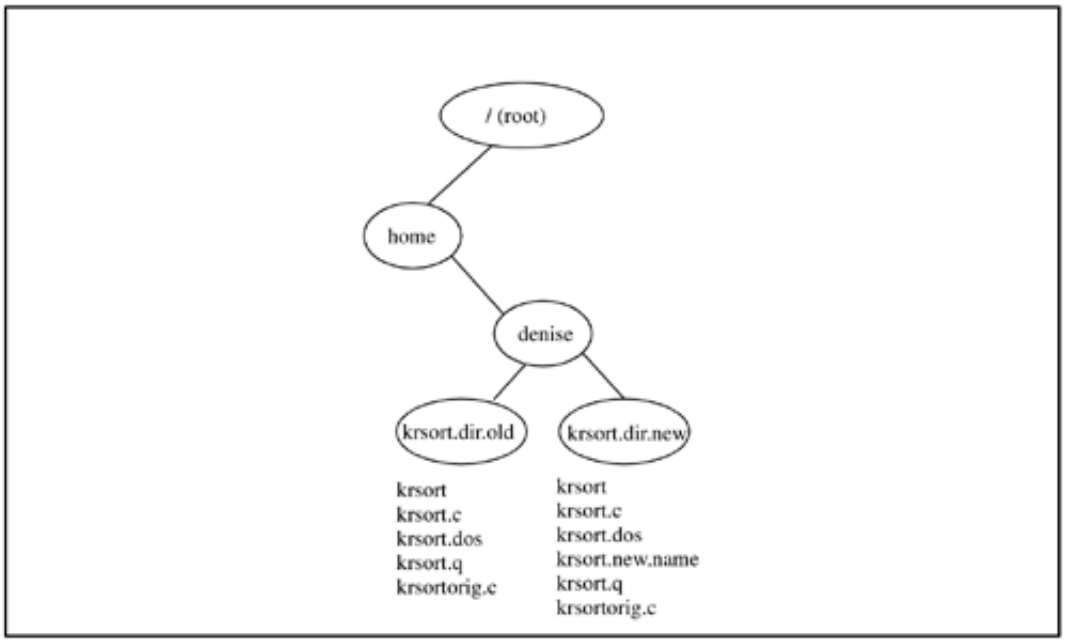


We can also use a wild card with **cp** to copy all files in **krsort.dir.old** to **krsort.dir.new** as shown in Figure

21-10:

```
$ cp * ../krsort.dir.new
$ ls -l ../krsort.dir.new
total 236
-rwxr-xr-x   1 denise   users     34592 Oct 27 18:30 krsort
-rwxr-xr-x   1 denise   users      3234 Oct 27 18:30 krsort.c
-rwxr-xr-x   1 denise   users     32756 Oct 27 18:30 krsort.dos
-rwxr-xr-x   1 denise   users     34592 Oct 27 18:29
krsort.new.name
-rw-r--r--   1 denise   users      9922 Oct 27 18:30 krsort.q
-rwxr-xr-x   1 denise   users      3085 Oct 27 18:30 krsortorig.c
$
```

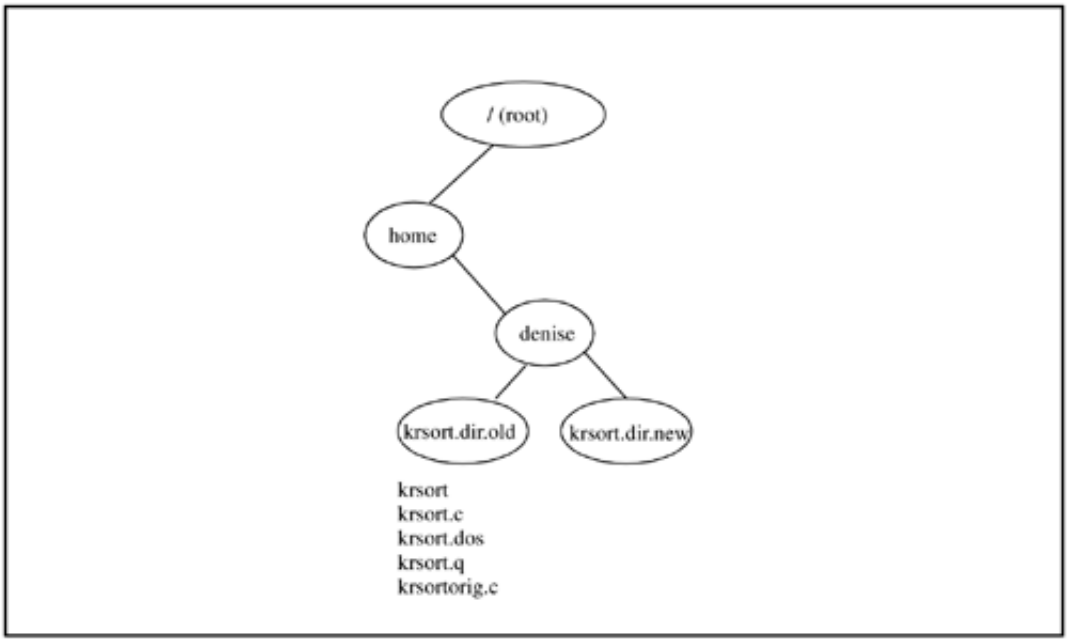
Figure 21-10. All Files Copied to /home/denise/krsort.dir.new



Using the mv Command

Let's start over at the point where the **krsort.dir.new** directory is empty, as shown in Figure 21-11:

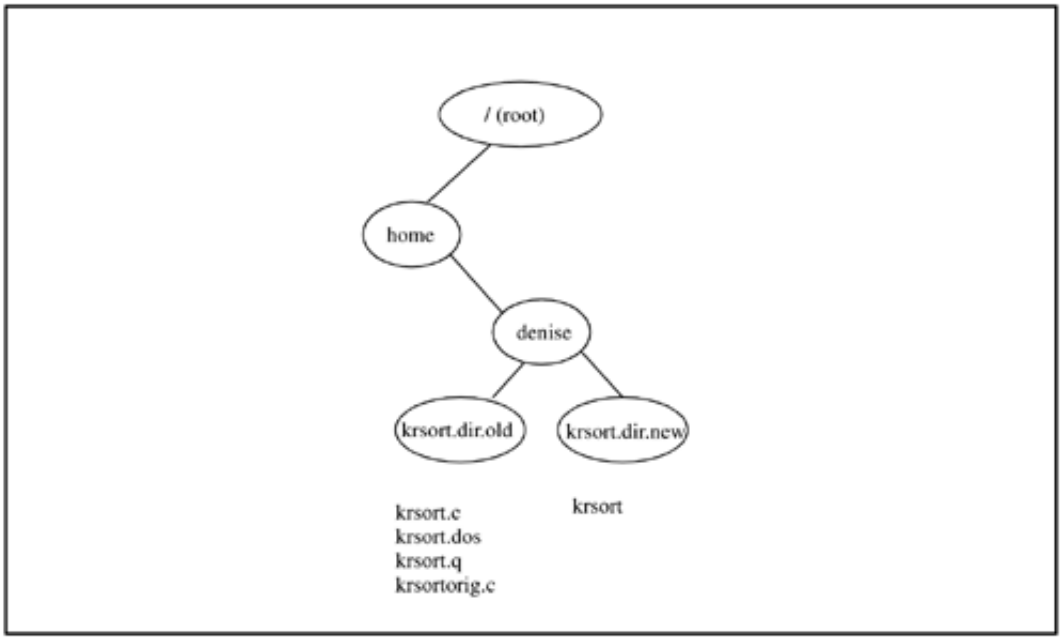
Figure 21-11. Empty /home/denise/krsort.dir.new Directory



We can now move the file **krsort** to the **krsort.dir.new** directory as shown in Figure 21-12:

```
$ mv krsort ../krsort.dir.new
$ ls -l ../krsort.dir.new
total 68
-rwxr-xr-x   1 denise   users      34592 Oct 27 18:20 krsort
$
```

Figure 21-12. krsort Moved to /krsort.dir.new



If we now attempt to move **krsort** to the **krsort.dir.new** directory with the *-i* option and write over the file **krsort**, we get the following:

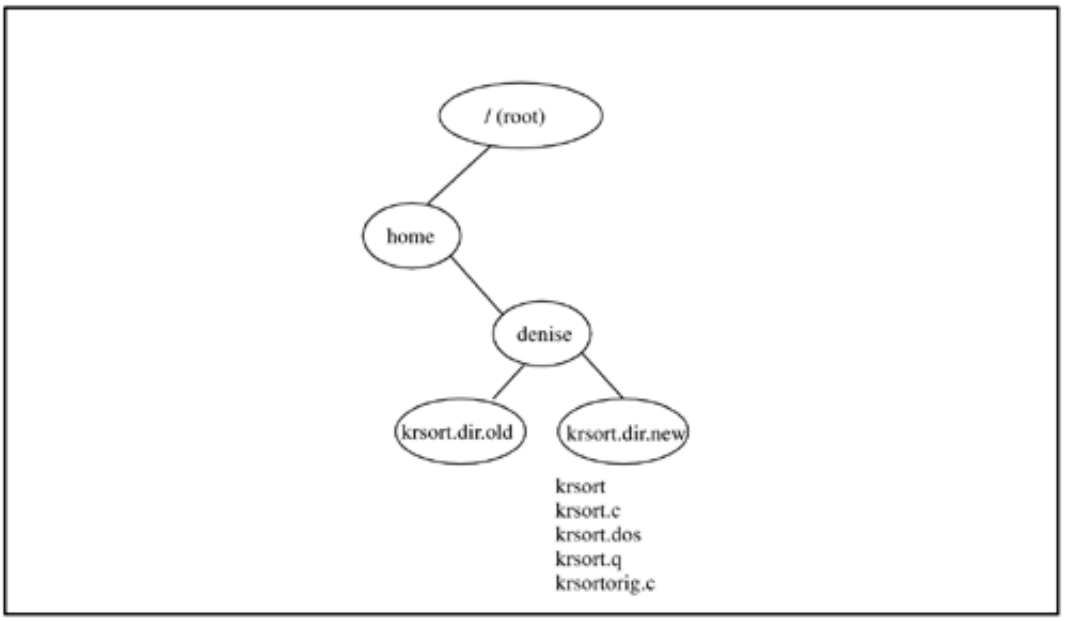
```
$ mv -i krsort ../krsort.dir.new/krsort
remove ../krsort.dir.new/krsort? (y/n) n
$
```

Because we used the *-i* option to **mv**, we are asked whether we wish to allow a file to be overwritten with the move. Because we responded *n* to the question, the file is not overwritten.

We can also use a wild card with the **mv** command to copy all files from the **krsort.dir.old** directory to the **krsort.dir.new** directory. Without the *-i* option, any files in the **krsort.dir.new** directory are overwritten by files that have the same name, as shown in Figure 21-13:

```
$ mv * ../krsort.dir.new
$ ls -l ../krsort.dir.new
total 168
-rwxr-xr-x   1 denise   users    34592 Oct 27 18:44 krsort
-rwxr-xr-x   1 denise   users     3234 Oct 27 18:46 krsort.c
-rwxr-xr-x   1 denise   users    32756 Oct 27 18:46 krsort.dos
-rw-r--r--   1 denise   users     9922 Oct 27 18:46 krsort.q
-rwxr-xr-x   1 denise   users     3085 Oct 27 18:46 krsortorig.c
$
```

Figure 21-13. All Files Moved to /home/denise/krsort.dir.new



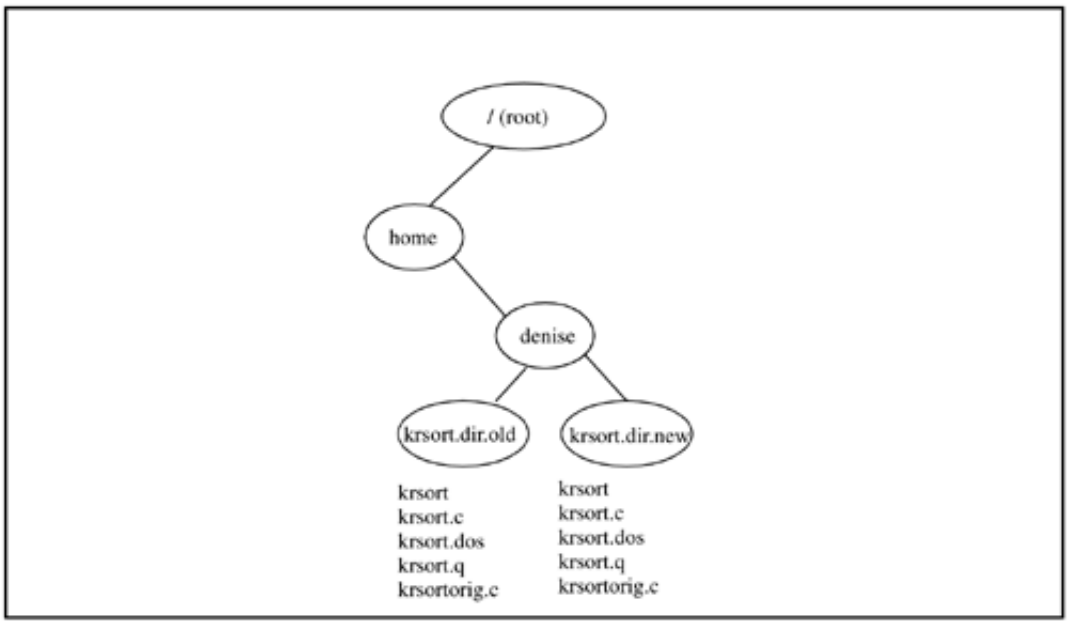
Down and Dirty with the rm and rmdir Commands

The most feared command in the UNIX world, with good reason I might add, is the **rm** command. **rm** removes whatever you want whenever you want, with no questions asked unless you use the *-i* option.

Want to blow away your system instantly? **rm** would be more than happy to help you. As an average user, and not the system administrator, you probably do not have the permissions to do so. It is, however, unnerving to know that this is a possibility. In addition, it is likely that you have permissions remove all of your own files and directories. All of this can be easily avoided by simply using the **-i** option to **rm**.

Let's assume that **krsort.dir.new** and **krsort.dir.old** are identical directories as shown in Figure 21-14:

Figure 21-14. /home/denise/krsort.dir.old and /home/denise/krsort.dir.new Identical



To interactively remove files from **krsort.dir.new**, you do the following:

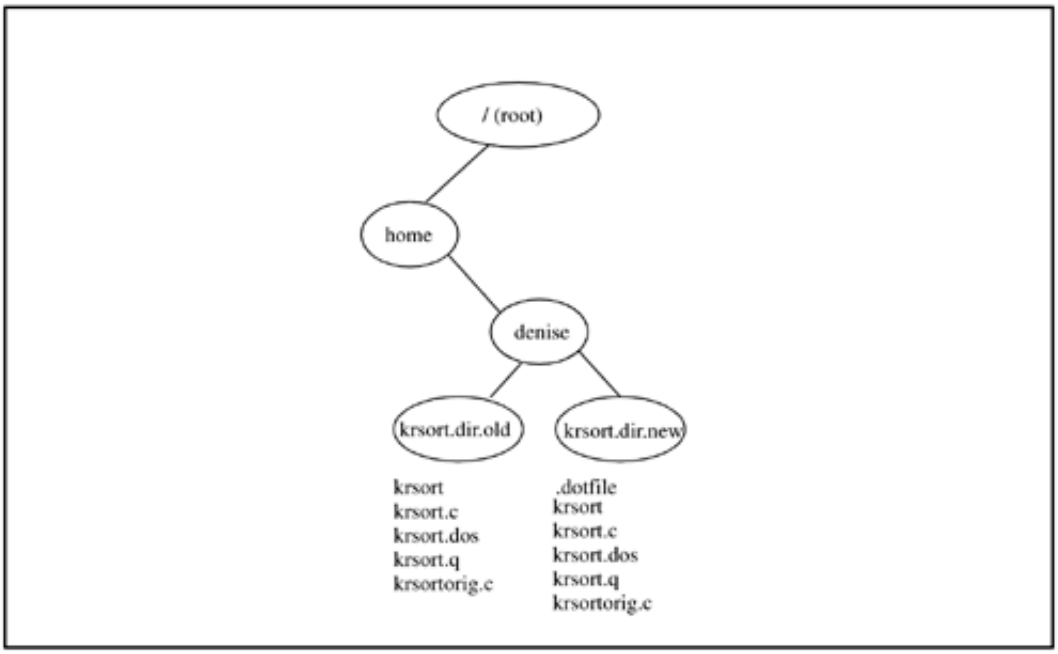
```
$ rm -i ../krsort.dir.new/*
../krsort.dir.new/krsort: ? (y/n) n
../krsort.dir.new/krsort.c: ? (y/n) n
../krsort.dir.new/krsort.dos: ? (y/n) n
../krsort.dir.new/krsort.q: ? (y/n) n
../krsort.dir.new/krsortorig.c: ? (y/n) n
$ ls -l ../krsort.dir.new
total 168
-rwxr-xr-x   1 denise   users    34592 Oct 27 18:44 krsort
-rwxr-xr-x   1 denise   users     3234 Oct 27 18:46 krsort.c
-rwxr-xr-x   1 denise   users    32756 Oct 27 18:46 krsort.dos
-rw-r--r--   1 denise   users     9922 Oct 27 18:46 krsort.q
-rwxr-xr-x   1 denise   users    3085 Oct 27 18:46 krsortorig.c
$
```

This obviously resulted in nothing being removed from **krsort.dir.new** because we responded *n* when asked whether we wanted to delete files.

Let's now go ahead and add a file beginning with a "." (period) to **krsort.dir.new**. The **touch** command does just that, it touches a file to create it with no contents, as shown in Figure 21-15 (**touch** can also be used to update the time stamp of an existing file):

```
$ touch ../krsort.dir.new/.dotfile
$ ls -al ../krsort.dir.new
total 172
drwxr-xr-x   2 denise   users    1024 Oct 27 18:54 .
drwxrwxr-x   4 denise   users    1024 Oct 27 18:40 ..
-rw-r--r--   1 denise   users       0 Oct 27 18:56 .dotfile
-rwxr-xr-x   1 denise   users   34592 Oct 27 18:44 krsort
-rwxr-xr-x   1 denise   users    3234 Oct 27 18:46 krsort.c
-rwxr-xr-x   1 denise   users   32756 Oct 27 18:46 krsort.dos
-rw-r--r--   1 denise   users    9922 Oct 27 18:46 krsort.q
-rwxr-xr-x   1 denise   users    3085 Oct 27 18:46 krsortorig.c
$
```

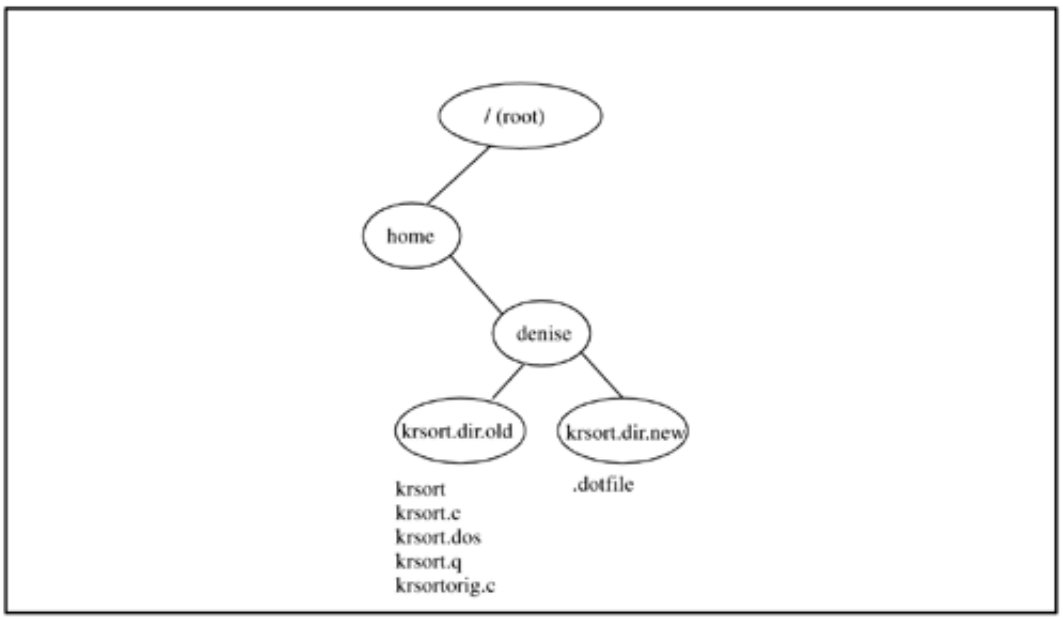
Figure 21-15. /home/denise/krsort.dir.new with .dotfile



If we now attempt to remove files using the same **rm** command earlier issued, we'll see the following, as shown in Figure 21-16:

```
$ rm -i ../krsort.dir.new/*
../krsort.dir.new/krsort: ? (y/n) y
../krsort.dir.new/krsort.c: ? (y/n) y
../krsort.dir.new/krsort.dos: ? (y/n) y
../krsort.dir.new/krsort.q: ? (y/n) y
../krsort.dir.new/krsortorig.c: ? (y/n) y
$ ls -al ../krsort.dir.new
total 4
drwxr-xr-x   2 denise   users       1024 Oct 27 18:57 .
drwxrwxr-x   4 denise   users       1024 Oct 27 18:40 ..
-rw-r--r--   1 denise   users          0 Oct 27 18:56 .dotfile
```

Figure 21-16. Only .dotfile Left in /home/denise/krsort.dir.new

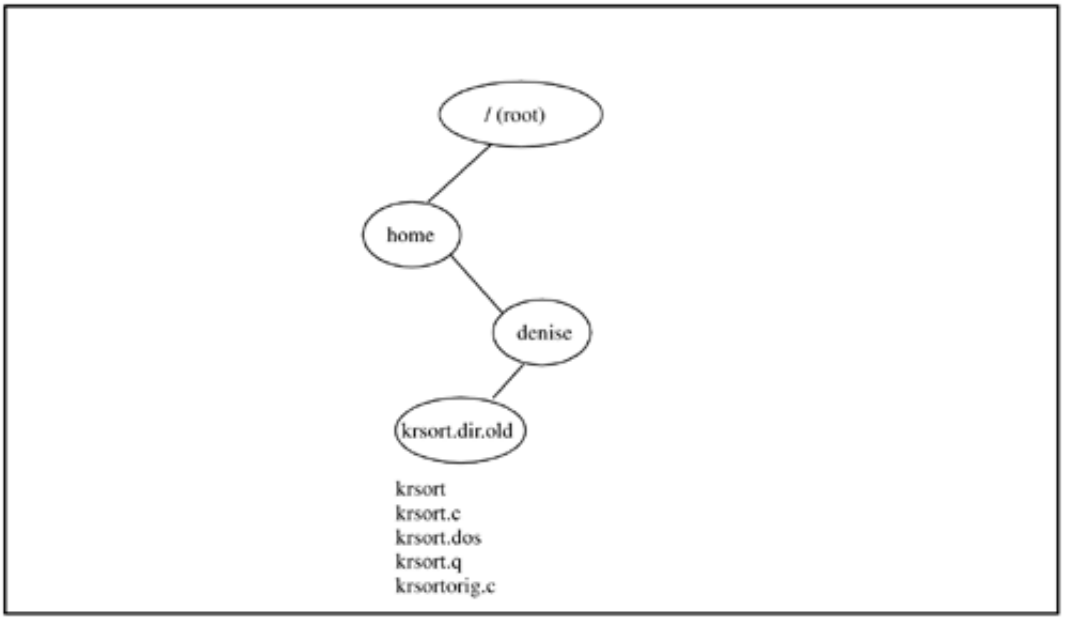


The "*" used as a wild card with the **rm** command does not remove the file **.dotfile**. The file .**dotfile** in this directory prevents the **rmdir** command from removing **krsort.dir.new**. This file must first be removed before the **rmdir** command can successfully delete **krsort.dir.new.**

```
$ rmdir -i ../krsort.dir.new
../krsort.dir.new: ? (y/n) y
rmdir: ../krsort.dir.new: Directory not empty
$ rm ../krsort.dir.new/.dotfile
$ rmdir -i ../krsort.dir.new
../krsort.dir.new: ? (y/n) y
$
```

rmdir has now successfully removed **krsort.dir.new** because **.dotfile** is gone, as shown in Figure 21-17:

Figure 21-17. rmdir Removes /home/denise/krsort.dir.new



Chapter 22. Viewing Files - Redirection, cat, more, pg, head, and tail Commands

First Things First

Redirection

Viewing Files with cat, more, pg, head, and tail

Manual Pages for Commands Used in Chapter 22



First Things First

Before we can get into any serious work with UNIX commands, we'll cover a few different ways to view files in this chapter. Many of the commands covered in this chapter for viewing files will be used in many chapters throughout the book.

Another topic important to many of the UNIX commands to be covered as well as viewing files is redirection. In this chapter, we'll see some commonly used redirection techniques and then some methonds of viewing files. In fact, the very first viewing command that is covered in this chapter uses redirection, so it's important that you understand the basics of redirection before proceeding further. Here's what we'll cover in this chapter:

- Redirection
- **cat, more, pg, head**, and **tail** commands for viewing files



Redirection

Before we cover viewing files, let's talk about redirection for a minute because I use some redirection in the upcoming section. I cover redirection under shell programming, but for now, I want to give you just a quick introduction to redirection so that we can more effectively cover some of the commands in this chapter.

UNIX is set up such that commands usually take their input from the keyboard, often called *standard input*, and usually send their output to the screen, often called *standard output*. Commands also send error information to the screen. It is not always desirable for input to come from standard input and output and errors to go to standard output. You are given a lot of control to override these defaults. This is called redirection. Table 22-1 shows many common forms of redirection.

As shown in the table, to redirect the output of a command from standard output to a file, you would use ">". This works almost all of the time. If you have an environment variable called **noclobber** set, then redirecting to an existing file does not work (we'll cover environment variables shortly). The **noclobber** does not permit redirection to write over an existing file. If you try to write over an existing file, such as **/tmp/processes** below, you receive a message that the file exists:




```
#  ps -ef  > /tmp/processes
/tmp/processes:  File exists
```

You can, however, use a "!" with redirection to force a file to be overwritten. Using ">!" forces a file to be overwritten, and ">>!" will force the output to be appended to the end of an existing file. Examples of these are shown in Table 22-1.

Table 22-1. Commonly Used Redirection Forms

Command or Assignment	Example	Description
<	**wc -l < .login**	Standard input redirection: execute **wc** (word count) and list number of lines (**-l**) in **.login**
>	**ps -ef > /tmp/processes**	Standard output redirection: execute **ps** and send output to file **/tmp/processes**
>>	**ps -ef >> /tmp/processes**	Append standard output: execute **ps** and append output to the end of file **/tmp/processes**
>!	**ps -ef >! /tmp/processes**	Append output redirection and override **noclobber**: write over **/tmp/processes**, even if it exists
>>!	**ps -ef >>! /tmp/processes**	Append standard output and override **noclobber**: append to the end of **/tmp/processes**
| (pipe)	**ps	wc -l**	Run **ps** and use the result as input to **wc**
0 - standard input		
1 - standard		

output		
2 - standard error	**cat program 2> errors**	**cat** the file **program** to standard output and redirect errors to the file **errors**
	cat program 2>> errors	**cat** the file **program** to standard output and append errors to the file **errors**
	find / -name '*.c' -print > cprograms 2>errors	**find** all files on the system ending in **.c** and place the list of files in **cprograms** in the current working directory and send all errors (file descriptor 2) to the file **errors** in current working directory
	find / -name '*.c' -print > cprograms 2>&1	**find** all files on the system ending in **.c**, place the list of files in **cprograms** in the current working directory, and send all errors (file descriptor 2) to same place as file descriptor 1 (**cprograms**)

Using the symbols shown in Table 22-1, you can redirect from *standard input* and *standard output*. For instance, rather than display the out put on the screen, you can send the output to file. We will use some of these redirection forms in upcoming examples.

Viewing Files with cat, more, pg, head, and tail

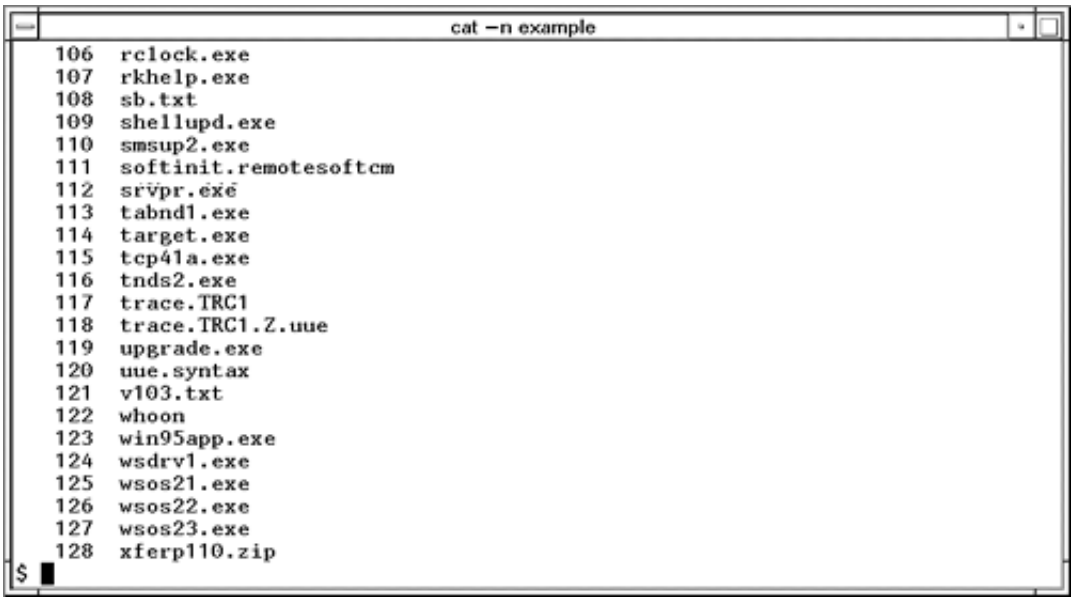
To begin with, let's look at a long file. In fact, let's look at a file so long that it will not fit on your screen if you were to print out the contents of the file to your screen.

The **cat** command (short for *concatenate)* does just this; it prints out the file to your screen. If, however, the file is long, then you will only see the end of the file on your screen. Remember the user *denise* from earlier in the book? She had a lot of files in her directory. Let's list the files in her directory and redirect the output to a file called **listing** with the following command:

```
$ ls -a /home/denise > listing
```

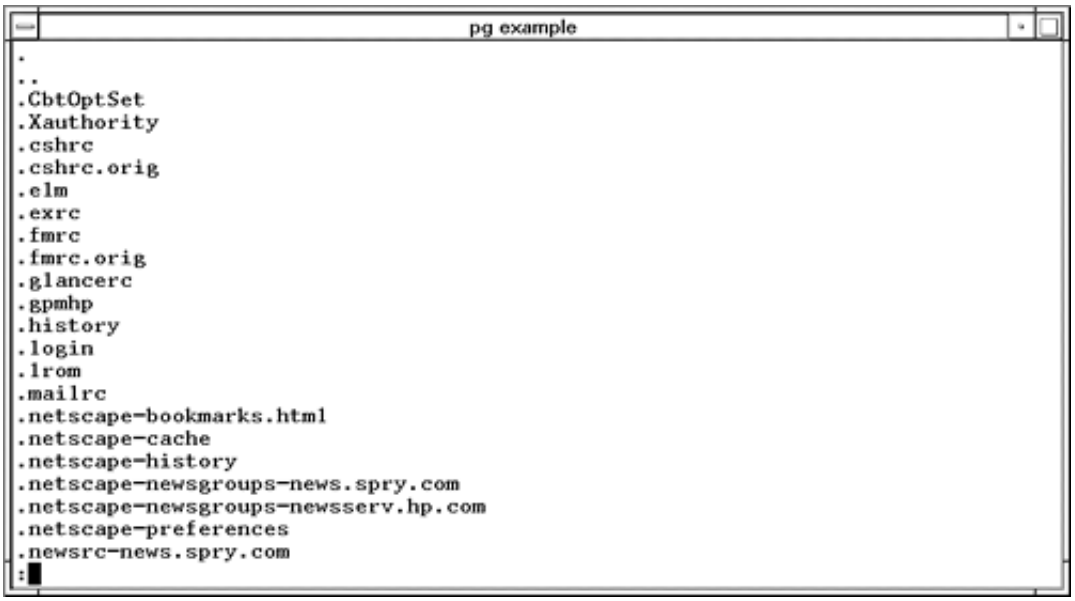
When we **cat listing** to the screen, we see only the end of the file, as shown in Figure 22-1. We know that this is the end of the file because I have issued **cat** with the *-n* option which includes line numbers. The line numbers indicate that this is not the beginning of the file.

Figure 22-1. cat -n Command



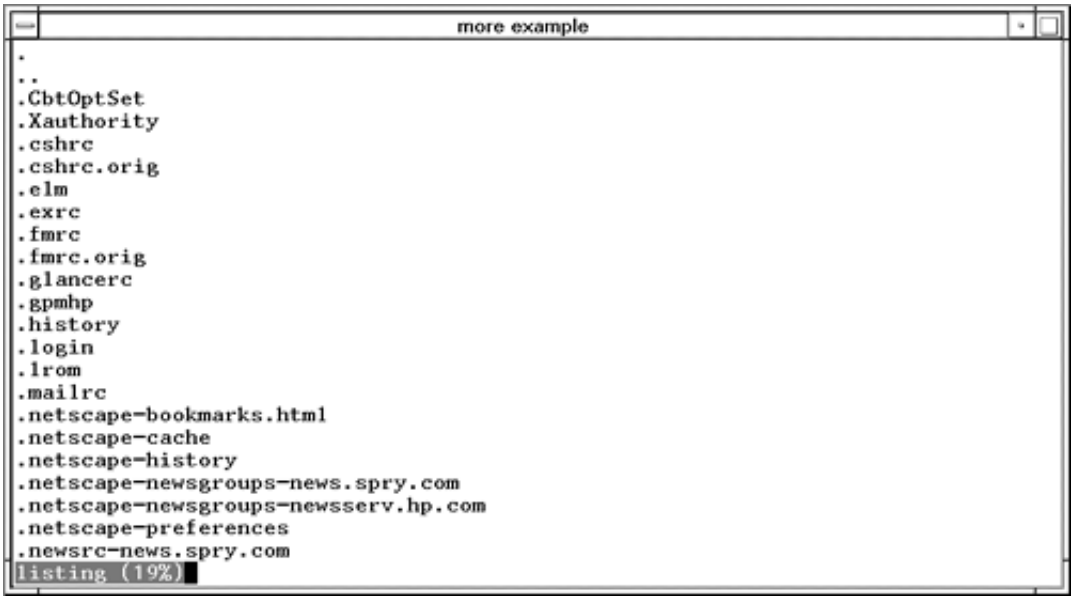
Seeing only the end of this file is not what we had in mind. Using **pg** (for page), we see one screen at time, as shown in Figure 22-2:

Figure 22-2. pg Command



The **more** command produces the same output as **pg,** as shown in Figure 22-3:

Figure 22-3. more Command



This is more like it; now we can scroll on a screen-by-screen basis with both **pg** and **more**. However, sometimes you want to view only the beginning or end of a file. You can use the **head** command to view the beginning of a file and the **tail** command to view the end of a file. The following two examples of **head** and **tail** (see Figures 22-4 and 22-5) show viewing the first 20 lines of **listing** and the last 20 lines of **listing**, respectively:

Figure 22-4. head Command

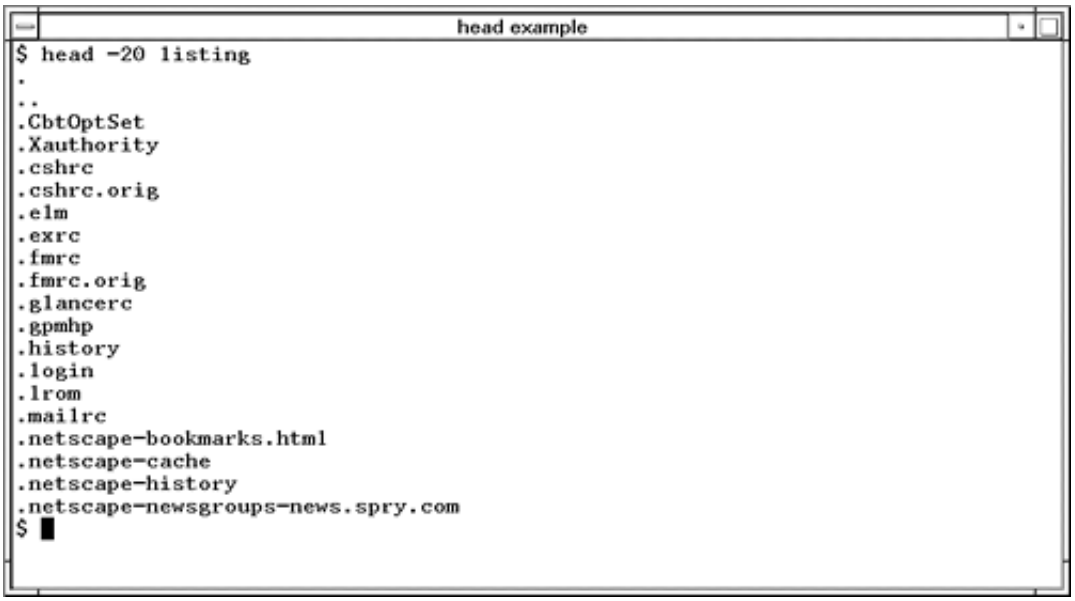
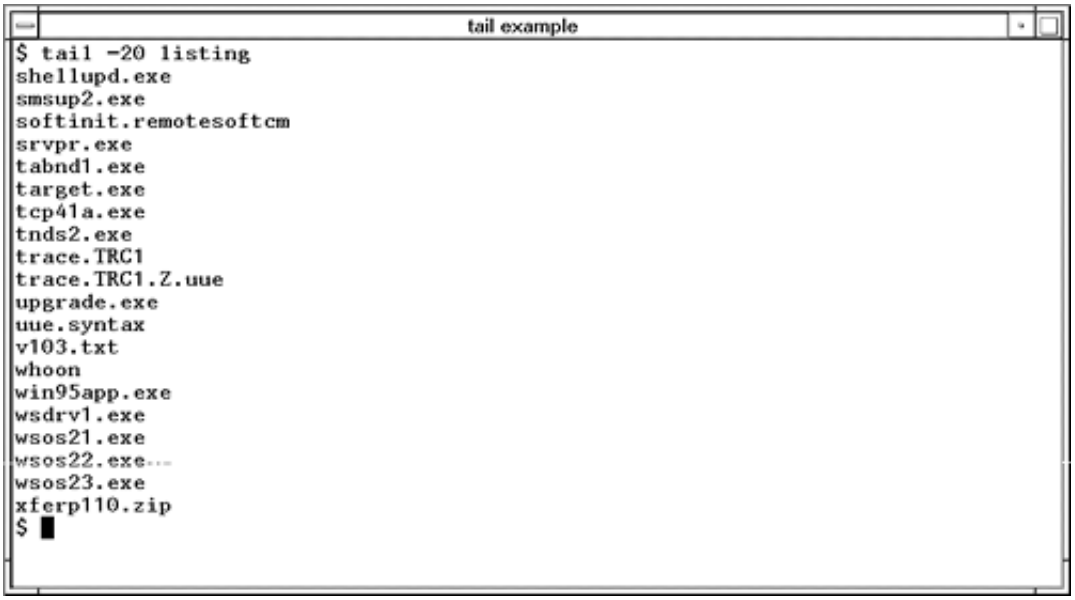


Figure 22-5. tail Command



The command you use depends on the information you wish to display. My personal preference, whether viewing the contents of a large file or a long listing of files, is to use **more**. I don't have a good reason for this, and all I can say is that we are creatures of habit and I have always used **more**. The following are command summaries for **cat, pg, more, head**, and **tail**. I included some of the most frequently used options associated with these commands. Because none of the commands are difficult to use, I suggest that you try each command and see whether one suits your needs better than the others.

Here are summaries of the **cat, pg** (not available on all UNIX variants), **more, head,** and **tail** commands.

cat - Display, combine, append, copy, or create files.

Options

- Used as a substitute for specifying a file name when you want to use the keyboard for standard input.

-n	Line numbers are displayed along with output lines.
-p	Replace multiple consecutive empty lines with only one empty line.
-s	This is silent mode, which suppresses information about nonexistent files.
-u	Output is unbuffered, which means that it is handled character by character.
-v	Print most non-printing characters visibly.

pg - Display all or parts of a file.

Options

-number	The number of lines you wish to display.
-p string	Use *string* to specify a prompt.
-c	Clear the screen before displaying the next page of the file.
-f	Don't split lines being displayed.
-n	A command is issued as soon as a command letter is typed, rather than having to issue a new line character.

more - Display all or parts of a file one screen at a time.

Options

-c	Clear the screen before displaying the next page of the file.
-d	Display a prompt at the bottom of the screen with brief instructions.
-f	Wrap text to fit the screen and judge page length accordingly.
-n	The number of lines in the display window is set to *n*.
-s	Squeeze multiple consecutive empty lines onto one empty line.

head - Provide only the first few lines of a file.

Options

-c	The output is produced with a specified number of bytes.
-l	The output is produced with a specified number of lines. This is the default.
-n count	The number of bytes or lines is specified by *count*. You can also use -count to specify the number of bytes or lines, which is shown in the example. The default count is 10.

tail - Provide the last few lines of a file.

Options

-bnumber	Specify the number of blocks from the end of the file you wish to begin displaying.

-cnumber	Specify the number of characters from the end of the file you wish to begin displaying.
-nnumber	Specify the number of lines from the end of the file you wish to begin displaying. You can also specify a number or minus sign and number, as shown in the example, to specify the number of lines from the end of file to begin displaying.

[Team LiB]

PREVIOUS NEXT

Manual Pages for Commands Used in Chapter 22

There are manual pages in Appendix A for the following command used in this chapter:

ps



Chapter 23. UNIX Tools - split, wc, sort, cmp, diff, comm, dircmp, cut, paste, join, and tr

Not All Commands on All UNIX Variants

Manual Pages for Commands Used in Chapter 23





Not All Commands on All UNIX Variants

A variety of commands are covered in this chapter, including:

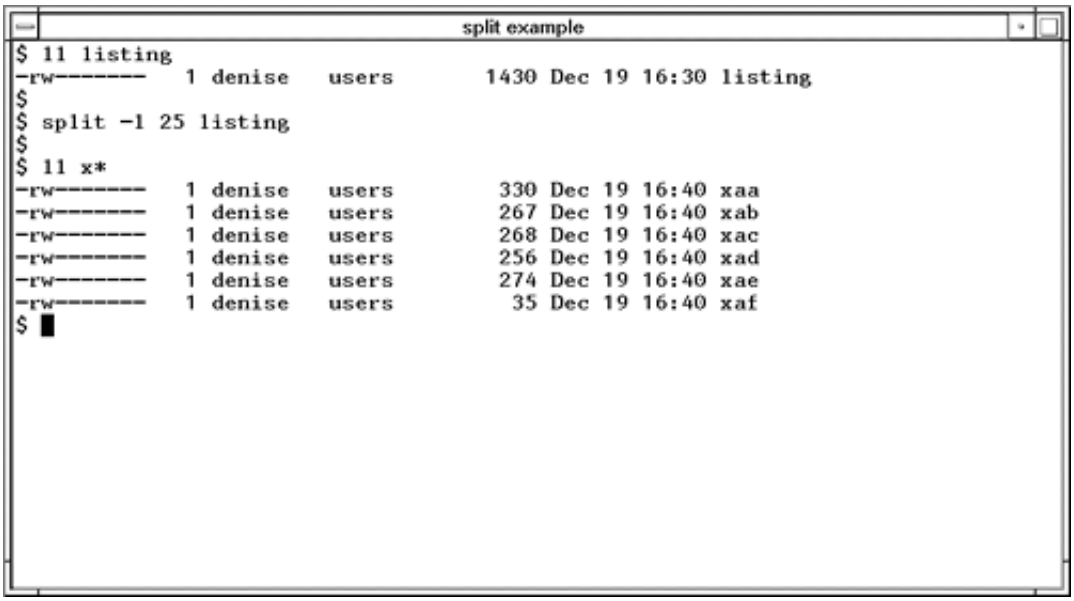
- **split, wc, sort, cmp, diff, comm, dircmp, cut, paste, join**, and **tr** commands

I cover many useful and enjoyable commands in this chapter. All the commands, however, are not available on all UNIX variants. If a specific command is not available on your system, then you probably have a similar command or can combine more than one command to achieve the desired result.

split

Some files are just too long. The file **listing** we earlier looked at may be more easily managed if split into multiple files. We can use the **split** command to make **listing** into files 25 lines long, as shown in Figure 23-1:

Figure 23-1. split Command



Note that the **split** command produced several files from **listing**, called **xaa, xab**, and so on. The -/ option is used to specify the number of lines in files produced by **split**.

Here is a summary of the **split** command:

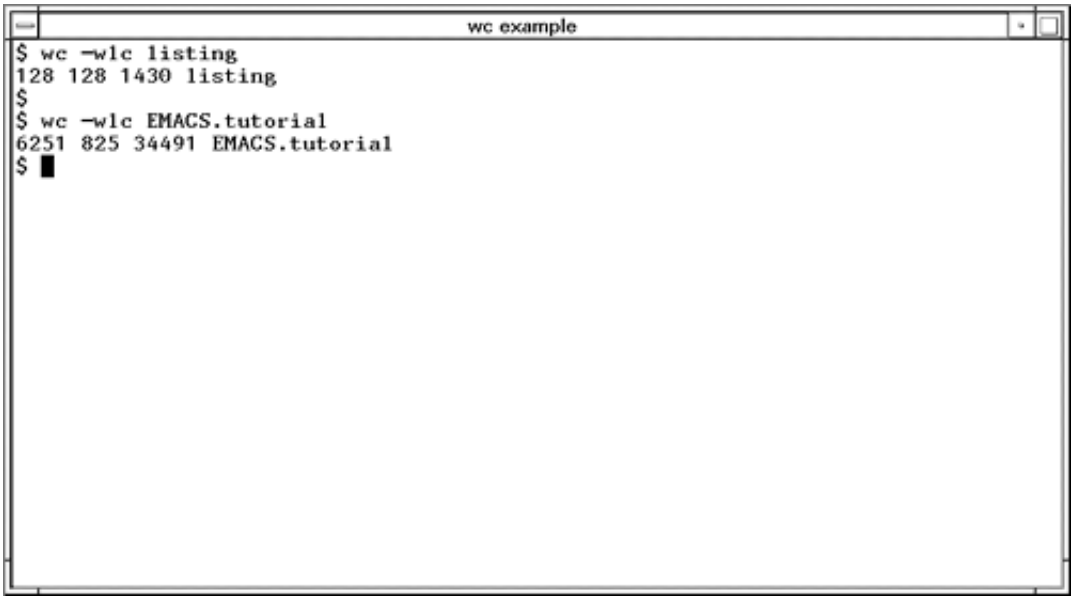
split - Split a file into multiple files.

Options	
-l line_count	Split the file into files with line_count lines per file.
-b n	Split the file into files with n bytes per file.

WC

We know that we have split **listing** into separate files of 25 lines each, but how many lines were in **listing** originally? How about the number of words in **listing**? Those of us who get paid by the word for some of the articles we write often want to know. How about the number of characters in a file? The **wc** command can produce a word, line, and character count for you. Figure 23-2 shows issuing the **wc** command with the *-wlc* options, which produce a count of words with the *-w* option, lines with the *-l* option, and characters with the *-c* option.

Figure 23-2. wc Command



Notice that the number of words and lines produced by **wc** is the same for the file **listing**. The reason is that each line contains exactly one word. When we display the words, lines, and characters with the **wc** command for the text file **EMACS.tutorial**, we can see that the number of words is *6251*, the number of lines is *825*, and the number of characters is *34491*. In a text file, in this case a tutorial, you would expect many more words than lines.

Here is a summary of the **wc** command:

wc - Produce a count of words, lines, and characters.

Options

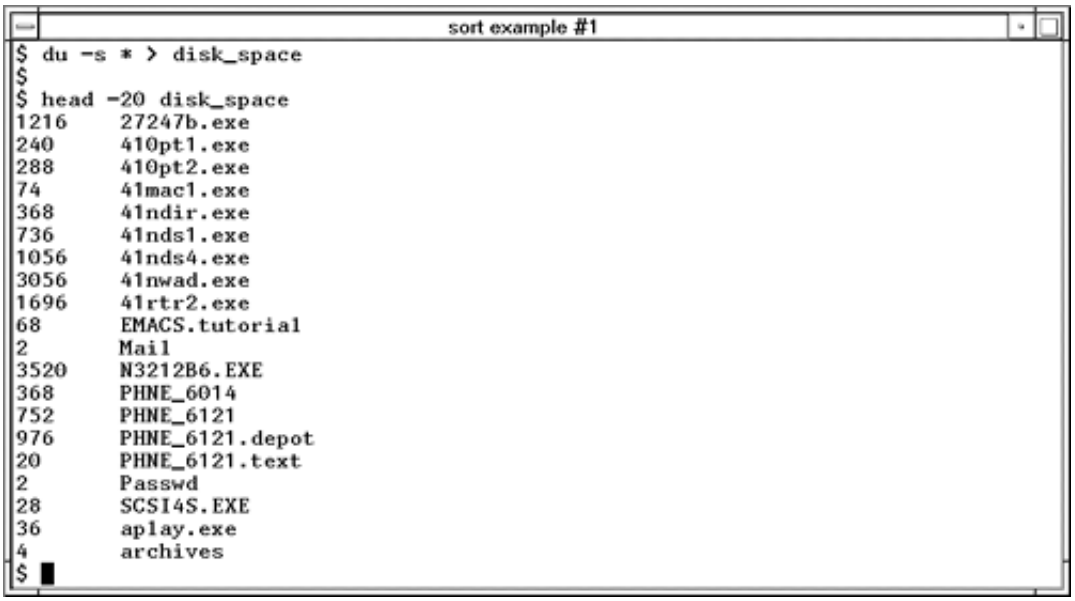
-l	Print the number of lines in a file.
-w	Print the number of words in a file.
-c	Print the number of characters in a file.

sort

Sometimes the contents of files are not sorted in the way you would like. You can use the **sort** command to sort files with a variety of options.

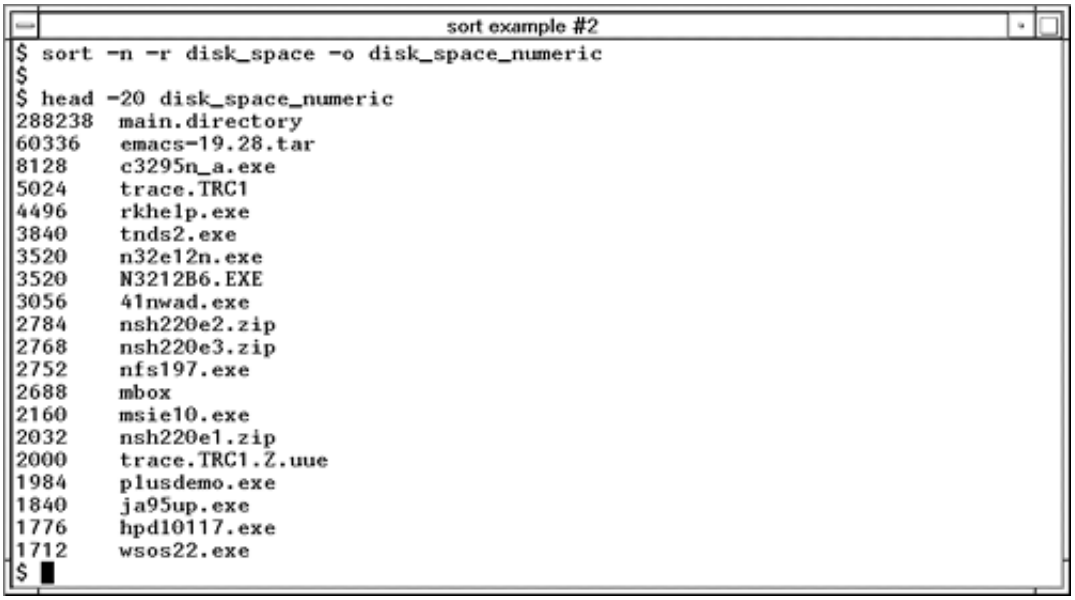
You may find as you use your UNIX system more and more that your system administrator is riding you about the amount of disk space you are consuming. You can monitor the amount of disk space you are consuming with the **du** command. Figure 23-3 shows creating a file called **disk_space** that lists the amount of disk space consumed by files and directories and shows the first 20 lines of the file:

Figure 23-3. sort Command Example #1



Notice that the result is sorted alphabetically. In many cases, this is what you want. If the file were not sorted alphabetically, you could use the **sort** command to do so. In this case, we don't care as much about seeing entries in alphabetical order as we do in numeric order, that is, the files and directories that are consuming the most space. Figure 23-4 shows sorting the file **disk_space** numerically with the *-n* option and reversing the order of the sort with the *-r* option so that the biggest numbers appear first. We then specify the output file name with the *-o* option.

Figure 23-4. sort Command Example #2



What if the items being sorted had many more fields than our two-column disk usage example? Let's go back to the **passwd.test** file for a more complex sort. Let's **cat passwd.test** so we can again see its contents:

```
# cat passwd.test
root:PgYQCkVH65hyQ:0:0:root:/root:/bin/bash
bin:*:1:1:bin:/bin:
daemon:*:2:2:daemon:/sbin:
adm:*:3:4:adm:/var/adm:
```

```
lp:*:4:7:lp:/var/spool/lpd:
sync:*:5:0:sync:/sbin:/bin/sync
shutdown:*:6:11:shutdown:/sbin:/sbin/shutdown
halt:*:7:0:halt:/sbin:/sbin/halt
mail:*:8:12:mail:/var/spool/mail:
news:*:9:13:news:/var/spool/news:
uucp:*:10:14:uucp:/var/spool/uucp:
operator:*:11:0:operator:/root:
games:*:12:100:games:/usr/games:
gopher:*:13:30:gopher:/usr/lib/gopher-data:
ftp:*:14:50:FTP User:/home/ftp:
man:*:15:15:Manuals Owner:/:
nobody:*:65534:65534:Nobody:/:/bin/false
col:Wh0yzfAV2qm2Y:100:100:Caldera OpenLinux
        User:/home/col:/bin/bash
```

Now let's use **sort** to determine which users are in the same group. Fields are separated in **passwd.test** by a colon (:). The fourth field is the group to which a user belongs. For instance, *bin* is in group *1, daemon* in group *2*, and so on. To sort by group, we would have to specify three options to the **sort** command. The first is to specify the delimiter (or field separator) of colon (:) using the *-t* option. Next we would have to specify the field on which we wish to sort with the *-k* option. Finally, we want a numeric sort, so use the *-n* option. The following example shows a numeric sort of the **passwd.test** file by the fourth field:

```
# sort -t: -k4 -n passwd.test
halt:*:7:0:halt:/sbin:/sbin/halt
operator:*:11:0:operator:/root:
root:PgYQCkVH65hyQ:0:0:root:/root:/bin/bash
sync:*:5:0:sync:/sbin:/bin/sync
bin:*:1:1:bin:/bin:
daemon:*:2:2:daemon:/sbin:
adm:*:3:4:adm:/var/adm:
lp:*:4:7:lp:/var/spool/lpd:
shutdown:*:6:11:shutdown:/sbin:/sbin/shutdown
mail:*:8:12:mail:/var/spool/mail:
news:*:9:13:news:/var/spool/news:
uucp:*:10:14:uucp:/var/spool/uucp:
man:*:15:15:Manuals Owner:/:
gopher:*:13:30:gopher:/usr/lib/gopher-data:
ftp:*:14:50:FTP User:/home/ftp:
col:Wh0yzfAV2qm2Y:100:100:Caldera OpenLinux
                    User:/home/col:/bin/bash
games:*:12:100:games:/usr/games:
nobody:*:65534:65534:Nobody:/:/bin/false
```

The following is a summary of the **sort** command.

sort - Sort lines of files (alphabetically by default).

Options

-b Ignore leading spaces and tabs.

-c Check whether files are already sorted, and if so, do nothing.

-d Ignore punctuation and sort in dictionary order.

-f Ignore the case of entries when sorting.

-i Ignore non-ASCII characters when sorting.

-k*s* Use field *s* as the field on which to base the sort.

-m Merge sorted files.

-n Sort in numeric order.

-o file Specify the output file name rather than write to standard output.

-r Reverse the order of the sort by starting with the last letter of the alphabet or with the largest number, as we did in the example.

+n Skip n fields or columns before sorting.

cmp, diff, and comm

A fact of life is that as you go about editing files, you may occasionally lose track of what changes you have made to which files. You may then need to make comparisons of files. Let's take a look at three such commands, **cmp, diff**, and **comm**, and see how they compare files.

Let's assume that we have modified a script called **llsum**. The unmodified version of **llsum** was saved as **llsum.orig**. Using the **head** command, we can view the first 20 lines of **llsum** and then the first 20 lines of **llsum.orig**:

```
# head -20 llsum
#
#!/bin/sh
# Displays a truncated long listing (ll) and
# displays size statistics
# of the files in the listing.

ll $* | \
awk ' BEGIN { x=i=0; printf "%-25s%-10s%8s%8s\n",\
                    "FILENAME","OWNER","SIZE","TYPE" }
      $1 ~ /^[-dlps]/  {# line format for normal files
              printf "%-25s%-10s%8d",$9,$3,$5
              x = x + $5
                 i++
                    }
      $1 ~ /^-/ { printf "%8s\n","file" }  # standard file types
      $1 ~ /^d/ { printf "%8s\n","dir" }
      $1 ~ /^l/ { printf "%8s\n","link" }
      $1 ~ /^p/ { printf "%8s\n","pipe" }
      $1 ~ /^s/ { printf "%8s\n","socket" }
      $1 ~ /^[bc]/ { # line format for device files
                          printf              "%-25s%-10s%8s%8s\n",$10,$3,"","dev"
                 }
#
# head -20 llsum.orig
#
#!/bin/sh
# Displays a truncated long listing (ll) and
# displays size statistics
# of the files in the listing.

ll $* | \
awk ' BEGIN { x=i=0; printf "%-16s%-10s%8s%8s\n",\
                    "FILENAME","OWNER","SIZE","TYPE" }
      $1 ~ /^[-dlps]/  {# line format for normal files
              printf "%-16s%-10s%8d",$9,$3,$5
              x = x + $5
                 i++
                    }
     $1 ~ /^-/ { printf "%8s\n","file" }  # standard file types
      $1 ~ /^d/ { printf "%8s\n","dir" }
      $1 ~ /^l/ { printf "%8s\n","link" }
      $1 ~ /^p/ { printf "%8s\n","pipe" }
      $1 ~ /^s/ { printf "%8s\n","socket" }
      $1 ~ /^[bc]/ { # line format for device files
                          printf              "%-16s%-10s%8s%8s\n",$10,$3,"","dev"
                 }
```

I'm not sure what changes I made to **llsum.orig** to improve it, so we can first use **cmp** to see whether indeed differences exist between the files.

```
$
$ cmp llsum llsum.orig
llsum llsum.orig differ: char 154, line 6
$
```

cmp does not report back much information, only that character *154* in the file at line *6* is different in the two files. There may indeed be other differences, but this is all we know about so far.

To get information about all of the differences in the two files, we could use the *-l* option to **cmp**:

```
$ cmp -l llsum llsum.orig
   154  62  61
   155  65  66
   306  62  61
   307  65  66
   675  62  61
   676  65  66
```

This is not all that useful an output, however. W want to see not only the position of the differences, but also the differences themselves.

Now we can use **diff** to describe all the differences in the two files:

```
$ diff llsum llsum.orig
6c6
< awk ' BEGIN { x=i=0; printf "%-25s%-10s%8s%8s\n",\
---
> awk ' BEGIN { x=i=0; printf "%-16s%-10s%8s%8s\n",\
9c9
<             printf "%-25s%-10s%8d",$9,$3,$5
---
>             printf "%-16s%-10s%8d",$9,$3,$5
19c19
<             printf "%-25s%-10s%8s%8s\n",$10,$3,"","dev"
---
>             printf "%-16s%-10s%8s%8s\n",$10,$3,"","dev"
$
```

We now know that lines *6, 9*, and *25* are different in the two files and these lines are also listed for us. From this listing, we can see that the number *16* in **llsum.orig** was changed to *25* in the newer **llsum** file, and this accounts for all of the differences in the two files. The "less than" sign (<) precedes lines from the first file, in this case **llsum**. The "greater than" sign (>) precedes lines from the second file, in this case **llsum.orig**. I made this change, starting the second group of information from character *16* to character *25*, because I wanted the second group of information, produced by **llsum,** to start at column *25*. The second group of information is the *owner,* as shown in the following example:

```
$ llsum
FILENAME                OWNER         SIZE    TYPE

README                  denise         810    file
backup_files            denise        3408    file
biography               denise         427    file
cshtest                 denise        1024     dir
gkill                   denise        1855    file
gkill.out               denise         191    file
hostck                  denise         924    file
ifstat                  denise        1422    file
ifstat.int              denise        2147    file
ifstat.out              denise         723    file
introdos                denise       54018    file
introux                 denise       52476    file
letter                  denise       23552    file
letter.auto             denise       69632    file
letter.auto.recover     denise       71680    file
letter.backup           denise       23552    file
letter.lck              denise          57    file
letter.recover          denise       69632    file
llsum                   denise        1267    file
llsum.orig              denise        1267    file
llsum.out               denise        1657    file
llsum.tomd.out          denise        1356    file
psg                     denise         670    file
psg.int                 denise         802    file
psg.out                 denise         122    file
sam_adduser             denise        1010    file
tdolan                  denise        1024     dir
trash                   denise        4554    file
trash.out               denise         329    file
typescript              denise        2017    file
```

```
The files listed occupy 393605 bytes (0.3754 Mbytes)
Average file size is 13120 bytes

$
```

When we run **llsum.orig**, clearly the second group of information, which is the *owner*, starts at column 16 and not column 32:

```
$ llsum.orig
FILENAME        OWNER         SIZE    TYPE

README          denise         810    file
backup_files    denise        3408    file
biography       denise         427    file
cshtest         denise        1024     dir
gkill           denise        1855    file
gkill.out       denise         191    file
hostck          denise         924    file
ifstat          denise        1422    file
ifstat.int      denise        2147    file
ifstat.out      denise         723    file
introdos        denise       54018    file
introux         denise       52476    file
letter          denise       23552    file
letter.auto     denise       69632    file
letter.auto.rec denise       71680    file
letter.backup   denise       23552    file
letter.lck      denise          57    file
letter.recover  denise       69632    file
llsum           denise        1267    file
llsum.orig      denise        1267    file
llsum.out       denise        1657    file
llsum.tomd.out  denise        1356    file
psg             denise         670    file
psg.int         denise         802    file
psg.out         denise         122    file
sam_adduser     denise        1010    file
tdolan          denise        1024     dir
trash           denise        4554    file
trash.out       denise         329    file
typescript      denise        3894    file

The files listed occupy 395482 bytes (0.3772 Mbytes)
Average file size is 13182 bytes


script done on Mon Dec 11 12:59:18

$
```

We can compare two sorted files using **comm** and see the lines that are unique to each file, as well as the lines found in both files. When we compare two files with **comm**, the lines that are unique to the first file appear in the first column, the lines unique to the second file appear in the second column, and the lines contained in both files appear in the third column. Let's go back to the **/etc/passwd** file to illustrate this comparison. We'll compare two **/etc/passwd** files, the active **/etc/passwd** file in use and an old **/etc/passwd** file from a backup:

```
# comm /etc/passwd /etc/passwd.backup
                root:PgYQCkVH65hyQ:0:0:root:/root:/bin/bash
                bin:*:1:1:bin:/bin:
                daemon:*:2:2:daemon:/sbin:
                adm:*:3:4:adm:/var/adm:
                lp:*:4:7:lp:/var/spool/lpd:
                sync:*:5:0:sync:/sbin:/bin/sync
                    shutdown:*:6:11:shutdown:/sbin:/sbin/shutdown
                halt:*:7:0:halt:/sbin:/sbin/halt
                mail:*:8:12:mail:/var/spool/mail:
                news:*:9:13:news:/var/spool/news:
                uucp:*:10:14:uucp:/var/spool/uucp:
        operator1:*:12:0:operator:/root:
                operator:*:11:0:operator:/root:
games:*:12:100:games:/usr/games:
                gopher:*:13:30:gopher:/usr/lib/gopher-data:
```

```
ftp:*:14:50:FTP User:/home/ftp:
man:*:15:15:Manuals Owner:/:
nobody:*:65534:65534:Nobody:/:/bin/false
col:Wh0yzfAV2qm2Y:100:100:Caldera
OpenLinux User:/home/col:/bin/bash
```

You can see from this output that the user *games* appears only in the active **/etc/passwd** file, the user *operator1* appears only in the **/etc/passwd.backup** file, and all of the other entries appear in both files.

The following is a summary of the **cmp** and **diff** commands.

cmp - Compare the contents of two files. The byte position and line number of the first difference between the two files is returned.

Options

-l Display the byte position and differing characters for all differences within a file.

-s Work silently; that is, only exit codes are returned.

diff - Compares two files and reports differing lines.

Options

-b Ignore blanks at the end of a line.

-i Ignore case differences.

-t Expand tabs in output to spaces.

-w Ignore spaces and tabs.

dircmp

Why stop at comparing files? You will probably have many directories in your user area as well. **dircmp** compares two directories and produces information about the contents of directories.

To begin with let's perform a long listing of two directories:

```
$ ls -l krsort.dir.old
total 168
-rwxr-xr-x   1 denise   users      34592 Oct 31 11:27 krsort
-rwxr-xr-x   1 denise   users       3234 Oct 31 11:27 krsort.c
-rwxr-xr-x   1 denise   users      32756 Oct 31 11:27 krsort.dos
-rw-r--r--   1 denise   users       9922 Oct 31 11:27 krsort.q
-rwxr-xr-x   1 denise   users       3085 Oct 31 11:27 krsortorig.c
$
$ ls -l krsort.dir.new
total 168
-rwxr-xr-x   1 denise   users      34592 Oct 31 15:17 krsort
-rwxr-xr-x   1 denise   users      32756 Oct 31 15:17 krsort.dos
-rw-r--r--   1 denise   users       9922 Oct 31 15:17 krsort.q
-rwxr-xr-x   1 denise   users       3234 Oct 31 15:17 krsort.test.c
-rwxr-xr-x   1 denise   users       3085 Oct 31 15:17 krsortorig.c
$
```

From this listing, you can see clearly that one file is unique to each directory. **krsort.c** appears in only the **krsort.dir.old** directory, and **krsort.test.c** appears in only the **krsort.dir.new** directory. Let's now use **dircmp** to inform us of the differences in these two directories:

```
$ dircmp krsort.dir.old krsort.dir.new


krsort.dir.old only and krsort.dir.new only Page 1
```

```
./krsort.c        ./krsort.test.c



Comparison of krsort.dir.old krsort.dir.new Page 1


directory     .
same          ./krsort
same          ./krsort.dos
same          ./krsort.q
same          ./krsortorig.c

$
```

This is a useful output. First, the files that appear in only one directory are listed. Then the files common to both directories are listed.

The following is a summary of the **dircmp** command.

dircmp - Compare directories.

Options

- d — Compare the contents of files with the same name in both directories and produce a report of what must be done to make the files identical.
- s — Suppress information about different files.

cut

There are times when you have an output that has too many fields in it. When we issued the **llsum** command earlier, it produced four fields: *FILENAME, OWNER, SIZE*, and *TYPE*. What if we want to take this output and look at just the *FILENAME* and *SIZE*. We could modify the **llsum** script, or we could use the **cut** command to eliminate the *OWNER* and *TYPE* fields with the following commands:

```
$ llsum | cut -c 1-25,37-43

FILENAME                    SIZE
README                       810
backup_files                3408
biography                    427
cshtest                     1024
gkill                       1855
gkill.out                    191
hostck                       924
ifstat                      1422
ifstat.int                  2147
ifstat.out                   723
introdos                   54018
introux                    52476
letter                     23552
letter.auto                69632
letter.auto.recover        71680
letter.backup              23552
letter.lck                    57
letter.recover             69632
llsum                       1267
llsum.orig                  1267
llsum.out                   1657
llsum.tomd.out              1356
psg                          670
psg.int                      802
psg.out                      122
sam_adduser                1010
tdolan                     1024
trash                      4554
trash.out                   329
```

```
typescript                    74

The files listed occupy 3 (0.373
Average file size is 1305
$
```



This has produced a list from **llsum**, which is piped to **cut**. Only characters *1* through *25* and *37* through *43* have been extracted. These characters correspond to the fields we want. At the end of the output are two lines that are only partially printed. We don't want these lines, so we can use **grep -v** to eliminate them and print all other lines. The output of this command is saved to the file **llsum.out** at the end of this output, which we'll use later:

```
$ ./llsum | grep -v "bytes" | cut -c 1-25,37-43

FILENAME                    SIZE
README                       810
backup_files                3408
biography                    427
cshtest                     1024
gkill                       1855
gkill.out                    191
hostck                       924
ifstat                      1422
ifstat.int                  2147
ifstat.out                   723
introdos                   54018
introux                    52476
letter                     23552
letter.auto                69632
letter.auto.recover        71680
letter.backup              23552
letter.lck                    57
letter.recover             69632
llsum                       1267
llsum.orig                  1267
llsum.out                   1657
llsum.tomd.out              1356
psg                          670
psg.int                      802
psg.out                      122
sam_adduser                1010
tdolan                     1024
trash                      4554
trash.out                   329
typescript                 1242

$ llsum | grep -v "bytes" | cut -c 1-25,37-4_3 > llsum.out
$
```

The following is a summary of the **cut** command, with some of the more commonly used options.

cut - Extract specified fields from each line.

Options

-c list	Extract based on character position, as shown in the example.
-f list	Extract based on fields.
-d char	The character following the *d* is the delimiter when using the -f option. The delimiter is the character that separates fields.

paste and join

Files can be merged together in a variety of ways. If you want to merge files on a line-by-line basis, you can use the **paste** command. The first line in the second file is pasted to the end of the first line in the first file, and so on.

Let's use the **cut** command just covered and extract only the permissions field, or characters *1* through *10*, to get only the permissions for files. We'll then save these in the file **ll.out**:

```
$ ls -al | cut -c 1-10

total 798
drwxrwxrwx
drwxrwxrwx
-rwxrwxrwx
-rwxrwxrwx
-rwxrwxrwx
drwxr-xr-x
-rwxrwxrwx
-rw-r--r--
-rwxrwxrwx
-rwxrwxrwx
-rwxr-xr-x
-rw-r--r--
-rw-r--r--
-rwxrwxrwx
-rw-r--r--
-rw-r--r--
-rw-r--r--
-rw-r--r--
-rw-rw-rw-
-rw-r--r--
-rw-r--r--
-rwxrwxrwx
-rwxr-xr-x
-rw-r--r--
-rw-r--r--
-rwxrwxrwx
-rwxr-xr-x
-rw-r--r--
-rwxrwxrwx
drwxr-xr-x
-rwxrwxrwx
-rw-r--r--
-rw-r--r--

$ ls -al | cut -c 1-10 > ll.out
$
```

We can now use the **paste** command to paste the permissions saved in the **ll.out** file to the other file-related information in the **llsum.out** file:

```
$ paste llsum.out ll.out

FILENAME                    SIZE       total 792
README                       810       -rwxrwxrwx
backup_files                3408       -rwxrwxrwx
biography                    427       -rwxrwxrwx
cshtest                     1024       drwxr-xr-x
gkill                       1855       -rwxrwxrwx
gkill.out                    191       -rw-r--r--
hostck                       924       -rwxrwxrwx
ifstat                      1422       -rwxrwxrwx
ifstat.int                  2147       -rwxr-xr-x
ifstat.out                   723       -rw-r--r--
introdos                   54018       -rw-r--r--
introux                    52476       -rwxrwxrwx
letter                     23552       -rw-r--r--
letter.auto                69632       -rw-r--r--
letter.auto.recover        71680       -rw-r--r--
letter.backup              23552       -rw-r--r--
letter.lck                    57       -rw-rw-rw-
letter.recover             69632       -rw-r--r--
ll.out                      1057       -rw-r--r--
llsum                       1267       -rwxrwxrwx
llsum.orig                  1267       -rwxr-xr-x
llsum.out                   1657       -rw-r--r--
llsum.tomd.out              1356       -rw-r--r--
psg                          670       -rwxrwxrwx
psg.int                      802       -rwxr-xr-x
psg.out                      122       -rw-r--r--
```

```
sam_adduser                 1010       -rwxrwxrwx
tdolan                      1024       drwxr-xr-x
trash                       4554       -rwxrwxrwx
trash.out                    329       -rw-r--r--
typescript                   679       -rw-r--r--

$
```

This has produced a list that includes *FILENAME* and *SIZE* from **llsum.out** and permissions from **ll.out**.

If both files have the same first field, you can use the **join** command to merge the two files:

The following is a summary of the **paste** and **join** commands, with some of the more commonly used options:

paste - Merge lines of files.

Options

-d list	Use *list* as the delimiter between columns. You can use special escape sequences for *list* such as \n for newline and \t for tab.

join - Combine two presorted files that have a common key field.

Options

-a n	Produce the normal output and also generate a line for each line that can't be joined in 1 or 2.
-e string	Replace empty fields in output with *string*.
-t char	Use *char* as the field separator.

tr

tr translates characters. **tr** is ideal for such tasks as changing case. For instance, what if you want to translate all lowercase characters to upper case? The following example shows listing files that have the suffix "zip" and then translates these files into uppercase:

```
$ ls -al *.zip
file1.zip
file2.zip
file3.zip
file4.zip
file5.zip
file6.zip
file7.zip
$ ls -al *.zip | tr "[:lower:]" "[:upper:]"
FILE1.ZIP
FILE2.ZIP
FILE3.ZIP
FILE4.ZIP
FILE5.ZIP
FILE6.ZIP
FILE7.ZIP
$
```

We use brackets in this case because we are translating a class of characters.

The following is a summary of the **tr** command, with some of the more commonly used options.

tr - Translate characters.

Options

-A	Translate on a byte-by-byte basis.

-d	Delete all occurrences of characters specified.
[:class:]	Translate from one character class to another, such as from lowercase class to uppercase class, as shown in the example.

[Team LiB]

PREVIOUS NEXT

Manual Pages for Commands Used in Chapter 23

There are manual pages in Appendix A for the following command used in this chapter:

grep



Chapter 24. Advanced UNIX Tools - Regular Expressions, sed, awk, and grep

Three Commands

Regular Expression Words-of-Caution

Expressions Are Strings and Wildcards

sed

awk

grep

Manual Pages for Commands Used in Chapter 24



Three Commands



The three commands covered in this chapter, along with regular expressions (pattern matching), are often grouped together. There are even many books available devoted to **awk** and **sed**. In these books, **grep** usually goes along for the ride because **awk** is derived to some extent from **sed** and **grep**. In addition, the use of regular expressions for pattern matching that are used for **awk**, **sed**, and **grep** are similar.





We'll take a look at regular expressions and pattern matching in general, and then cover the three commands in this chapter:

- Regular expressions
- **sed**, **awk**, and **grep** commands

Regular Expression Words-of-Caution

Regular expressions describe patterns for which you are searching. A regular expression usually defines the pattern for which you are searching using wildcards. Since a regular expression defines a pattern you are searching for, the terms "regular expression" and "pattern matching" are often used interchangably.

Let's get down to a couple of words-of-caution immediately:

- **Regular expressions are different from file matching patterns used by the shell**.



Regular expressions are used by both the shell and many other programs, including those covered in this chapter. The file matching done by the shell and programs such as **find** is different from the regular expressions covered in this chapter.

- **Use single quotes around regular expressions**. The meta-characters used in this chapter must be quoted in order to be passed to the shell as an argument. You will, therefore, see most regular expressions in this chapter quoted.

Expressions Are Strings and Wildcards



When using the programs in this book, such as **grep** and **vi**, you provide a regular expression that the program evaluates. The command will search for the pattern you supply. The pattern could be as simple as a string or it could wildcards. The wildcards used by many programs are called *meta-characters*.

Table 24-1 shows a list of meta-characters and the program(s) to which they apply. Only the programs covered in this book (**awk**, **grep**, **sed**, and **vi**) are shown in Table 24-1. These meta-characters may be used with other programs, such as **ed** and **egrep**, as well, which are not covered in the book. Table 24-1 describes the meta-characters and their use.

Table 24-1. Meta-Characters and Programs to Which They Apply

Meta-Character	awk	grep	sed	vi	Use
.	Yes	Yes	Yes	Yes	Match any single character.
*	Yes	Yes	Yes	Yes	Match any number of the single character that precedes *.
[...]	Yes	Yes	Yes	Yes	Match any *one* of the characters in the set [...].
$	Yes	Yes	Yes	Yes	Matches the end of the line.
^	Yes	Yes	Yes	Yes	Matches the beginning of the line.
\	Yes	Yes	Yes	Yes	Escape the special character that follows \.
\{*n,m*\}	Yes	Yes	No	No	Match a range of occurrences of a single character between *n* and *m*.
+	Yes	No	No	No	Match one or more occurrences of the preceding regular expression.
?	Yes	No	No	No	Match zero or one occurrence of the preceding regular expression.
|	Yes	No	No	No	The preceding or following regular expression can be matched.
()	Yes	No	No	No	Groups regular expressions in a typical parenthetical fashion.
\{ \}	No	No	No	Yes	Match a word's beginning or end.

You may want to refer to this table when regular expressions are used for one of the commands in the table.

sed



Most of the editing performed on UNIX systems is done with **vi**. I have devoted a chapter to **vi** in this book, because of its prominence as a UNIX editor. Many times, we don't have the luxury of invoking **vi** when we need to edit a file. You may be writing a shell program or piping information between processes and need to edit in a non-interactive manner. **sed** can help here. Its name comes from *stream editor,* and it's a tool for filtering text files.

You can specify the name of the file you wish to edit with **sed** or it takes its input from standard input. **sed** reads one line at a time and performs the editing you specify to each line. You can specify line numbers for **sed** to edit as well.

sed uses many of the same commands as **ed**. You can view some of the **ed** commands in the **vi** chapter, and I also supply a summary of these at the end of this **sed** section.

You can invoke **sed** in the following two ways:

```
sed [-n][-e] 'command' filename(s)
sed [-n]-f scriptfile filename(s)
```

The first form of **sed** is for issuing commands on the command line. By default, **sed** will display all lines. The *-n* specifies that you want only to print lines that are specified with the **p** command.

If you supply more than one instruction on the command line, then you *-e* is used to inform **sed** that the next argument is an instruction.

The second form allows you to specify one or more scripts containing editing commands.

The following is a summary of the three options that appear in the two different forms of **sed**:

-n	Print only lines that are specified with the **p** command.
-e command	The argument following *-e* is an editing command.
-f filename	The argument following *-f* is a file containing editing commands.



Let's view a couple of simple examples of what you can do with **sed**. These examples use some of the **sed** commands that appear at the end of this section. We'll use a file called **passwd.test**. We'll view this file with **cat** and then view only lines *16*, *17*, and *18* using the *p* option to **sed**, indicating that we want only the specified lines printed:




```
# cat passwd.test
root:PgYQCkVH65hyQ:0:0:root:/root:/bin/bash
bin:*:1:1:bin:/bin:
daemon:*:2:2:daemon:/sbin:
adm:*:3:4:adm:/var/adm:
lp:*:4:7:lp:/var/spool/lpd:
sync:*:5:0:sync:/sbin:/bin/sync
shutdown:*:6:11:shutdown:/sbin:/sbin/shutdown
halt:*:7:0:halt:/sbin:/sbin/halt
mail:*:8:12:mail:/var/spool/mail:
news:*:9:13:news:/var/spool/news:
uucp:*:10:14:uucp:/var/spool/uucp:
operator:*:11:0:operator:/root:
```

```
games:*:12:100:games:/usr/games:
gopher:*:13:30:gopher:/usr/lib/gopher-data:
ftp:*:14:50:FTP User:/home/ftp:
man:*:15:15:Manuals Owner:/:
nobody:*:65534:65534:Nobody:/:/bin/false
col:Wh0yzfAV2qm2Y:100:100:Caldera OpenLinux User:/home/col:/bin/bash
#
# sed 16,18p passwd.test
root:PgYQCkVH65hyQ:0:0:root:/root:/bin/bash
bin:*:1:1:bin:/bin:
daemon:*:2:2:daemon:/sbin:
adm:*:3:4:adm:/var/adm:
lp:*:4:7:lp:/var/spool/lpd:
sync:*:5:0:sync:/sbin:/bin/sync
shutdown:*:6:11:shutdown:/sbin:/sbin/shutdown
halt:*:7:0:halt:/sbin:/sbin/halt
mail:*:8:12:mail:/var/spool/mail:
news:*:9:13:news:/var/spool/news:
uucp:*:10:14:uucp:/var/spool/uucp:
operator:*:11:0:operator:/root:
games:*:12:100:games:/usr/games:
gopher:*:13:30:gopher:/usr/lib/gopher-data:
ftp:*:14:50:FTP User:/home/ftp:
man:*:15:15:Manuals Owner:/:
man:*:15:15:Manuals Owner:/:
nobody:*:65534:65534:Nobody:/:/bin/false
nobody:*:65534:65534:Nobody:/:/bin/false
col:Wh0yzfAV2qm2Y:100:100:Caldera OpenLinux User:/home/col:/bin/bash
col:Wh0yzfAV2qm2Y:100:100:Caldera OpenLinux User:/home/col:/bin/bash
#
# sed -n 16,18p passwd.test
man:*:15:15:Manuals Owner:/:
nobody:*:65534:65534:Nobody:/:/bin/false
col:Wh0yzfAV2qm2Y:100:100:Caldera OpenLinux User:/home/col:/bin/bash
```

The first attempt to print only lines *16*, *17*, and *18* results in all of the lines in the file being printed and lines *16, 17*, and *18* being printed twice. The reason is that **sed** reads each line of input and acts on each line. In order to specify the lines on which to act, we used the *-n* switch to suppress all lines from going to standard output. We then specify the lines that we want to print and these will indeed go to standard output.

Now that we know how to view lines 16, 17, and 18 of the file, let's again view **passwd.test** and delete those same three lines with *d*:

```
root:PgYQCkVH65hyQ:0:0:root:/root:/bin/bash
bin:*:1:1:bin:/bin:
daemon:*:2:2:daemon:/sbin:
adm:*:3:4:adm:/var/adm:
lp:*:4:7:lp:/var/spool/lpd:
sync:*:5:0:sync:/sbin:/bin/sync
shutdown:*:6:11:shutdown:/sbin:/sbin/shutdown
halt:*:7:0:halt:/sbin:/sbin/halt
mail:*:8:12:mail:/var/spool/mail:
news:*:9:13:news:/var/spool/news:
uucp:*:10:14:uucp:/var/spool/uucp:
operator:*:11:0:operator:/root:
games:*:12:100:games:/usr/games:
gopher:*:13:30:gopher:/usr/lib/gopher-data:
ftp:*:14:50:FTP User:/home/ftp:
man:*:15:15:Manuals Owner:/:
nobody:*:65534:65534:Nobody:/:/bin/false
col:Wh0yzfAV2qm2Y:100:100:Caldera OpenLinux User:/home/col:/bin/bash
#
# sed 16,18d passwd.test
root:PgYQCkVH65hyQ:0:0:root:/root:/bin/bash
bin:*:1:1:bin:/bin:
daemon:*:2:2:daemon:/sbin:
adm:*:3:4:adm:/var/adm:
lp:*:4:7:lp:/var/spool/lpd:
sync:*:5:0:sync:/sbin:/bin/sync
shutdown:*:6:11:shutdown:/sbin:/sbin/shutdown
halt:*:7:0:halt:/sbin:/sbin/halt
mail:*:8:12:mail:/var/spool/mail:
news:*:9:13:news:/var/spool/news:
```

```
uucp:*:10:14:uucp:/var/spool/uucp:
operator:*:11:0:operator:/root:
games:*:12:100:games:/usr/games:
gopher:*:13:30:gopher:/usr/lib/gopher-data:
ftp:*:14:50:FTP User:/home/ftp:
```



As with our earlier **grep** example, we enclose any special characters in single quotes to make sure that they are not interfered with and are passed directly to **sed** unmodified and uninterpreted by the shell. In this example, we specify the range of lines to delete, *16* through *18*, and the *d* for delete. We could specify just one line to delete, such as *16*, and not specify an entire range. Because we did not redirect the output as part of the **sed** command line, the result is sent to standard output. The original file remains intact.



We could search for a pattern in a file and delete only those lines containing the pattern. The following example shows searching for *bash* and deleting the lines that contain *bash:*

```
# cat passwd.test
root:PgYQCkVH65hyQ:0:0:root:/root:/bin/bash
bin:*:1:1:bin:/bin:
daemon:*:2:2:daemon:/sbin:
adm:*:3:4:adm:/var/adm:
lp:*:4:7:lp:/var/spool/lpd:
sync:*:5:0:sync:/sbin:/bin/sync
shutdown:*:6:11:shutdown:/sbin:/sbin/shutdown
halt:*:7:0:halt:/sbin:/sbin/halt
mail:*:8:12:mail:/var/spool/mail:
news:*:9:13:news:/var/spool/news:
uucp:*:10:14:uucp:/var/spool/uucp:
operator:*:11:0:operator:/root:
games:*:12:100:games:/usr/games:
gopher:*:13:30:gopher:/usr/lib/gopher-data:
ftp:*:14:50:FTP User:/home/ftp:
man:*:15:15:Manuals Owner:/:
nobody:*:65534:65534:Nobody:/:/bin/false
col:Wh0yzfAV2qm2Y:100:100:Caldera OpenLinux User:/home/col:/bin/bash
#
# sed '/bash/ d' passwd.test
bin:*:1:1:bin:/bin:
daemon:*:2:2:daemon:/sbin:
adm:*:3:4:adm:/var/adm:
lp:*:4:7:lp:/var/spool/lpd:
sync:*:5:0:sync:/sbin:/bin/sync
shutdown:*:6:11:shutdown:/sbin:/sbin/shutdown
halt:*:7:0:halt:/sbin:/sbin/halt
mail:*:8:12:mail:/var/spool/mail:
news:*:9:13:news:/var/spool/news:
uucp:*:10:14:uucp:/var/spool/uucp:
operator:*:11:0:operator:/root:
games:*:12:100:games:/usr/games:
gopher:*:13:30:gopher:/usr/lib/gopher-data:
ftp:*:14:50:FTP User:/home/ftp:
man:*:15:15:Manuals Owner:/:
nobody:*:65534:65534:Nobody:/:/bin/false
```

Both lines containing *bash* were deleted from **passwd.test** (the *root* line and the *col* line).

As I had mentioned earlier, it is a good idea to use single quotes around all regular expressions. In this example, I enclosed in single quotes the pattern for which I was searching and the command to execute.

What if you wanted to delete all lines except those that contain *bash*? You would insert an exclamation mark before the *d* to delete all lines except those that contain *bash,* as shown in the following example:

```
# cat passwd.test
root:PgYQCkVH65hyQ:0:0:root:/root:/bin/bash
bin:*:1:1:bin:/bin:
daemon:*:2:2:daemon:/sbin:
adm:*:3:4:adm:/var/adm:
```

```
lp:*:4:7:lp:/var/spool/lpd:
sync:*:5:0:sync:/sbin:/bin/sync
shutdown:*:6:11:shutdown:/sbin:/sbin/shutdown
halt:*:7:0:halt:/sbin:/sbin/halt
mail:*:8:12:mail:/var/spool/mail:
news:*:9:13:news:/var/spool/news:
uucp:*:10:14:uucp:/var/spool/uucp:
operator:*:11:0:operator:/root:
games:*:12:100:games:/usr/games:
gopher:*:13:30:gopher:/usr/lib/gopher-data:
ftp:*:14:50:FTP User:/home/ftp:
man:*:15:15:Manuals Owner:/:
nobody:*:65534:65534:Nobody:/:/bin/false
col:Wh0yzfAV2qm2Y:100:100:Caldera OpenLinux User:/home/col:/bin/bash
#
# sed "/bash/ !d" passwd.test
root:PgYQCkVH65hyQ:0:0:root:/root:/bin/bash
col:Wh0yzfAV2qm2Y:100:100:Caldera OpenLinux User:/home/col:/bin/bash
```

This resulted in all but the two lines containing *bash* to be deleted from **passwd.test**.

Now that we have seen how to display and delete specific lines of the file, let's see how to add three lines to the end of the file:

```
# sed "$a\
> This is a backup of passwd file\
> for viewing purposes only\
> so do not modify" passwd.test
root:PgYQCkVH65hyQ:0:0:root:/root:/bin/bash
bin:*:1:1:bin:/bin:
daemon:*:2:2:daemon:/sbin:
adm:*:3:4:adm:/var/adm:
lp:*:4:7:lp:/var/spool/lpd:
sync:*:5:0:sync:/sbin:/bin/sync
shutdown:*:6:11:shutdown:/sbin:/sbin/shutdown
halt:*:7:0:halt:/sbin:/sbin/halt
mail:*:8:12:mail:/var/spool/mail:
news:*:9:13:news:/var/spool/news:
uucp:*:10:14:uucp:/var/spool/uucp:
operator:*:11:0:operator:/root:
games:*:12:100:games:/usr/games:
gopher:*:13:30:gopher:/usr/lib/gopher-data:
ftp:*:14:50:FTP User:/home/ftp:
man:*:15:15:Manuals Owner:/:
nobody:*:65534:65534:Nobody:/:/bin/false
col:Wh0yzfAV2qm2Y:100:100:Caldera OpenLinux User:/home/col:/bin/bash
This is a backup of passwd file
for viewing purposes only
so do not modify
```

The backslashes (\) are used liberally in this example. Each backslash represents a new line. We go to the end of the file, as designated by the $, then we add a new line with the backslash, and then add the text we wish and a new line after the text. These lines are great to add to the end of the file, but we really should add them to the beginning of the file. The following example shows this approach:

```
# sed "1i\
> This is a backup passwd file\
> for viewing purposes only\
> so do not modify\
> " passwd.test
This is a backup passwd file
for viewing purposes only
so do not modify
root:PgYQCkVH65hyQ:0:0:root:/root:/bin/bash
bin:*:1:1:bin:/bin:
daemon:*:2:2:daemon:/sbin:
adm:*:3:4:adm:/var/adm:
lp:*:4:7:lp:/var/spool/lpd:
sync:*:5:0:sync:/sbin:/bin/sync
shutdown:*:6:11:shutdown:/sbin:/sbin/shutdown
halt:*:7:0:halt:/sbin:/sbin/halt
mail:*:8:12:mail:/var/spool/mail:
news:*:9:13:news:/var/spool/news:
uucp:*:10:14:uucp:/var/spool/uucp:
operator:*:11:0:operator:/root:
```

```
games:*:12:100:games:/usr/games:
gopher:*:13:30:gopher:/usr/lib/gopher-data:
ftp:*:14:50:FTP User:/home/ftp:
man:*:15:15:Manuals Owner:/:
nobody:*:65534:65534:Nobody:/:/bin/false
col:Wh0yzfAV2qm2Y:100:100:Caldera OpenLinux User:/home/col:/bin/bash
```



First, we run **sed**, specifying that on the first line one we are going to begin inserting the text shown. We use the single quote immediately following **sed** and use another single quote on the last line when we are done specifying all the information, except for the input file, which is **passwd.test**.

We have only scratched the surface of commands you can use with **sed**. The following **sed** summary includes the commands we have used (*p* for print; *d* for delete; and *a* for add), as well as others that were not part of the examples.

sed - 

Stream editor.

Commands

a	Append text.
b	Branch to a label.
c	Replace lines with text.
d	Delete the current text buffer.
D	Delete the first line of the current text buffer.
g	Paste overwriting contents of the hold space.
G	Paste the hold space below the address rather than overwriting it.
h	Copy the pattern space into hold space.
H	Append the contents of pattern space into hold space.
i	Insert text.
L	List the contents of the pattern space.
n	Read the next line of input into the pattern space.
N	Append next line of input to pattern space.
p	Print the pattern space.
P	Print from the start of the pattern space up to and including new line.
q	Quit when address is encountered.
r	Read in a file.
s	Substitute patterns.
t	Branch if substitution has been made to the current pattern space.

w Append the contents of the pattern space to the specified *file*.

x Interchange the contents of the holding area and pattern space.

y Translate characters.

awk



awk can pretty much do it all. With **awk**, you can search, modify files, generate reports, and a lot more. **awk** performs these tasks by searching for patterns in lines of input (from standard input or from a file). For each line that matches the specified pattern, it can perform some very complex processing on that line. The code to process matching lines of input is a cross between a shell script and a C program.

Data manipulation tasks that would be very complex with combinations of **grep**, **cut,** and **paste** are very easily done with **awk**. Because **awk** is a programming language, it can also perform mathematical operations or check the input very easily (shells don't do math very well). It can even do floating-point math (shells deal only with integers and strings).

The basic form of an **awk** program looks like this:

```
awk '/pattern_to_match/ {prog to run}' input_file_names
```

Notice that the entire program is enclosed in single quotes. If no input file names are specified, **awk** reads from standard input (as from a pipe).

The *pattern_to_match* must appear between the **/** characters. The pattern is actually a regular expression. Regular expressions were covered earlier in this chapter. Some common regular expression examples will be given shortly.

The program to execute is written in **awk** code, which looks something like C. The program is executed whenever a line of input matches the *pattern_to_match*. If */pattern_to_match/* does not precede the program in braces *{ }*, then the program is executed for every line of input.

awk works with fields of the input lines. Fields are words separated by white space or some other field separator. **awk** uses white space as a field separator by default. You can use the *-F* option to specify the field separator as shown in a later example. The fields in **awk** patterns and programs are referenced with *$*, followed by the field number. For example, the second field of an input line is *$2*. If you are using an **awk** command in your shell programs, the fields (*$1, $2*, etc.) are not confused with the shell script's positional parameters because the **awk** variables are enclosed in single quotes ' causing the shell to ignore them.

You really need to see some examples of using **awk** to appreciate its power. The following few examples use a file called **newfiles**, which contains a list of files on a system less than 15 days old. This file is generated as part of a system administration audit program that checks various aspects of a UNIX system. The following shows the contents of **newfiles**:

```
# cat newfiles
PROG>>>>> report of files not older than 14 days by find
the file system is /
-rw-r--r--   1 root       root           567 Dec  7 07:16 ./etc/mnttab
-rw-r--r--   1 root       root         20713 Dec  7 07:18 ./etc/rc.log
-rw-r--r--   1 root      root            0 Dec  7 07:17 ./etc/hpC2400/hparray.map
-rw-r--r--   1 root      root            0 Dec  7 07:17 ./etc/hpC2400/hparray.devs
-rw-r--r--   1 root      root            0 Dec  7 07:17 ./etc/hpC2400/hparray.luns
-rw-r--r--   1 root      root            0 Dec  7 07:17 ./etc/hpC2400/hparray.addr
-r-s------   1 root       root            0 Dec  7 07:17 ./etc/hpC2400/pscan.lock
-r-s------   1 root      root            0 Dec  7 07:17 ./etc/hpC2400/monitor.lock
-rw-r--r--   1 root      root        14299 Dec  7 07:17 ./etc/hpC2400/HPARRAY.INFO
-rw-r--r--   1 bin        bin            8553 Dec  7 07:02 ./etc/shutdownlog
-rw-r--r--   1 root       mail          32768 Dec  7 07:16 ./etc/mail/aliases.db
-rw-r--r--   1 root      mail             33 Dec  7 07:16 ./etc/mail/sendmail.pid
-rw-r--r--   1 root      root             13 Dec  7 07:16 ./etc/opt/dce/boot_time
-rw-r--r--   1 root      root            720 Dec  7 13:34 ./etc/utmp
-rw-r--r--   1 root      root              0 Dec  7 07:16 ./etc/xtab
-rw-r--r--   1 root      root              0 Dec  7 07:18 ./etc/rmtab
```

```
-rw-r--r--   1 root      root         40814 Dec  7 07:15 ./etc/rc.log.old
-rw-r--r--   1 root      root          4620 Dec  7 13:34 ./etc/utmpx
-rw-r--r--   1 root      root             9 Dec  7 13:17 ./etc/ntp.drift
-rw-r--r--   1 root      root           616 Dec  7 07:15 ./etc/auto_parms.log
-rw-r--r--   1 root     sys             219 Dec  7 07:00 ./etc/auto_parms.log.old
-rw-rw-rw-   1 root    sys             520 Nov 23 12:37 ./.sw/sessions/swlist.last
-r--r--r--   1 root    informix         76 Dec  7 07:17 ./INFORMIXTMP/.inf.shmPSREP
-r--r--r--   1 root    informix         76 Dec  7 07:18 ./INFORMIXTMP/.inf.shmPSDEV
-rw-------   1 autosys  autosys       4052 Nov 25 14:08 ./home/autosys/.sh_history
-rw-------   1 tsaxs     users          2228 Dec  1 13:15 ./home/tsaxs/.sh_history
-rw-------   1 tsfxo     users          2862 Nov 24 10:08 ./home/tsfxo/.sh_history
PROG>>>>> report of files not older than 14 days by find
the file system is /usr
-rw-rw-rw-   1 opop6   users            21 Dec  7 13:46 ./local/adm/etc/lmonitor.hst
-rw-r--r--   1 tsgjf     users          1093 Dec  7 13:17
./local/flexlm/licenses/license.log
PROG>>>>> report of files not older than 14 days by find
the file system is /opt
-rw-rw-r--   1 bin        bin             200 Dec  7 07:17 ./pred/bin/OPSDBPF
-rw-r--r--   1 root       sys          800028 Dec  7 07:17 ./pred/bin/PSRNLOGD
PROG>>>>> report of files not older than 14 days by find
the file system is /var
-rw-r--r--   1 root      sys           45089 Dec  7 07:16 ./adm/sw/swagentd.log
-rw-rw-rw-   1 root    sys         562 Dec  7 07:16 ./adm/sw/sessions/swlist.last
-rw-rw-r--   1 root      root          12236 Dec  7 07:16 ./adm/ps_data
-rw-r--r--   1 root      root             65 Dec  7 07:17 ./adm/cron/log
-rw-r--r--   1 root      root            162 Dec  7 07:00 ./adm/cron/OLDlog
-r--r--r--   1 root      root         734143 Dec  7 07:16 ./adm/syslog/mail.log
-rw-r--r--   1 root      root          65743 Dec  7 13:56 ./adm/syslog/syslog.log
-rw-r--r--   1 root     root        4924974 Dec  7 07:02 ./adm/syslog/OLDsyslog.log
-rw-rw-r--   1 adm        adm         2750700 Dec  7 13:52 ./adm/wtmp
-rw-------   1 root       other        145920 Dec  3 14:36 ./adm/btmp
-rw-r--r--   1 lp         lp              33 Dec  7 07:17 ./adm/lp/log
-rw-r--r--   1 lp         lp              67 Dec  7 07:01 ./adm/lp/oldlog
-rw-r--r--   1 root      root           4330 Dec  7 07:18 ./adm/diag/device_table
-rw-r--r--   1 root      root             34 Dec  7 07:18 ./adm/diag/misc_sys_data
-rwxr-xr-x   1 root       root         995368 Nov 22 15:16 ./adm/diag/LOG0190
-rwxr-xr-x   1 root       root         995368 Nov 23 02:05 ./adm/diag/LOG0191
-rwxr-xr-x   1 root       root         453964 Nov 23 07:01 ./adm/diag/LOG0192
-rwxr-xr-x   1 root       root         970448 Nov 23 18:35 ./adm/diag/LOG0193
-rwxr-xr-x   1 root       root         995368 Nov 24 05:24 ./adm/diag/LOG0194
-rwxr-xr-x   1 root       root         995368 Nov 24 16:14 ./adm/diag/LOG0195
-rwxr-xr-x   1 root       root         995368 Nov 25 03:03 ./adm/diag/LOG0196
-rwxr-xr-x   1 root       root         995368 Nov 25 13:52 ./adm/diag/LOG0197
-rwxr-xr-x   1 root       root         995368 Nov 26 00:41 ./adm/diag/LOG0198
-rwxr-xr-x   1 root       root         995368 Nov 26 11:31 ./adm/diag/LOG0199
-rwxr-xr-x   1 root       root         995368 Nov 26 22:20 ./adm/diag/LOG0200
-rwxr-xr-x   1 root       root         995368 Nov 27 09:09 ./adm/diag/LOG0201
-rwxr-xr-x   1 root       root         995368 Nov 27 19:58 ./adm/diag/LOG0202
-rwxr-xr-x   1 root       root         995368 Nov 28 06:48 ./adm/diag/LOG0203
-rwxr-xr-x   1 root       root         995368 Nov 28 17:37 ./adm/diag/LOG0204
-rwxr-xr-x   1 root       root         995368 Nov 29 04:26 ./adm/diag/LOG0205
-rwxr-xr-x   1 root       root         995368 Nov 29 15:16 ./adm/diag/LOG0206
-rwxr-xr-x   1 root       root         995368 Nov 30 02:05 ./adm/diag/LOG0207
-rwxr-xr-x   1 root       root         452020 Nov 30 06:59 ./adm/diag/LOG0208
-rwxr-xr-x   1 root       root         970448 Nov 30 18:35 ./adm/diag/LOG0209
-rwxr-xr-x   1 root       root         995368 Dec  1 05:24 ./adm/diag/LOG0210
-rwxr-xr-x   1 root       root         995368 Dec  1 16:13 ./adm/diag/LOG0211
-rwxr-xr-x   1 root       root         995368 Dec  2 03:03 ./adm/diag/LOG0212
-rwxr-xr-x   1 root       root         995368 Dec  2 13:52 ./adm/diag/LOG0213
-rwxr-xr-x   1 root       root         995368 Dec  3 00:41 ./adm/diag/LOG0214
-rwxr-xr-x   1 root       root         995368 Dec  3 11:31 ./adm/diag/LOG0215
-rwxr-xr-x   1 root       root         995368 Dec  3 22:20 ./adm/diag/LOG0216
-rwxr-xr-x   1 root       root         995368 Dec  4 09:09 ./adm/diag/LOG0217
-rwxr-xr-x   1 root       root         995368 Dec  4 19:58 ./adm/diag/LOG0218
-rwxr-xr-x   1 root       root         995368 Dec  5 06:48 ./adm/diag/LOG0219
-rwxr-xr-x   1 root       root         995368 Dec  5 17:37 ./adm/diag/LOG0220
-rwxr-xr-x   1 root       root         995368 Dec  6 04:26 ./adm/diag/LOG0221
-rwxr-xr-x   1 root       root         995368 Dec  6 15:15 ./adm/diag/LOG0222
-rwxr-xr-x   1 root       root         995368 Dec  7 02:05 ./adm/diag/LOG0223
-rwxr-xr-x   1 root       root         453964 Dec  7 07:00 ./adm/diag/LOG0224
-rwxr-xr-x   1 root       root         543740 Dec  7 13:57 ./adm/diag/LOG0225
-rw-r--r--   1 root       root          19587 Dec  7 07:16 ./adm/ptydaemonlog
-rw-r--r--   1 root       root             52 Dec  7 07:16 ./adm/conslog.opts
-rw-r--r--   1 root       root              0 Dec  7 07:16 ./adm/rpc.statd.log
```

```
-rw-r--r--   1 root       root             0 Dec  7 07:16 ./adm/rpc.lockd.log
-rw-r--r--   1 root       root         24250 Dec  7 07:16 ./adm/vtdaemonlog
-rw-------   1 root       root           214 Dec  7 12:07 ./adm/sulog
-rw-------   1 root       root           381 Dec  3 17:34 ./adm/OLDsulog
-rw-r--r--   1 root       sys            145 Dec  7 07:16 ./adm/rbootd.log
-rw-------   1 sysadm     psoft           60 Dec  1 16:59 ./tmp/EAAa09057
-rw-r--r--   1 tsgjf      users            0 Dec  7 13:17 ./tmp/lockHPCUPLANGS
-rw-r--r--   1 tsgjf     users           175 Dec  7 06:40 ./tmp/.flexlm/lmgrd.1507
-rw-r--r--   1 tsgjf     users           175 Dec  7 13:28 ./tmp/.flexlm/lmgrd.1505
-rw-r--r--   1 lp         lp               0 Dec  7 07:17 ./spool/lp/outputq
-rw-rw-rw-   1 lp         lp               4 Dec  7 07:17 ./spool/lp/SCHEDLOCK
-rw-------   1 root       sys              0 Nov 23 07:00
./spool/cron/tmp/croutAAAa01030
-rw-------   1 root       sys              0 Nov 30 07:00
./spool/cron/tmp/croutAAAa01039
-rw-------   1 root       sys              0 Dec  7 07:00
./spool/cron/tmp/croutAAAb01039
-rw-r--r--   1 root       root             4 Dec  7 07:16 ./run/syslog.pid
-rw-r--r--   1 root       root             4 Dec  7 07:16 ./run/gated.pid
-rw-r--r--   1 root       sys            145 Dec  7 07:16 ./run/gated.version
-rw-r--r--   1 root       sys              3 Dec  7 07:16 ./statmon/state
-rw-r--r--   1 root       root         29771 Dec  7 07:16
./opt/dce/config/dce_config.log
-rw-r--r--   1 root       sys             74 Dec  7 07:16
./opt/dce/rpc/local/00404/srvr_socks
-rw-r--r--   1 root       root            72 Dec  7 07:16
./opt/dce/rpc/local/00927/srvr_socks
-rw-r--r--   1 root       root         32768 Dec  7 07:16 ./opt/dce/dced/Ep.db
-rw-r--r--   1 root       root         32768 Dec  7 07:20 ./opt/dce/dced/Llb.db
-rw-r--r--   1 root       root             0 Nov 30 07:16 ./opt/perf/status.ttd
-rw-r--r--   1 root      root             33 Dec  7 07:17 ./opt/perf/datafiles/RUN
-rwxrwxrwx   1 root    sys        9243180 Dec  7 13:55 ./opt/perf/datafiles/logappl
-rwxrwxrwx   1 root     sys        8697612 Dec  7 13:55 ./opt/perf/datafiles/logdev
-rwxrwxrwx   1 root    sys        9195152 Dec  7 13:55 ./opt/perf/datafiles/logglob
-rwxrwxrwx   1 root    sys          11112 Dec  7 07:17 ./opt/perf/datafiles/logindx
-rwxrwxrwx   1 root       sys          17639080 Dec  7 13:57
./opt/perf/datafiles/logproc
-rwxrwxrwx   1 root       sys             3797 Dec  7 07:17
./opt/perf/datafiles/mikslp.data
-rw-rw-rw-   1 root      sys            105 Nov 30 10:45 ./opt/perf/datafiles/agdb
-rw-r--r--   1 root        root             5 Dec  7 07:17
./opt/perf/datafiles/.perflbd.pid
-rw-rw-rw-   1 root       sys          21176 Dec  7 07:20 ./opt/perf/status.scope
-rw-rw-rw-   1 root        root            5 Nov 30 07:16 ./opt/perf/ttd.pid
-rw-r--r--   1 root        root            0 Dec  7 07:17 ./opt/perf/status.mi
-rw-rw-rw-   1 root      sys            8254 Dec  7 07:17 ./opt/perf/status.perflbd
-rw-rw-rw-   1 root     sys           21507 Dec  7 07:20 ./opt/perf/status.rep_server
-rw-rw-rw-   1 root      sys           24570 Dec  7 07:20 ./opt/perf/status.alarmgen
-rw-rw-rw-   1 root        sys         160956 Dec  6 21:13 ./opt/omni/log/inet.log
-rw-rw-rw-   1 root        sys         158796 Dec  7 07:17 ./sam/log/samlog
-rw-r--r--   1 root        root         64730 Dec  7 07:17 ./sam/boot.config
-rw-rw-rw-   1 root        sys          11906 Nov 24 14:27 ./sam/poe.iout
-rw-rw-rw-   1 root        sys          11906 Nov 23 09:10 ./sam/poe.iout.old
-rw-rw-rw-   1 root        sys             29 Nov 24 14:27 ./sam/poe.dion
```



You can see that this file contains several fields separated by whitespace. The next example evaluates the third field to determine whether it equals "adm," and if so, the line is printed:

```
# awk '$3 == "adm" {print}' newfiles
-rw-rw-r--   1 adm        adm        2750700 Dec  7 13:52 ./adm/wtmp
```

There is precisely one line that contains exactly "adm" in the third field.

The next example evaluates the third field to determine whether it approximately equals "adm," meaning that the third field has "adm" embedded in it, and if so, the line is printed:

```
# awk '$3 ~ "adm" {print}' newfiles
-rw-rw-r--   1 adm        adm        2750700 Dec  7 13:52 ./adm/wtmp
-rw-------   1 sysadm     psoft           60 Dec  1 16:59 ./tmp/EAAa09057
```

This result prints the line from the last example, which has "adm" in the third field as well as a line that contains "sysadm."

The next example performs the same search as the previous example; however, this time only fields nine and five are printed:

```
# awk '$3 ~ "adm" {print $9, $5}' newfiles
./adm/wtmp 2750700
./tmp/EAAa09057 60
```

This time only the name of the file, field nine, and the size of the file were printed.

The next example evaluates the third field to determine if it does not equal "root," and if so, prints the entire line:

```
# awk '$3 != "root" {print}' newfiles
PROG>>>>> report of files not older than 14 days by find
the file system is /
-rw-r--r--   1 bin          bin           8553 Dec  7 07:02 ./etc/shutdownlog
-rw-------   1 autosys    autosys       4052 Nov 25 14:08 ./home/autosys/.sh_history
-rw-------   1 tsaxs       users         2228 Dec  1 13:15 ./home/tsaxs/.sh_history
-rw-------   1 tsfxo       users         2862 Nov 24 10:08 ./home/tsfxo/.sh_history
PROG>>>>> report of files not older than 14 days by find
the file system is /usr
-rw-rw-rw-   1 opop6      users           21 Dec  7 13:46 ./local/adm/etc/lmonitor.hst
-rw-r--r--   1 tsgjf        users         1093 Dec  7 13:17
./local/flexlm/licenses/license.log
PROG>>>>> report of files not older than 14 days by find
the file system is /opt
-rw-rw-r--   1 bin        bin            200 Dec  7 07:17 ./pred/bin/OPSDBPF
PROG>>>>> report of files not older than 14 days by find
the file system is /var
-rw-rw-r--   1 adm        adm        2750700 Dec  7 13:52 ./adm/wtmp
-rw-r--r--   1 lp         lp              33 Dec  7 07:17 ./adm/lp/log
-rw-r--r--   1 lp         lp              67 Dec  7 07:01 ./adm/lp/oldlog
-rw-------   1 sysadm     psoft           60 Dec  1 16:59 ./tmp/EAAa09057
-rw-r--r--   1 tsgjf      users            0 Dec  7 13:17 ./tmp/lockHPCUPLANGS
-rw-r--r--   1 tsgjf     users          175 Dec  7 06:40 ./tmp/.flexlm/lmgrd.1507
-rw-r--r--   1 tsgjf     users          175 Dec  7 13:28 ./tmp/.flexlm/lmgrd.1505
-rw-r--r--   1 lp         lp              0 Dec  7 07:17 ./spool/lp/outputq
-rw-rw-rw-   1 lp         lp              4 Dec  7 07:17 ./spool/lp/SCHEDLOCK
```

This command results in many lines being printed that do not have "root" in the third field.

newfiles had whitespace to separate the fields. We don't often have this luxury in the UNIX world. The upcoming examples use **passwd.test**, which has a colon(:) as a field separator. **passwd.test** is shown below:

```
# cat passwd.test
root:PgYQCkVH65hyQ:0:0:root:/root:/bin/bash
bin:*:1:1:bin:/bin:
daemon:*:2:2:daemon:/sbin:
adm:*:3:4:adm:/var/adm:
lp:*:4:7:lp:/var/spool/lpd:
sync:*:5:0:sync:/sbin:/bin/sync
shutdown:*:6:11:shutdown:/sbin:/sbin/shutdown
halt:*:7:0:halt:/sbin:/sbin/halt
mail:*:8:12:mail:/var/spool/mail:
news:*:9:13:news:/var/spool/news:
uucp:*:10:14:uucp:/var/spool/uucp:
operator:*:11:0:operator:/root:
games:*:12:100:games:/usr/games:
gopher:*:13:30:gopher:/usr/lib/gopher-data:
ftp:*:14:50:FTP User:/home/ftp:
man:*:15:15:Manuals Owner:/:
nobody:*:65534:65534:Nobody:/:/bin/false
col:Wh0yzfAV2qm2Y:100:100:Caldera OpenLinux User:/home/col:/bin/bash
```

You can specify the field separator with the *-F* option followed by a separator, which is a colon(:) in **passwd.test**. The following example specifies the field separator and then evaluates the first field to determine whether it equals "root," and if so, prints out the entire line:

```
# awk -F: '$1 == "root" {print}' passwd.test
root:PgYQCkVH65hyQ:0:0:root:/root:/bin/bash
```

The following example specifies the field separator and then evaluates the fourth field to determine whether it equals "0," which means that the user is a member of the same group as root, and if so, prints out the entire line:

```
# awk -F: "$4 == "0" {print}" passwd.test
root:PgYQCkVH65hyQ:0:0:root:/root:/bin/bash
sync:*:5:0:sync:/sbin:/bin/sync
halt:*:7:0:halt:/sbin:/sbin/halt
operator:*:11:0:operator:/root:
```



You can perform many types of comparisons besides == using **awk**. The following examples show the use of several comparison operators on our trusty **passwd.test** file. The first example prints all users who are in a group with a value less than *14*:

```
# awk -F: "$4 < 14 {print}" passwd.test
root:PgYQCkVH65hyQ:0:0:root:/root:/bin/bash
bin:*:1:1:bin:/bin:
daemon:*:2:2:daemon:/sbin:
adm:*:3:4:adm:/var/adm:
lp:*:4:7:lp:/var/spool/lpd:
sync:*:5:0:sync:/sbin:/bin/sync
shutdown:*:6:11:shutdown:/sbin:/sbin/shutdown
halt:*:7:0:halt:/sbin:/sbin/halt
mail:*:8:12:mail:/var/spool/mail:
news:*:9:13:news:/var/spool/news:
operator:*:11:0:operator:/root:
```

The next example prints all users who are in a group with a value less than or equal to *14*:

```
# awk -F: "$4 <= 14 {print}" passwd.test
root:PgYQCkVH65hyQ:0:0:root:/root:/bin/bash
bin:*:1:1:bin:/bin:
daemon:*:2:2:daemon:/sbin:
adm:*:3:4:adm:/var/adm:
lp:*:4:7:lp:/var/spool/lpd:
sync:*:5:0:sync:/sbin:/bin/sync
shutdown:*:6:11:shutdown:/sbin:/sbin/shutdown
halt:*:7:0:halt:/sbin:/sbin/halt
mail:*:8:12:mail:/var/spool/mail:
news:*:9:13:news:/var/spool/news:
uucp:*:10:14:uucp:/var/spool/uucp:
operator:*:11:0:operator:/root:
```

Let's now print all users who are in a group that does not have a value of *14*:

```
# awk -F: "$4 != 14 {print}" passwd.test
root:PgYQCkVH65hyQ:0:0:root:/root:/bin/bash
bin:*:1:1:bin:/bin:
daemon:*:2:2:daemon:/sbin:
adm:*:3:4:adm:/var/adm:
lp:*:4:7:lp:/var/spool/lpd:
sync:*:5:0:sync:/sbin:/bin/sync
shutdown:*:6:11:shutdown:/sbin:/sbin/shutdown
halt:*:7:0:halt:/sbin:/sbin/halt
mail:*:8:12:mail:/var/spool/mail:
news:*:9:13:news:/var/spool/news:
operator:*:11:0:operator:/root:
games:*:12:100:games:/usr/games:
gopher:*:13:30:gopher:/usr/lib/gopher-data:
ftp:*:14:50:FTP User:/home/ftp:
man:*:15:15:Manuals Owner:/:
nobody:*:65534:65534:Nobody:/:/bin/false
col:Wh0yzfAV2qm2Y:100:100:Caldera OpenLinux User:/home/col:/bin/bash
```

Let's now print all users who are in a group with a value greater than or equal to *14*:

```
# awk -F: "$4 >= 14 {print}" passwd.test
uucp:*:10:14:uucp:/var/spool/uucp:
games:*:12:100:games:/usr/games:
gopher:*:13:30:gopher:/usr/lib/gopher-data:
```

```
ftp:*:14:50:FTP User:/home/ftp:
man:*:15:15:Manuals Owner:/:
nobody:*:65534:65534:Nobody:/:/bin/false
col:Wh0yzfAV2qm2Y:100:100:Caldera OpenLinux User:/home/col:/bin/bash
```

The last example shows all users who are in a group with a value greater than 14:

```
# awk -F: "$4 > 14 {print}" passwd.test
games:*:12:100:games:/usr/games:
gopher:*:13:30:gopher:/usr/lib/gopher-data:
ftp:*:14:50:FTP User:/home/ftp:
man:*:15:15:Manuals Owner:/:
nobody:*:65534:65534:Nobody:/:/bin/false
col:Wh0yzfAV2qm2Y:100:100:Caldera OpenLinux User:/home/col:/bin/bash
```



There is much more to **awk** than what I covered in this section. There are additional **awk** examples in the shell programming Chapter 28.

The following table summarizes some of the comparison operators of **awk** covered in this section:

awk - Search a line for a specified pattern and perform operation(s).

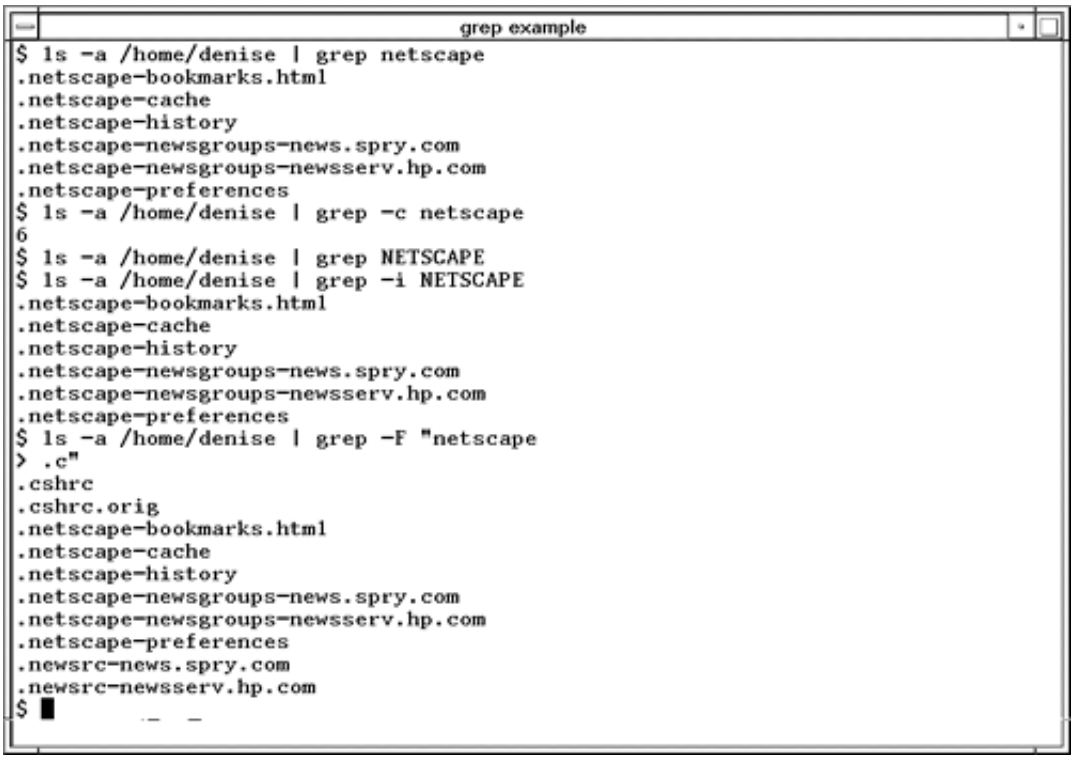
Comparison operators:	
<	Less than.
<=	Less than or equal to.
==	Equal to.
~	Strings match.
!=	Not equal to.
>=	Greater than or equal to.
>	Greater than.

grep



Here in the information age, we have too much information. We are constantly trying to extract the information we are after from stacks of information. The **grep** command is used to search for text and display it. **grep** stands for General Regular Expression Parser. Let's first look at a few simple searches and display the output with **grep**. Figure 24-1 shows creating a long listing for **/home/denise**, and using **grep**, we search for patterns.

Figure 24-1. grep Command



First, we search for the pattern *netscape*. This produces a list of files, all of which begin with *.netscape*.

Next we use the *-c* option to create a count for the number of times that *netscape* is found. The result is 6.



Do you think that **grep** is case-sensitive? The next example shows searching for the pattern *NETSCAPE,* and no matching patterns exist.

Using the *-i* option causes **grep** to ignore uppercase and lower case and just search for the pattern, and again, all the original matches are found.

Also, more than one pattern can be searched for. Using the **-F** option, both *netscape* and *.c* are searched for and a longer list of matches are found. Notice that two patterns to search for are enclosed in double quotes and are separated by a new line.

Let's now take a look at a several more advanced searches using **grep**. We'll use the **passwd.test** file as

the basis for our searches because each line in it contains a lot of information. To start, the following is the contents of the **passwd.test** file on a Linux system:

```
# cat passwd.test
root:PgYQCkVH65hyQ:0:0:root:/root:/bin/bash
bin:*:1:1:bin:/bin:
daemon:*:2:2:daemon:/sbin:
adm:*:3:4:adm:/var/adm:
lp:*:4:7:lp:/var/spool/lpd:
sync:*:5:0:sync:/sbin:/bin/sync
shutdown:*:6:11:shutdown:/sbin:/sbin/shutdown
halt:*:7:0:halt:/sbin:/sbin/halt
mail:*:8:12:mail:/var/spool/mail:
news:*:9:13:news:/var/spool/news:
uucp:*:10:14:uucp:/var/spool/uucp:
operator:*:11:0:operator:/root:
games:*:12:100:games:/usr/games:
gopher:*:13:30:gopher:/usr/lib/gopher-data:
ftp:*:14:50:FTP User:/home/ftp:
man:*:15:15:Manuals Owner:/:
nobody:*:65534:65534:Nobody:/:/bin/false
col:Wh0yzfAV2qm2Y:100:100:Caldera OpenLinux
    User:/home/col:/bin/bash
```



We can search for a string in the password file just as we did in the earlier **grep** example. The following example searches for *news* in the **passwd.test** file:

```
# grep news passwd.test
news:*:9:13:news:/var/spool/news:
```

Now let's check to see whether there is a user named *bin* in the **passwd.test** file. In order for a user named *bin* to have an entry in the **passwd.test** file, the user name, in this case *bin*, would be the first entry in the line. Here is the result of searching for this user:

```
# grep bin passwd.test
root:PgYQCkVH65hyQ:0:0:root:/root:/bin/bash
bin:*:1:1:bin:/bin:
daemon:*:2:2:daemon:/sbin:
sync:*:5:0:sync:/sbin:/bin/sync
shutdown:*:6:11:shutdown:/sbin:/sbin/shutdown
halt:*:7:0:halt:/sbin:/sbin/halt
nobody:*:65534:65534:Nobody:/:/bin/false
col:Wh0yzfAV2qm2Y:100:100:Caldera OpenLinux
    User:/home/col:/bin/bash
```

Many lines from **passwd.test** are indeed produced that contain the string *bin*; however, we have to search through these lines in order to find the user *bin,* which is the line in which *bin* is the first string that appears. This is more than we wanted when we initiated our search. We wanted to see a user name *bin* that would appear at the beginning of a line. We can further qualify our search, in this case to limit the search to a string at the beginning of a line, by using pattern matching discussed at the beginning of this chapter (see the Table 24-1). In this case, we want to search only at the beginning of a line for *bin,* so we'll qualify our search with a caret (^) to restrict the search to only the beginning of the line, as shown in the following example:

```
# grep ^bin passwd.test
bin:*:1:1:bin:/bin:
```



This search results in exactly the information in which we are interested, that is, a line beginning with *bin*. When using special characters, such as the caret(^) in this example, you should enclose the special characters in single quotes ('). Special characters may be interpreted by the shell and cause problems with the arguments we're trying to send to **grep**. Enclosing the search pattern in single quotes will ensure that the search pattern, in this case ^*bin,* is passed directly to **grep**. The search pattern in single quotes looks like the following:

```
# grep '^bin' passwd.test
bin:*:1:1:bin:/bin:
```

Because we are going to have to search for this line in the **passwd.test** file after we find it, we may as well print out the line number as well as the line itself by using the *-n* option, as shown in the following example:

```
# grep -n "^bin" passwd.test
2:bin:*:1:1:bin:/bin:
```

The following is a summary of the **grep** command:

grep - Search for text and display results.

Options

-c Return the number of matches without showing you the text.

-h Show the text with no reference to file names.

-i Ignore the case when searching.

-l Return the names of files containing a match without showing you the text.

-n Return the line number of the text searched for in a file as well as the text itself.

-v Return the lines that do not match the text you searched for.

-E Search for more than one expression (same as **egrep**).

-F Search for more than one expression (same as **fgrep**).

Manual Pages for Commands Used in Chapter 24

There are manual pages in Appendix A for the following commands used in this chapter:

```
awk
grep
sed
```

Chapter 25. find Command

find Overview

Finding Files of a Specific Type

Find Empty Files and Directories

Finding Files By Name, Size, and Both Name and Size

Finding Files By Owner, Type, and Permissions

Finding Long Unaccessed Files and Running Commands on Found Files

Find Summary

Manual Pages for Commands Used in Chapter 25

find Overview



The **find** command is used to locate files by traversing the UNIX tree structure. You can start from any point on the system, even the root level, and traverse through the entire hierarchy. After finding files, you can also perform actions on them.

The general format of the **find** command is as follows:

```
find path operators
```

path is the directory in which **find** will begin a search. *operators* are one or more of the many **find** options you specify. The end of this section contains a list of commonly used operators to **find**. We'll work with several of the most commonly used operators in some of the upcoming examples.

The most common result of **find** is to produce a list of files. You can produce a list of files in the current working directory, specified by a dot (.), and *print* those files, as shown in the following example:




```
# cd /home
#
# ls -l
total 3
drwxr-xr-x   3 col      users        1024 Nov  8 14:09 col
drwxr-xr-x   6 root     root         1024 Nov  8 14:08 ftp
drwxr-xr-x   6 root     root         1024 Nov  8 14:08 httpd
#
# find . -print
.
./httpd
./httpd/apache
./httpd/apache/doc
./httpd/apache/doc/manual.ps.gz
./httpd/cgi-bin
./httpd/cgi-bin/HelpIndex
./httpd/cgi-bin/HelpScreen
./httpd/html
./httpd/html/dt
./httpd/html/dt/dt.html
./httpd/html/dt/dt.html.idx
./httpd/html/dt/dt.index
./httpd/html/dt/expert.gif
./httpd/html/dt/hint.gif
./httpd/html/dt/index.gif
./httpd/html/dt/info2.gif
./httpd/html/dt/note.gif
./httpd/html/dt/sysadm.gif
./httpd/html/dt/up.gif
./httpd/html/dt/warning.gif
./httpd/icons
./ftp
./ftp/bin
./ftp/bin/gzip
./ftp/bin/ls
./ftp/bin/tar
./ftp/bin/zcat
./ftp/etc
./ftp/etc/group
./ftp/etc/passwd
./ftp/lib
```

```
./ftp/pub
./col
./col/.bashrc
./col/.cshrc
./col/.login
./col/.profile
./col/lg
./col/lg/lg_layouts
./col/lg/lg_layouts/User
./col/lg/lg3_prefs
./col/lg/lg3_soundPref
./col/lg/lg3_startup
```

This **find** operation was performed from the **/home** directory. Notice that there are only three home directories under **/home**, and the **find** command traverses the hierarchy for each of the three home directories. Keep in mind that you probably don't want to perform this **find** operation at the root level. You will traverse the entire hierarchy and get a list of every file on the system.



When using **find**, you may discover that you receive a message like the following when a file or directory is encountered for which you do not have adequate permission to traverse:

```
find: /var/spool/cron: Permission denied
```

This is not an ucommon message when running **find**, so don't panic. Users working outside their home directory often encounter this message when running a variety of commands, including **find**.

A typical **find** command will specify the path in which to search for a specific file. In this case, the *expression* is the *name* of the file for which you wish to search, as shown in the following example:

```
# find /home -name ftp
/home/ftp
```

In this example we search the *path* **/home** looking for the *name* **ftp**.

[Team LiB]

PREVIOUS NEXT

Finding Files of a Specific Type

You may want to perform a **find** operation to produce a list of files only and not include directories in the operation. The following **find** is similar to what we performed earlier, but this time it produces a list of files only. This is achieved by specifying that we are looking for type *f* for files:

```
# find /home -type f -print
/home/httpd/apache/doc/manual.ps.gz
/home/httpd/cgi-bin/HelpIndex
/home/httpd/cgi-bin/HelpScreen
/home/httpd/html/dt/dt.html
/home/httpd/html/dt/dt.html.idx
/home/httpd/html/dt/dt.index
/home/httpd/html/dt/expert.gif
/home/httpd/html/dt/hint.gif
/home/httpd/html/dt/index.gif
/home/httpd/html/dt/info2.gif
/home/httpd/html/dt/note.gif
/home/httpd/html/dt/sysadm.gif
/home/httpd/html/dt/up.gif
/home/httpd/html/dt/warning.gif
/home/ftp/bin/gzip
/home/ftp/bin/ls
/home/ftp/bin/tar
/home/ftp/etc/group
/home/ftp/etc/passwd
/home/col/.bashrc
/home/col/.cshrc
/home/col/.login
/home/col/.profile
/home/col/lg/lg_layouts/User
/home/col/lg/lg3_prefs
/home/col/lg/lg3_soundPref
/home/col/lg/lg3_startup
```

You can search for a variety of different *types* such as *f* for a file as shown in the example, *b* for a block special file, *l* for a symbolic link, and so on.

Find Empty Files and Directories



A useful test when performing a **find** operation is to locate empty files. The following example searches for all empty files and directories on the system with the *-empty* operator that is available on some UNIX variants, including Linux. This is only a partial output because it's so long:

```
# find / -empty -print
/lost+found
/var/adm/LST/analyse
/var/spool/lpd
/var/spool/news
/var/spool/uucp
/var/spool/mqueue
/var/spool/atjobs/.SEQ
/var/spool/atspool
/var/spool/cron
/var/spool/fax/outgoing/locks
/var/spool/fax/incoming
/var/spool/voice/incoming
/var/spool/voice/messages
/var/spool/rwho
/var/spool/uucppublic
/var/lib/LST/log
/var/lib/LST/analyse
/var/lib/LST/disks
/var/lib/LST/catalog
/var/lib/LST/conflicts
/var/lib/LST/saved
/var/lib/LST/replaced
/var/lib/LST/deleted
/var/lib/games
/var/local
/var/lock/subsys/inet
/var/lock/subsys/ipx
/var/lock/subsys/syslog
/var/lock/subsys/amd
/var/lock/subsys/cron
/var/lock/subsys/atd
/var/lock/subsys/mta
/var/lock/subsys/rstatd
/var/lock/subsys/httpd
/var/log/httpd/apache/access_log
/var/log/xferlog
/var/log/uucp
/var/log/secure
/var/log/spooler
/var/named
/var/nis
/var/preserve
/var/run/xlaunch
/var/tmp
                  -
                  -
                  -
/proc/net/snmp
/proc/net/raw
/proc/net/igmp
/proc/net/arp
/proc/net/unix
/proc/cpuinfo
/proc/pci
/proc/version
/proc/kmsg
/proc/meminfo
/proc/uptime
```

/proc/loadavg
/proc/mdstat
/etc/modules/options
/etc/motd
/etc/exports
/etc/ppp/options
/tmp/LST
/tmp/.XF86Setup235/f51cb27-315ae55/ServerOut-2
/tmp/.XF86Setup246/13617839-11c71c71/ServerOut-2
/tmp/.XF86Setup262/1f2cc0a9-aea314d/ServerOut-2
/tmp/.XF86Setup288/17963ff7-21ca966f/ServerOut-2
/tmp/fsslog
/mnt/floppy
/mnt/cdrom
/usr/doc/html/woven/LDP/install-guide-2.2.2.html/images.idx
/usr/lib/games
/usr/lib/groff/tmac/mm/locale
/usr/lib/groff/tmac/mm/se_locale
/usr/lib/kbd/keytables/patch_tables.orig
/usr/lib/perl5/i386-linux/5.003/auto/GDBM_File/GDBM_File.bs
/usr/lib/perl5/i386-linux/5.003/auto/DB_File/DB_File.bs
/usr/lib/perl5/i386-linux/5.003/auto/Fcntl/Fcntl.bs
/usr/lib/perl5/i386-linux/5.003/auto/FileHandle/FileHandle.bs
/usr/lib/perl5/i386-linux/5.003/auto/NDBM_File/NDBM_File.bs
/usr/lib/perl5/i386-linux/5.003/auto/POSIX/POSIX.bs
/usr/lib/perl5/i386-linux/5.003/auto/SDBM_File/SDBM_File.bs
/usr/lib/perl5/i386-linux/5.003/auto/Safe/Safe.bs
/usr/lib/perl5/i386-linux/5.003/auto/Socket/Socket.bs
/usr/lib/perl5/i386-linux/5.003/auto/Text/ParseWords
/usr/lib/perl5/site_perl/i386-linux
/usr/lib/linuxdoc-sgml/null.sty
/usr/etc
/usr/include/g++
/usr/include/netax25
/usr/include/netipx
/usr/include/readline
/usr/local/bin
/usr/local/doc
/usr/local/etc
/usr/local/games
/usr/local/info
/usr/local/lib
/usr/local/man/man1
/usr/local/man/man2
/usr/local/man/man3
/usr/local/man/man4
/usr/local/man/man5
/usr/local/man/man6
/usr/local/man/man7
/usr/local/man/man8
/usr/local/man/man9
/usr/local/man/mann
/usr/local/sbin
/usr/local/src
/usr/man/man9
/usr/man/mann
/usr/src/linux-2.0.29/include/linux/modules
/usr/src/linux-2.0.29/modules
/usr/src/redhat/BUILD
/usr/src/redhat/RPMS/i386
/usr/src/redhat/SOURCES
/usr/src/redhat/SPECS
/usr/src/redhat/SRPMS
/usr/X11R6/doc
/usr/X11R6/lib/X11/x11perfcomp
/usr/visix/fss/En_US.8859
/usr/visix/vls/En_US.8859
/usr/openwin/share/src/xview
/auto
/initrd
/home/httpd/icons
/home/ftp/lib
/home/ftp/pub
/opt/bin
/opt/man

```
/root/lg/lg3_hosts
/amd/nycald1/auto/.vTRASH
#
# ll /auto
total 0
```



All the files and directories listed as a result of this **find** operation are empty. The long listing of **/auto** shown as part of the example, confirms this fact. Keep in mind that *-empty* is not available on all UNIX variants.

[Team LiB]

PREVIOUS NEXT

Finding Files By Name, Size, and Both Name and Size

Let's perform a couple of independent finds and then combine the criteria of the finds. The finds in the following example are performed on a small desktop system. This allows me to search the entire system for files meeting various criteria. You may be working on a much larger, more elaborate system, so use caution when searching the entire system for files. You may slow down other users for a long period of time, depending on the number of files on the system both locally and accessible over the network.

First, let's find for all files on the system ending in **.c** with the command below:

```
# find / -name *.c -print
/usr/X11R6/include/X11/Xaw/Template.c
/usr/X11R6/include/Xaw3d/Template.c
/usr/X11R6/lib/X11/etc/et4000clock.c
/usr/doc/glibc-2.1.1/examples.threads/ex1.c
/usr/doc/glibc-2.1.1/examples.threads/ex2.c
/usr/doc/glibc-2.1.1/examples.threads/ex3.c
/usr/doc/glibc-2.1.1/examples.threads/ex4.c
/usr/doc/glibc-2.1.1/examples.threads/ex5.c
/usr/doc/glibc-2.1.1/examples.threads/ex6.c

             .
             .
             .

/usr/doc/bind-8.2/notes/db_names.c
/usr/doc/libpng-1.0.3/example.c
/usr/doc/FAQ/html/clone.c
/usr/doc/gnome-libs-devel/devel-docs/gnomeui/gnomeui-scan.c
#
```

You can see from this find that there are many files on the system ending in **.c**. I included only the beginning and end of this search because the entire output would be too long.

I also ran this command and piped the output to **wc**, which showed 1737 files on the system ending in **.c**, as shown in the following example:

```
# find / -name *.c | wc
   1737    1737   77044
#
```



Now we can search for all files on the system greater than *500,000* characters in size with the **find** command below:

```
# find / -size +500000c -print
/dev/core
/etc/X11/core
/var/lib/rpm/packages.rpm
/var/lib/rpm/fileindex.rpm
/var/lib/dosemu/hdimage.freedos
/var/lib/slocate/slocate.db
/var/lock/samba/SHARE_MEM_FILE
/proc/kcore
/home/ftp/lib/libc-2.1.1.so
/home/test/linux1.xwd
/lib/libc-2.1.1.so
/lib/libdb-2.1.1.so
/lib/libm-2.1.1.so
/root/core

             .
             .
             .
```

```
/usr/share/gnome/help/users-guide/C/figs/full.gif
/usr/share/emacs/20.3/etc/DOC-20.3.1
/usr/share/emacs/20.3/leim/quail/ZIRANMA.el
/usr/share/emacs/20.3/leim/quail/ZIRANMA.elc
/usr/share/emacs/20.3/leim/skk/skkdic.el
/usr/share/emacs/20.3/leim/skk/skkdic.elc
/usr/share/emacs/20.3/lisp/loaddefs.el
/usr/share/sounds/card_shuffle.wav
/usr/share/sounds/startup2.wav
/usr/share/guavac/classes.zip
```

I only included the beginning and end of this listing as well since there were many files of size greater than *500,000* characters on the system.

I also ran this command and piped the output to **wc**, which showed *215* files on the system with a size greater than *500,000* characters, as shown in the following example:

```
# find / -size +500000c -print | wc
    215     215    6281
#
```

Let's now combine the two commands and see how many of the files on the system ending in **.c** are also greater than *500,000* characters in size:

```
# find / -name *.c -size +500000c -print
/usr/src/drivers/scsi/advansys.c
#
```



Of the 1737 files on the system ending in **.c** and the 215 files greater than *500,000* characters in size, only one file, **advansys.c**, meets both criteria. There is an implied *and* in the previous **find** command. We could have explicitly specified an *and*, however, it is implied in the **find** command. The **find** did indeed result in files that end in **.c** *and* are greater than 500,000 characters. The upcoming **find** uses the *or* operator, which is not implied.

What if we wanted to find both files ending in **.c** and **.o** that are greater than *500000* characters in size? We could use the *-o* operator which would "or" the files ending in **.c** and **.o**. The following example shows this **find** operation:

```
find / -size +500000c \(  -name *.c -o -name *.a \) -print
find / -size +500000c \(  -name *.c -o -name *.a \) -print | wc
```

```
# find / -size +500000c \( -name *.c -o -name *.a \) -print
/usr/X11R6/lib/libMagick.a
/usr/X11R6/lib/libX11.a
/usr/lib/bind/lib/libbind.a
/usr/lib/bind/lib/libbind_r.a
/usr/lib/libbfd.a
/usr/lib/libBLT.a
/usr/lib/libstdc++-2-libc6.1-1-2.9.0.a
/usr/lib/libc.a

             .
             .
             .

/usr/lib/libinn.a
/usr/lib/libtiff.a
/usr/lib/liblinuxconf.a
/usr/lib/libform_g.a
/usr/lib/libncurses_g.a
/usr/lib/python1.5/config/libpython1.5.a
/usr/lib/libpisock.a
/usr/lib/libcustoms.a
/usr/lib/libqt.a
/usr/lib/libst.a
/usr/src/drivers/scsi/advansys.c
#
```

The two file extensions for which we are searching are placed in parentheses. A backslash is required before the open and close parentheses because the



parentheses have meaning to the shell and we want them to be used only to specify the precedence in our **find** command and not to be used by the shell. The result of this **find** shows that many files ending in **.a** and **.c** meet the criteria of greater than *500,000* characters.

Let's now pipe the output of this commnd to **wc** to see precisely the number of files ending in either **.c** *or* **.o** *and* having a size greater than *500,000* characters:

```
# find / -size +500000c \( -name *.c -o -name *.a \) -print | wc
     39      39     982
#
```

Of the *39* files that meet the criteria, we know that *38* ended in **.a** because our earlier example showed that only one file ending in **.c** met the criteria.

[Team LiB]

PREVIOUS NEXT

Finding Files By Owner, Type, and Permissions



You can find objects on the system owned by particular users and groups. To find all of the objects owned by user "news" on the system, we would use the following command:

```
# find / -user news -print
/etc/rc.d/rc.news
/etc/news
/etc/news/cleanfeed.conf
/etc/news/actsync.cfg
/etc/news/actsync.ign
/etc/news/control.ctl
/etc/news/cycbuff.conf
/etc/news/distrib.pats
/etc/news/expire.ctl
/etc/news/incoming.conf
/etc/news/inn.conf
/etc/news/innfeed.conf
/etc/news/innreport.conf
/etc/news/innwatch.ctl
/etc/news/moderators
/etc/news/motd.news
/etc/news/news2mail.cf
/etc/news/newsfeeds
/etc/news/nnrp.access
/etc/news/nnrpd.track
/etc/news/nntpsend.ctl
/etc/news/overview.ctl
/etc/news/overview.fmt
/etc/news/passwd.nntp
/etc/news/storage.conf

             .
             .
             .

/usr/man/man8/tally.control.8
/usr/man/man8/tally.unwanted.8
/usr/man/man8/writelog.8
```



Using the *-user* operator we can specify either the name of the user, in this case "news", or the user identification number. The following example shows performing the same **find** operation using the user identification number of "news," in this case "9," instead of the name "news":

```
# find / -user 9 -print
/etc/rc.d/rc.news
/etc/news
/etc/news/cleanfeed.conf
/etc/news/actsync.cfg
/etc/news/actsync.ign
/etc/news/control.ctl
/etc/news/cycbuff.conf
/etc/news/distrib.pats
/etc/news/expire.ctl
/etc/news/incoming.conf
/etc/news/inn.conf
/etc/news/innfeed.conf
/etc/news/innreport.conf
/etc/news/innwatch.ctl
/etc/news/moderators
```

```
/etc/news/motd.news
/etc/news/news2mail.cf
/etc/news/newsfeeds
/etc/news/nnrp.access
/etc/news/nnrpd.track
/etc/news/nntpsend.ctl
/etc/news/overview.ctl
/etc/news/overview.fmt
/etc/news/passwd.nntp
/etc/news/storage.conf

             .
             .
             .

/usr/man/man8/tally.control.8
/usr/man/man8/tally.unwanted.8
/usr/man/man8/writelog.8
```

This **find** operation produced exactly the same results using the name "news" and the user idenfication number "9."

You can search for a variety of different *types* such as *f* for a file, as shown in the example, *b* for a block special file, *l* for a symbolic link, and so on. We could add *type -d* to find only directories belonging to "news," as in the following command:

```
# find / -user news -type d -perm 775 -print
/etc/news
/var/lib/news
/var/log/news
/var/log/news/OLD
/var/spool/news
/var/spool/news/archive
/var/spool/news/articles
/var/spool/news/incoming
/var/spool/news/incoming/bad
/var/spool/news/innfeed
/var/spool/news/outgoing
/var/spool/news/overview
/var/spool/news/uniover
/var/spool/slrnpull
/usr/bin/auth
/usr/bin/control
/usr/bin/filter
/usr/bin/rnews.libexec
```



This is another example of using the implied *and* of **find** meaning that the **find** will print items that are both owned by "news" *and* are directories only.

Let's now add a specific permission for the directories to our implied *and*. We'll **find** only objects belonging to "news" that are directories with a permission of 775 in the following example:

```
# find / -user news -type d -perm 775 -print
/etc/news
/var/lib/news
/var/log/news
/var/log/news/OLD
/var/spool/news
/var/spool/news/archive
/var/spool/news/articles
/var/spool/news/incoming
/var/spool/news/incoming/bad
/var/spool/news/innfeed
/var/spool/news/outgoing
/var/spool/news/overview
/var/spool/news/uniover
/var/spool/slrnpull
/usr/bin/auth
/usr/bin/control
```

```
/usr/bin/filter
/usr/bin/rnews.libexec
```



We searched for directories belonging to "news" in which both the owner and those in the group have read-write-execute permission, and others having read-execute access. This is a common type of **find** operation for system administrators to perform - looking for files and directories belonging to a specific user and having specific permissions.

[Team LiB]

PREVIOUS NEXT

Finding Long Unaccessed Files and Running Commands on Found Files

We all have old files on our systems and in our home directories that have not been accessed in a long time. To find files in a home directory that have not been accessed in the last 200 days, you would issue the following command:

```
# find . -atime +200 -print
./hp/.Xdefaults
./hp/.bash_logout
./hp/.bash_profile
./hp/.bashrc
./hp/.kde/share/apps/kfm/desktop
./hp/.kde/share/config/desktop0rc
./hp/.kde/share/config/kcmdisplayrc
./hp/.kde/share/config/kfmrc
./hp/.kde/share/config/kpanelrc
./hp/.kde/share/config/kvtrc
./hp/.kderc
./hp/Desktop/.directory
./hp/Desktop/Printer.kdelnk
./hp/Desktop/Templates/Device.kdelnk
./hp/Desktop/Templates/Ftpurl.kdelnk
./hp/Desktop/Templates/MimeType.kdelnk
./hp/Desktop/Templates/Program.kdelnk
./hp/Desktop/Templates/URL.kdelnk
./hp/Desktop/Templates/WWWUrl.kdelnk
./hp/Desktop/cdrom.kdelnk
./hp/Desktop/floppy.kdelnk
./hp/.screenrc
./hp/.bash_history
./test/.Xdefaults
./test/.bash_logout
./test/.bash_profile
./test/.bashrc
./test/.kde/share/apps/kfm/desktop
./test/.kde/share/config/desktop0rc
./test/.kde/share/config/kcmdisplayrc
./test/.kde/share/config/kfmrc
./test/.kde/share/config/kpanelrc
./test/.kde/share/config/kvtrc
./test/.kderc
./test/Desktop/.directory
./test/Desktop/Printer.kdelnk
./test/Desktop/Templates/Device.kdelnk
./test/Desktop/Templates/Ftpurl.kdelnk
./test/Desktop/Templates/MimeType.kdelnk
./test/Desktop/Templates/Program.kdelnk
./test/Desktop/Templates/URL.kdelnk
./test/Desktop/Templates/WWWUrl.kdelnk
./test/Desktop/cdrom.kdelnk
./test/Desktop/floppy.kdelnk
./test/.screenrc
./test/linux1.xwd
./test/.bash_history
#
```

Several files were produced as a result of this search. I could run the same command and add to it an **ls -l** command so I could look at additional information on the file with the following command:

```
# find . -atime +200 -exec ls -l {} \;

-rw-r--r--   1 hp       hp        1422 Aug 28   ./hp/.Xdefaults
-rw-r--r--   1 hp       hp          24 Aug 28   ./hp/.bash_logout
-rw-r--r--   1 hp       hp         230 Aug 28   ./hp/.bash_profile
-rw-r--r--   1 hp       hp         124 Aug 28   ./hp/.bashrc
-rw-r--r--   1 hp       hp         260 Aug 28   ./hp/.kde/share/apps/kfm/desktop
-rw-r--r--   1 hp       hp         234 Aug 28   ./hp/.kde/share/config/desktop0rc
```

```
-rw-r--r--   1 hp       hp          49 Aug 28   ./hp/.kde/share/config/kcmdisplayrc
-rw-r--r--   1 hp       hp          95 Aug 28   ./hp/.kde/share/config/kfmrc
-rw-r--r--   1 hp       hp         456 Aug 28   ./hp/.kde/share/config/kpanelrc
-rw-r--r--   1 hp       hp         255 Aug 28   ./hp/.kde/share/config/kvtrc
-rw-r--r--   1 hp       hp         966 Aug 28   ./hp/.kderc
-rw-r--r--   1 hp       hp          85 Aug 28   ./hp/Desktop/.directory
-rw-r--r--   1 hp       hp         222 Aug 28   ./hp/Desktop/Printer.kdelnk
-rw-r--r--   1 hp       hp         607 Aug 28   ./hp/Desktop/Templates/Device.kdelnk
-rw-r--r--   1 hp       hp         296 Aug 28   ./hp/Desktop/Templates/Ftpurl.kdelnk
-rw-r--r--   1 hp       hp         324 Aug 28   ./hp/Desktop/Templates/MimeType.kdelnk
-rw-r--r--   1 hp       hp         170 Aug 28   ./hp/Desktop/Templates/Program.kdelnk
-rw-r--r--   1 hp       hp         411 Aug 28   ./hp/Desktop/Templates/URL.kdelnk
-rw-r--r--   1 hp       hp         458 Aug 28   ./hp/Desktop/Templates/WWWUrl.kdelnk
-rw-r--r--   1 hp       hp         376 Aug 28   ./hp/Desktop/cdrom.kdelnk
-rw-r--r--   1 hp       hp         378 Aug 28   ./hp/Desktop/floppy.kdelnk
-rw-rw-r--   1 hp       hp        3505 Aug 28   ./hp/.screenrc
-rw-------   1 hp       hp          34 Aug 28   ./hp/.bash_history
-rw-r--r--   1 test     test      1422 Aug 28   ./test/.Xdefaults
-rw-r--r--   1 test     test        24 Aug 28   ./test/.bash_logout
-rw-r--r--   1 test     test       230 Aug 28   ./test/.bash_profile
-rw-r--r--   1 test     test       124 Aug 28   ./test/.bashrc
-rw-r--r--   1 test     test       260 Aug 28   ./test/.kde/share/apps/kfm/desktop
-rw-r--r--   1 test     test       234 Aug 28   ./test/.kde/share/config/desktop0rc
-rw-r--r--   1 test     test        49 Aug 28   ./test/.kde/share/config/kcmdisplayrc
-rw-r--r--   1 test     test        95 Aug 28   ./test/.kde/share/config/kfmrc
-rw-r--r--   1 test     test       456 Aug 28   ./test/.kde/share/config/kpanelrc
-rw-r--r--   1 test     test       255 Aug 28   ./test/.kde/share/config/kvtrc
-rw-r--r--   1 test     test       966 Aug 28   ./test/.kderc
-rw-r--r--   1 test     test        85 Aug 28   ./test/Desktop/.directory
-rw-r--r--   1 test     test       222 Aug 28   ./test/Desktop/Printer.kdelnk
-rw-r--r--   1 test     test       607 Aug 28   ./test/Desktop/Templates/Device.kdelnk
-rw-r--r--   1 test     test       296 Aug 28   ./test/Desktop/Templates/Ftpurl.kdelnk
-rw-r--r--   1 test    test        324 Aug 28  ./test/Desktop/Templates/MimeType.kdelnk
-rw-r--r--   1 test     test       170 Aug 28   ./test/Desktop/Templates/Program.kdelnk
-rw-r--r--   1 test     test       411 Aug 28   ./test/Desktop/Templates/URL.kdelnk
-rw-r--r--   1 test     test       458 Aug 28   ./test/Desktop/Templates/WWWUrl.kdelnk
-rw-r--r--   1 test     test       376 Aug 28   ./test/Desktop/cdrom.kdelnk
-rw-r--r--   1 test     test       378 Aug 28   ./test/Desktop/floppy.kdelnk
-rw-rw-r--   1 test     test      3505 Aug 28   ./test/.screenrc
-rwxrwxrwx   1 root     root    614955 Aug 28   ./test/linux1.xwd
-rw-------   1 test     test        13 Aug 28   ./test/.bash_history
#
```



This command produces the same output as the previous example and puts the output of **find** in the place where the curly brackets appear. The **ls -l** will use the filename produced by **find**, which is inside the curly brackets, as an argument. You follow the command with a semicolon so that the shell knows where the command ends, and you must use the backslash as an escape for the semicolon so that the **find** command recognizes the semicolon.

Find Summary

This chapter contained many examples of common **find** operations. There are many addtional **find** operators that were not covered. The following is a summary of some of the more commonly used operators of **find**. There is also a manual page in Appendix A describing **find** in more detail. Depending on the UNIX variant you are using, the *-print* option may or may not be needed.

find - 

Search for files.

Commonly used operators or options:

-atime *n*

Find files that were accessed *n* days ago. +*n* finds files accessed greater than *n* days ago and -*n* will find files accessed less than *n* days ago.

-ctime *n*

Find files in which the inode was modified *n* days ago. +*n* finds files in which the inode was modified greater than *n* days ago and -*n* will find files in which the inode modified less than *n* days ago.

-exec *command*

Execute *command.*

-group *name*

Find files belonging to the given group *name* where *name* can also be the group ID number.

-mount

Do not descend directories on other file systems (not available on all UNIX variants.)

-mtime *n*

Find files that were modified *n* days ago. +*n* finds files modified greater than *n* days ago and -*n* will find files modified less than *n* days ago.

-newer *file*

File was modified more recently than *file*.

-name *pattern*

Look for file name of *pattern*.

-ok *command*

Check to see that it is okay to execute *command* before doing so. This is similar to -exec except that you are prompted for permission.

-perm *mode*

Find files with the specified access *mode*. You would supply the access *mode* in octal.

-print

Print the current file name to standard output.

-type t

File has a type of *t,* such as *d* for directory and *f* for file.

-size *n*

Find files with a size of *n* blocks. A block is usually 512 bytes. Using +*n* will find files greater than *n* blocks, *nc* will find files *n* characters in size and +*nc* will find files greater than *n* characters in size.

-user *uname*

File is owned by *uname.*

Multiple Criteria:

-a to and two operators (operator1 -a operator2)

-o to or two operators (operator1 -o operator2)

! to specify that the operator not be matched (!operator)

\(expression)\ to specify that this expression be evaluated before others to specify preference.

Manual Pages for Commands Used in Chapter 25

There are manual pages in Appendix A for the following command used in this chapter:

find

Chapter 26. The vi Editor

The vi Editor

Regular Expression Words-of-Caution

Expressions Are Strings and Wildcards

Manual Pages for Commands Used in Chapter 26

The vi Editor

In this chapter, we'll cover the following topics:

- Regular expressions
- **vi**

Many UNIX users have a Graphical User Interface (GUI) through which they access their UNIX system. The Common Desktop Environment (CDE) is the most commonly used GUI on UNIX systems. It is based on the X Windows System and Motif, which together provide an advanced windowing environment. Chapter 14 in this book is devoted to CDE. Most UNIX GUIs provide a graphical editor. Despite the fact that these graphical editors are a standard part of most GUIs, the visual editor, **vi**, remains the most popular UNIX editor. With many fine graphics-based editors as a standard part of most UNIX GUIs and a plethora of editors available as part of personal computer windowing environments, why am I covering **vi**? The answer is two-fold. First, not everyone using a UNIX system has access to a graphics display and may therefore need to know and use **vi**. Because **vi** comes with most UNIX-based systems and is a powerful editor, many new UNIX users end up using and liking it. Second, **vi** has traditionally been thought of as *the* UNIX editor. Few UNIX users have not used **vi**. This fact does not mean that it is everyone's primary editor; however, virtually all UNIX users have had some experience with **vi**.

Also, a line editor called **ed** comes with many UNIX systems. It is now seldom used because **vi** is a screen editor. Also available is an enhanced version of **ed** called **ex. vi** is much more widely used than either of the line editors, so I'll cover only **vi** in this chapter.

I'll cover the *basics* of using **vi** in this chapter. You can experiment with what is covered here, and if you really like it, you can investigate some of the more advanced features of **vi**. A quick reference card summarizing all the **vi** commands covered in this chapter is included with this book.

The following table is a list of tables in this chapter that summarize some of the more commonly used **vi** commands by function:

Table Number	vi Function
Expr	Regular Expressions
Introduction	Modes and Notations in **vi**
1	Starting a **vi** Session
2	Cursor Control Commands in **vi**
3	Adding Text in **vi**
4	Deleting Text in **vi**
5	Changing Text in **vi**
6	Search and Replace in **vi**
7	Copying in **vi**
8	Undo in **vi**
9	Saving Text and Exiting **vi**
10	Options in **vi**
11	Status in **vi**
12	Positioning and Marking in **vi**
13	Joining Lines in **vi**
14	Cursor Placement and Adjusting Screen in **vi**

15	Shell Escape Commands in **vi**
16	Macros and Abbreviations in **vi**
17	Indenting Text in **vi**
18	Shell Filters in **vi**
19	Pattern Matching in **vi**

[Team LiB]

PREVIOUS NEXT



Regular Expression Words-of-Caution

Regular expressions describe patterns for which you are searching. The regular expression usually defines the pattern for which you are searching using wildcards. Since a regular expression defines a pattern you are searching for, the terms *regular expression* and *pattern matching* are often used interchangably.

Let's get down to a couple of words-of-caution immediately:

- **Regular expressions are different from file-matching patterns used by the shell**. Regular expressions are used by both the shell and many programs, including those covered in this chapter. The file matching done by the shell and programs such as **find** are different from the regular expressions covered in this chapter.

- **Use single quotes around regular expressions**. The meta-characters used in this chapter must be quoted in order to be passed to the shell as an argument. You will, therefore, see most regular expressions in this chapter quoted.



Expressions Are Strings and Wildcards



When using the programs in this book, such as **grep** and **vi**, you provide a regular expression that the program evaluates. The command will search for the pattern you supply. The pattern could be as simple as a string or it could include wildcards. The wildcards used by many programs are called meta-characters.



Table 26-Expr shows a list of meta-characters and the program(s) to which they apply. Only the programs covered in this book (**awk, grep, sed**, and **vi**) are shown in Table 26-Expr. These meta-characters may be used with other programs, such as **ed** and **egrep**, as well, which are not covered in the book. Table 26-Expr describes the meta-characters and their use.

Table 26-Expr. Meta-Characters and Programs to Which They Apply

Meta-Character	awk	grep	sed	vi	Use
.	Yes	Yes	Yes	Yes	Match any single character.
*	Yes	Yes	Yes	Yes	Match any number of the single character that precedes *.
[...]	Yes	Yes	Yes	Yes	Match any *one* of the characters in the set [...].
$	Yes	Yes	Yes	Yes	Matches the end of the line.
^	Yes	Yes	Yes	Yes	Matches the beginning of the line.
\	Yes	Yes	Yes	Yes	Escape the special character that follows \.
\{*n,m*\}	Yes	Yes	No	No	Match a range of occurrences of a single character between *n* and *m*.
+	Yes	No	No	No	Match one or more occurrences of the preceding regular expression.
?	Yes	No	No	No	Match zero or one occurrence of the preceding regular expression.
|	Yes	No	No	No	The preceding or following regular expression can be matched.
()	Yes	No	No	No	Groups regular expressions in a typical parenthesis fashion.
\{ \}	No	No	No	Yes	Match a word's beginning or end.

You may want to refer to this table when regular expressions are used for one of the commands in the table.

Modes and Notations

We're first going to cover some of the fundamentals of the operation of **vi** called modes, and then go over some of the notations used in the tables in this chapter.

A feature of **vi** that often confuses new users is that it has modes. When you are in *command mode*, everything you type is interpreted as a command. In *command mode,* you can specify such actions as the location to which you want the cursor to move. When you are in *input mode,* everything you type is information to be added to the file. *Command mode* is the default when you start **vi**. You can move into *command mode* from *input mode* at any time by pressing the *escape* key. You move into *insert mode* from *command mode* by typing one of the *input mode* commands covered shortly.

vi commands don't really have a standard form. For this reason, I cover common notations. Table 26-Introduction summarizes modes and commands in **vi**:

Table 26-Introduction. Modes and Notations in vi

Mode or Notation	Description
Command Mode	You are issuing commands such as moving the cursor or deleting text, rather than inserting or changing text when in command mode. You can switch to insert mode by issuing an insert mode command such as **i** for insert or **a** for add text.
Insert Mode	You are in insert mode when changing or inserting more than one character of text. You can switch to command mode by pressing the *escape* key.
: (colon commands)	Commands that start with a **:** are completed by pressing the *return* key.
control (**^**) commands	When a command uses the *control* (**^**) key, you press and hold down the *control* key and then press the next key that is part of the command. For instance, **^g** means press and hold *control* and then **g** to get the status on the file you are editing.
file for the name of a file	Many commands require you to specify the name of a file. For instance, in the command **vi** *file,* you would substitute the name of the file you wish to edit for *file.*
char for the name of a character	Many commands require you to specify a single character. For instance, in the command **f***char,* you would substitute the character you wish to search for in place of *char*.
cursor_command for a cursor movement command	Many commands require you to specify a cursor command to execute. For instance, in the command **d***cursor_command,* you would substitute for *cursor_command* the command you wish to execute.
string for a character string	Many commands require you to specify a character string. For instance, in the command **/***string,* you would substitute for *string* the character string for which you wish to search.

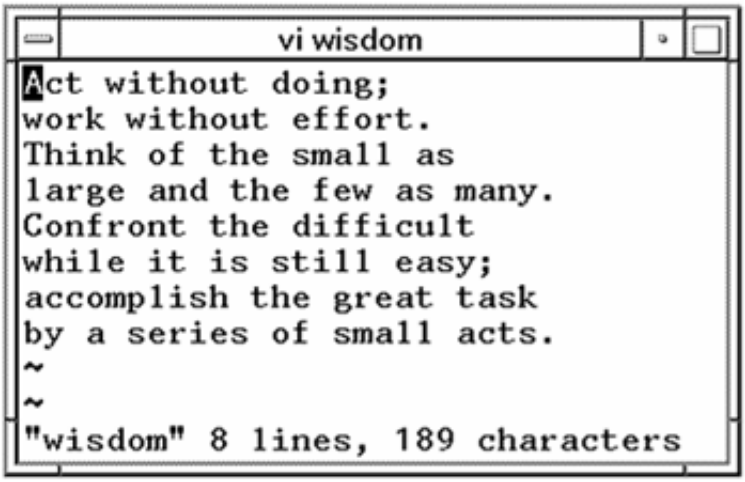
Starting a vi Session

Let's jump right in and edit a file. For most of the examples in this chapter, I perform various **vi** commands and capture the results in an X Window. The best way to learn any topic is by example. I not only provide many examples, but I also capture each example in an X Window so that you can see the results of each command. From the command line, we type **vi** and the name of the file we wish to edit, in this case **wisdom**:

```
$ vi wisdom
```

We are then editing the file **wisdom** as shown in Figure 26-1. **wisdom** contains a passage from *Tao Te Ching* or "Book of the Way." We use this file throughout this chapter.

Figure 26-1. Editing the File wisdom



The bottom line in Figure 26-1 is the message line in **vi**. After invoking **vi,** the message line indicates the name of the file, the number of lines, and the number of characters in the file. Different messages appear on the message line, depending on the command you issue, as we see in upcoming examples. If a tilde appears on any lines in the file, as it does in the two lines above the message line in **wisdom**, it means that not enough lines exist to fill up the screen. The cursor is the dark box that appears at line 1 in Figure 26-1.

We can specify several file names, and after saving the first file move on to the second file by entering **:n**, and continue going through the list of files in this way. Or we can specify a file and position the cursor on the last line in the file. The default is for the cursor to appear over the first character in the file, as shown in Figure 26-1.

Table 26-1 shows some of the ways in which we can start a **vi** session:

Table 26-1. Starting a vi Session

Command	Description
vi *file*	Edit *file.*
vi -r *file*	Edit last saved version of *file* after a crash.
vi -R *file*	Edit *file* in read-only mode.
vi + *n file*	Edit *file* and place cursor at line *n*.
vi + *file*	Edit *file* and place cursor on last line.
vi *file1 file2 file3 ...*	Edit *file1* through *file3,* and after saving changes in *file1,* you can move to *file2* by entering **:n**.
vi +/*string file*	Edit *file* and place cursor at the beginning of the line containing *string*.

Figure 26-2 shows editing **wisdom** and placing the cursor at line 5 with the command **vi +5 wisdom**:

Figure 26-2. Editing the File wisdom and Placing Cursor at Line 5 with vi +5 wisdom

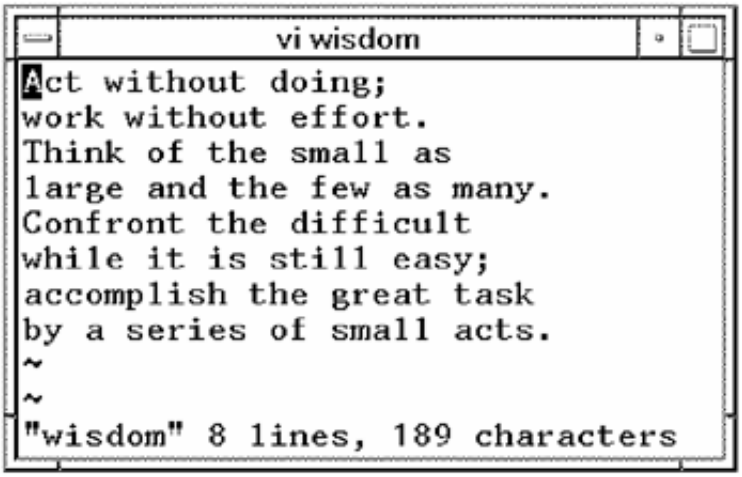


Figure 26-3 shows editing **wisdom** and placing the cursor at the last line of the file with the command **vi + wisdom**.

Figure 26-3. Editing the File wisdom and Placing Cursor at Last Line with vi + wisdom

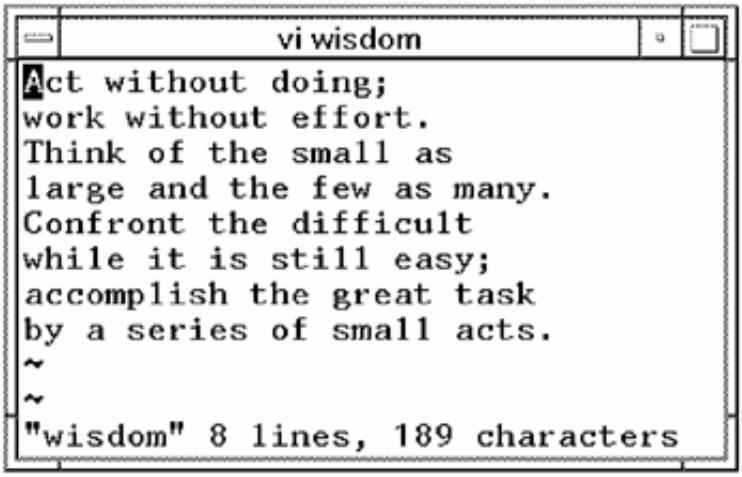
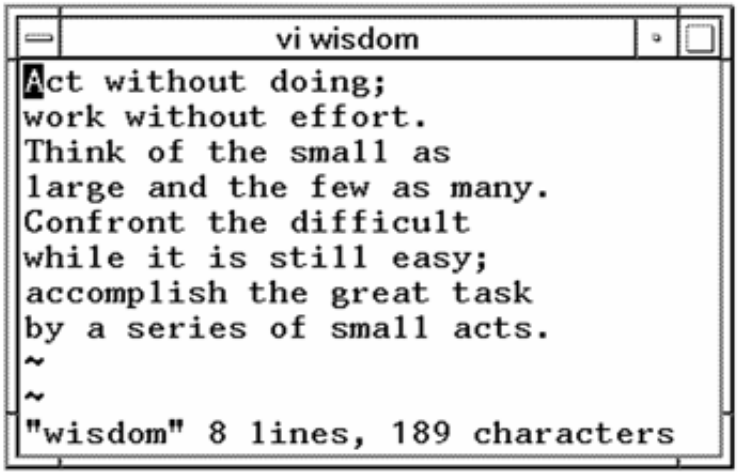


Figure 26-4 shows editing **wisdom** and placing the cursor at the line containing *task* with **vi +/task wisdom**.

Figure 26-4. Editing the File wisdom and Placing Cursor at Line Containing *task* with vi +/task wisdom



Cursor Control Commands

A key skill to develop in **vi** is getting the cursor to the desired position. You do this in *command mode*. You have a variety of ways to move the cursor around the screen. Table 26-2 summarizes some of the more commonly used cursor movements:

Table 26-2. Cursor Control Commands In vi

Command	Cursor Movement
h or **^h**	Move left one character.
j or **^j** or **^n**	Move down one line.
k or **^p**	Move up one line.
l or **space**	Move right one character.
G	Go to the last line of the file.
*n***G**	Go to line number *n*.
G$	Go to the last character in the file.
1G	Go to the first line in the file.
w	Go to the beginning of the next word.
W	Go to the beginning of next word, ignore punctuation.
b	Go to the beginning of the previous word.
B	Go to the start of previous word, ignore punctuation.
L	Go to the last line of the screen.
M	Go to the middle line of the screen.
H	Go to the first line of the screen.
e	Move to the end of the next word.
E	Move to the end of the next word, ignore punctuation.
(	Go to the beginning of the sentence.
)	Go to the end of the sentence.

{	Go to the beginning of the paragraph.
}	Go to the beginning of the next paragraph.
0 or |	Go to the first column in the current line.
*n***|**	Go to column *n* in the current line.
^ (caret)	Go to the first non-blank character in the current line.
$	Go to the last character in the current line.
+ or *return*	Go to the first character in the next line.
-	Go to the first non-blank character in the previous line.

I know that the fact that you have to remember these commands in order to get the cursor to the desired position may seem a little strange at first, but this is the way **vi** works. Let's use **wisdom** to show how some of these cursor movements work. Figures 26-5 and 26-6 show some cursor movements. Like all of the upcoming figures, Figures 26-5 and 26-6 show **wisdom** appears before a command is entered on the left and the result after the command is entered on the right. The command issued appears in the middle. Some of the commands in upcoming figures use the *enter* and *escape* keys.

Figure 26-5. Examples of Cursor Movement in vi (h, j, k, and l)

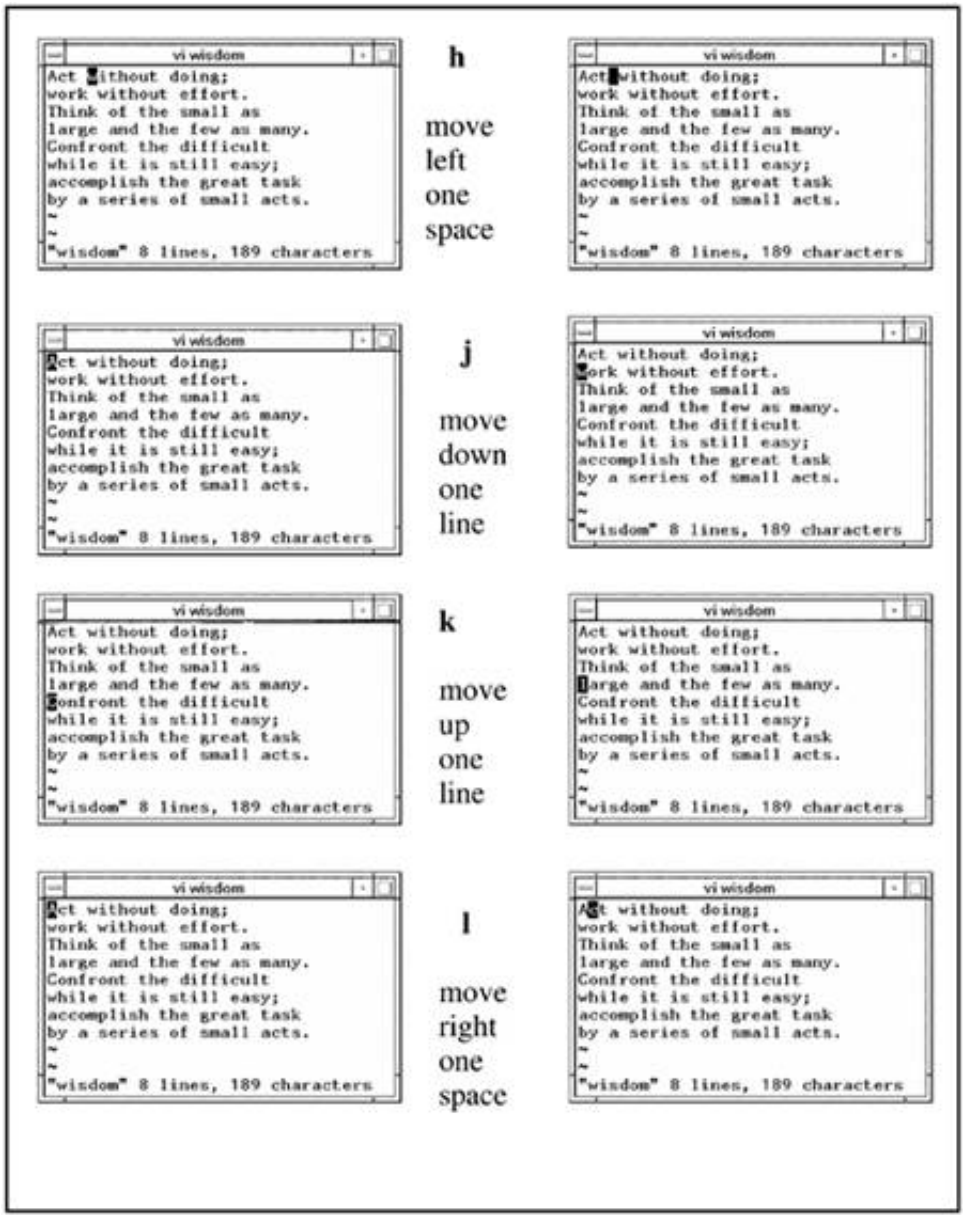
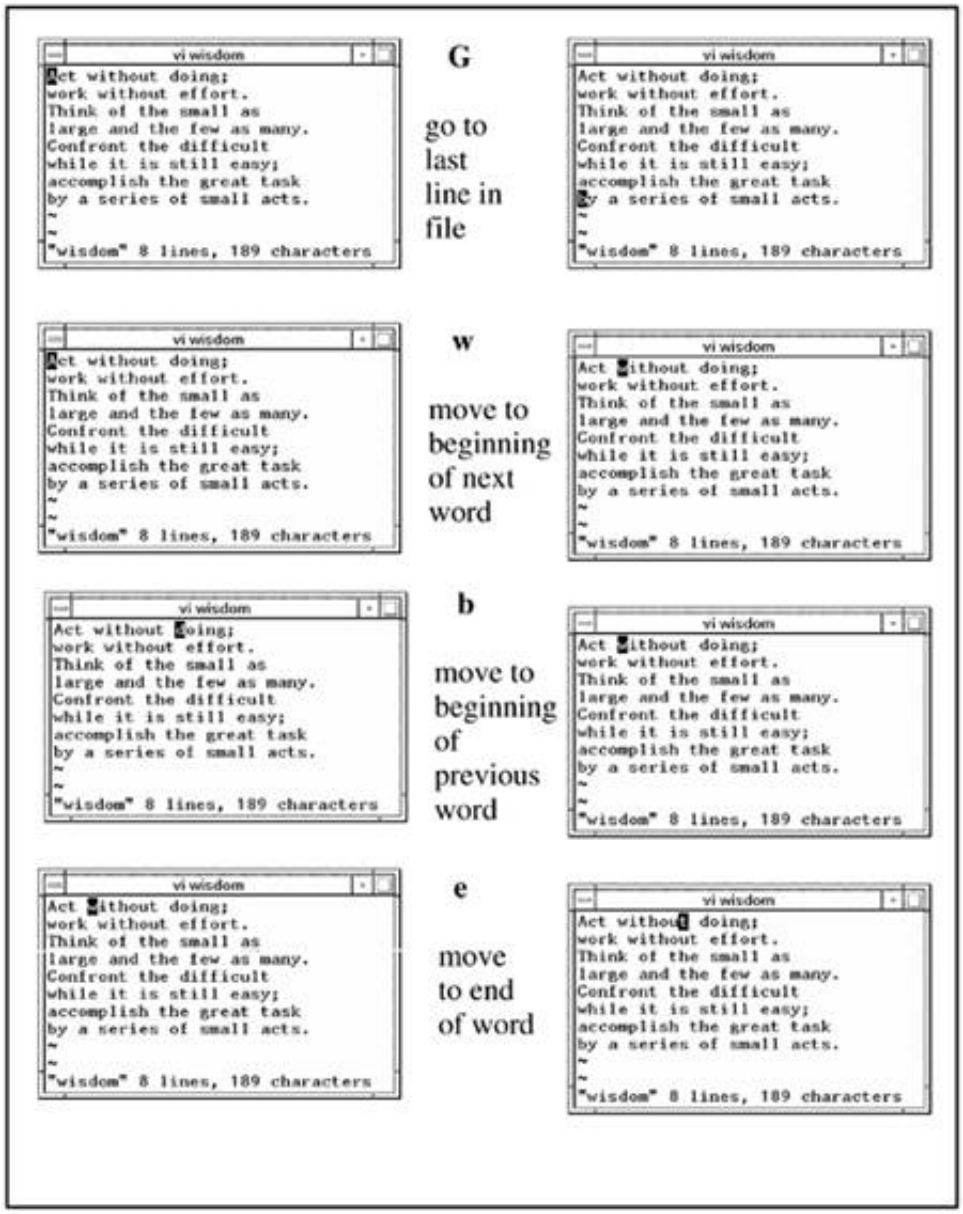


Figure 26-6. Examples of Cursor Movement in vi (G, w, b, and e)



Adding Text in vi

Now that we know how to move around the cursor, let's do something with it. You need to first learn about cursor movement, because the commands for adding text take place relative to the position of the cursor. Table 26-3 summarizes some commands for adding text:

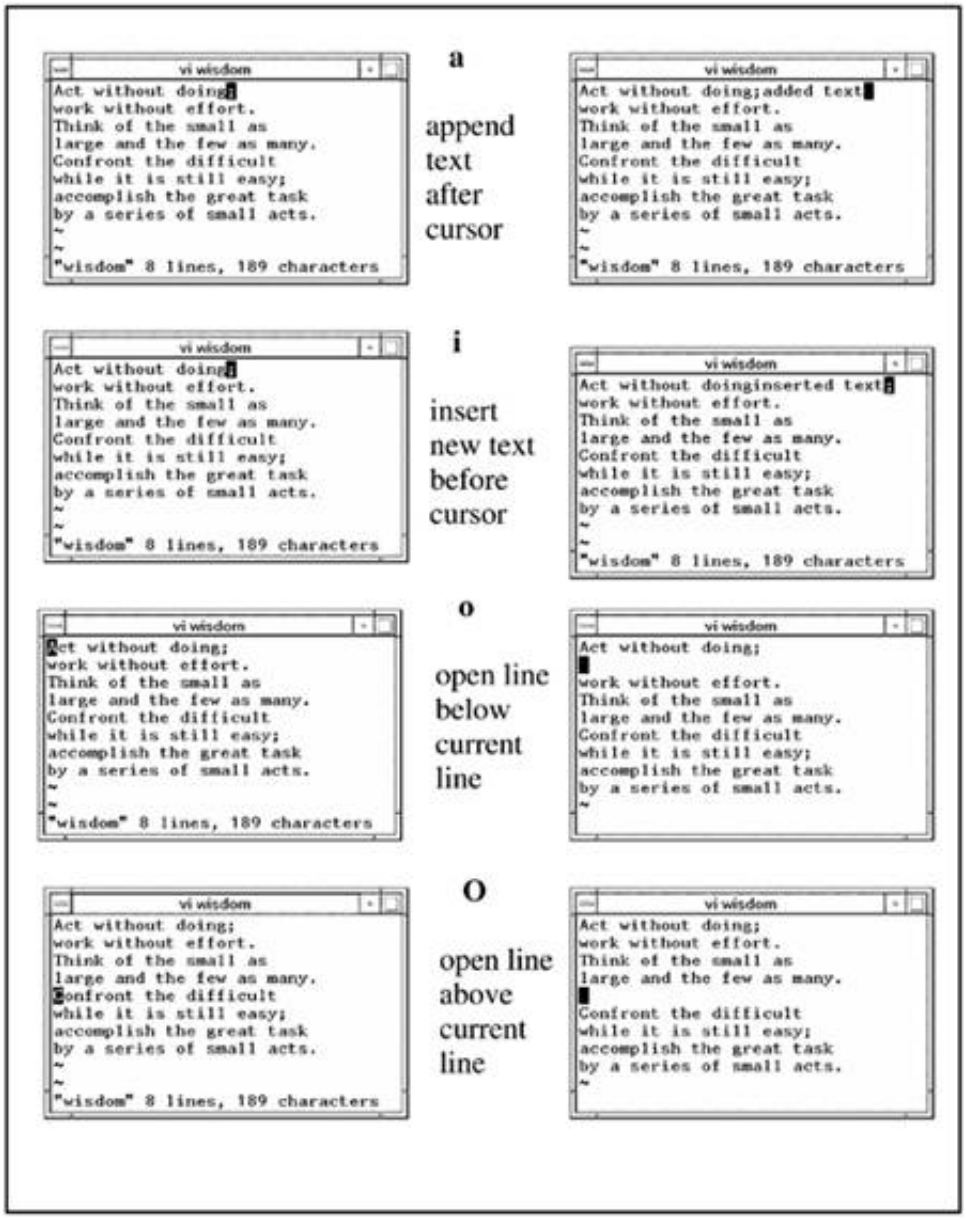
Table 26-3. Adding Text In vi

Command	Insertion Action
a	Append new text after the cursor.
A	Append new text after the end of the current line.
i	Insert new text before the cursor.

I	Insert new text before the beginning of the current line.
o	Open a line below the current line and insert.
O	Open a line above the current line and insert.
:r *file*	Read *file* and insert after the current line.
:*n***r** *file*	Read *file* and insert after line *n*.
escape	Get back to command mode.
^v *char*	Ignore special meaning of *char* when inserting. This is for inserting special characters.

Let's now look at some examples of adding text into **wisdom** in Figure 26-7:

Figure 26-7. Examples of Adding Text in vi



Deleting Text in vi

We also need to learn about cursor movement before learning how to delete text, because the commands for deleting text take place relative to the position of the cursor. Table 26-4 summarizes some commands for deleting text:

Table 26-4. Deleting Text In vi

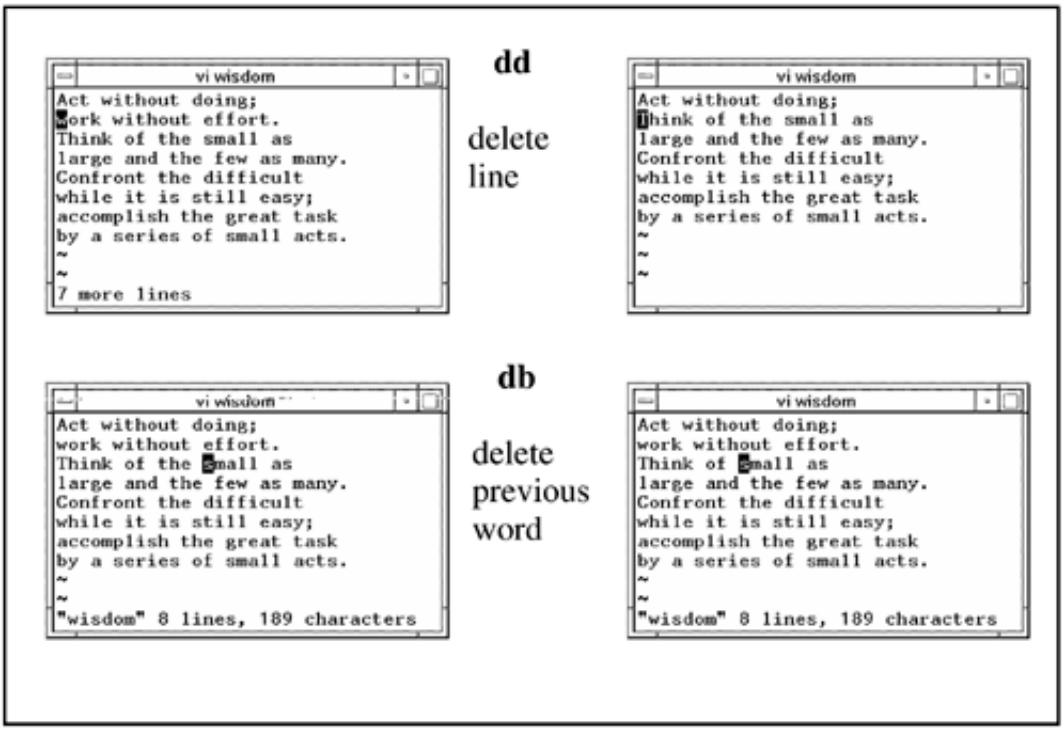
Command	Deletion Action
x	Delete the character at the cursor. You can also put a number in front of x to specify the number of characters to delete.
*n***x**	Delete *n* characters beginning with the current.
X	Delete the previous character. You can also put a number in front of X to specify the number of previous characters to delete.
*n***X**	Delete previous *n* characters.
dw	Delete to the beginning of the next word.
*n***dw**	Delete the next *n* words beginning with the current.
dG	Delete lines to the end of the file.
dd	Delete the entire line.
*n***dd**	Delete *n* lines beginning with the current.
db	Delete the previous word.
*n***db**	Delete the previous *n* words beginning with the current.
:*n***,***m***d**	Deletes lines *n* through *m*.
D or **d$**	Delete from the cursor to the end of the line.
d*cursor_command*	Delete text to the *cursor_command*. **dG** would delete from the current line to the end of the file.
^h or *backspace*	While inserting, delete the previous character.
^w	While inserting, delete the previous word.

Let's now look at some examples of deleting text from **wisdom** in Figures 26-8 and 26-9:

Figure 26-8. Examples of Deleting Text in vi (x, X, dw, and dG)



Figure 26-9. Examples of Deleting Text in vi (dd and db)



Changing Text in vi

Okay, you've added text and deleted text, and now you want to change text. **vi** isn't so bad so far, is it? Table 26-5 summarizes some commands for changing text:

Table 26-5. Changing Text In vi

Command (Preceding these commands with a number repeats the commands any number of times.)	Replacement Action
r*char*	Replace the current character with *char*.
R*text escape*	Replace the current characters with *text* until *escape* is entered.
s*text escape*	Substitute *text* for the current character.
S or **cc***text escape*	Substitute *text* for the entire line.
cw*text escape*	Change the current word to *text*.
C*text escape*	Change the rest of the current line to *text.*
cG *escape*	Change to the end of the file.
c*cursor_cmd* ***text*** *escape*	Change to *text* from the current position to *cursor_cmd.*

Let's now look at some examples of replacing text from **wisdom** in Figures 26-10 and 26-11:

Figure 26-10. Examples of Changing Text in vi (r, R, cw, and cG)

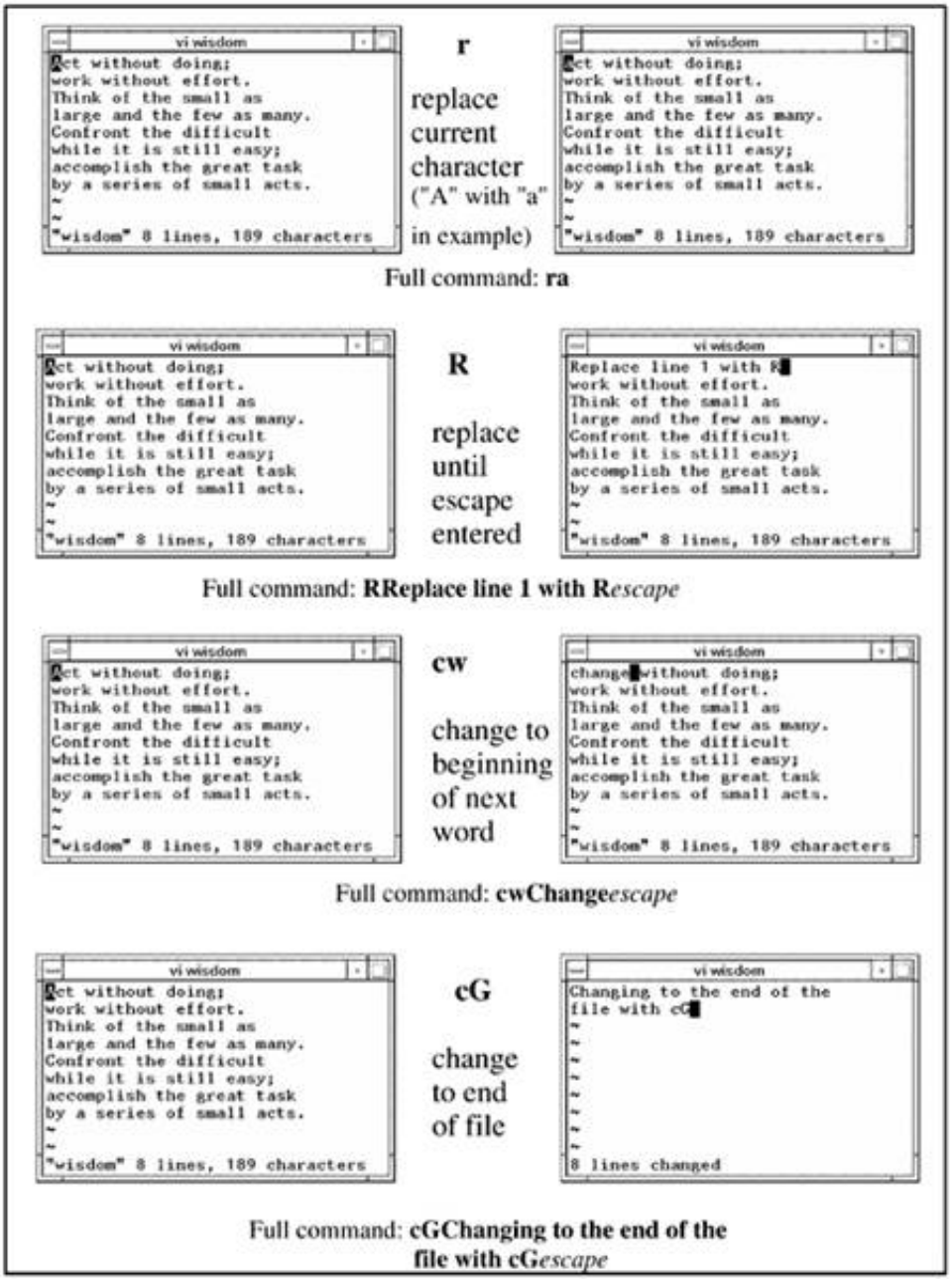
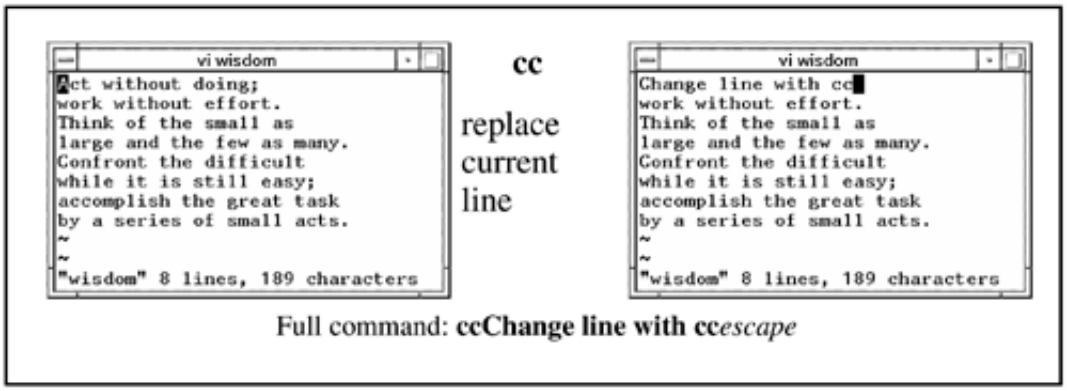


Figure 26-11. Example of Changing Text in vi with cc



Search and Replace in vi

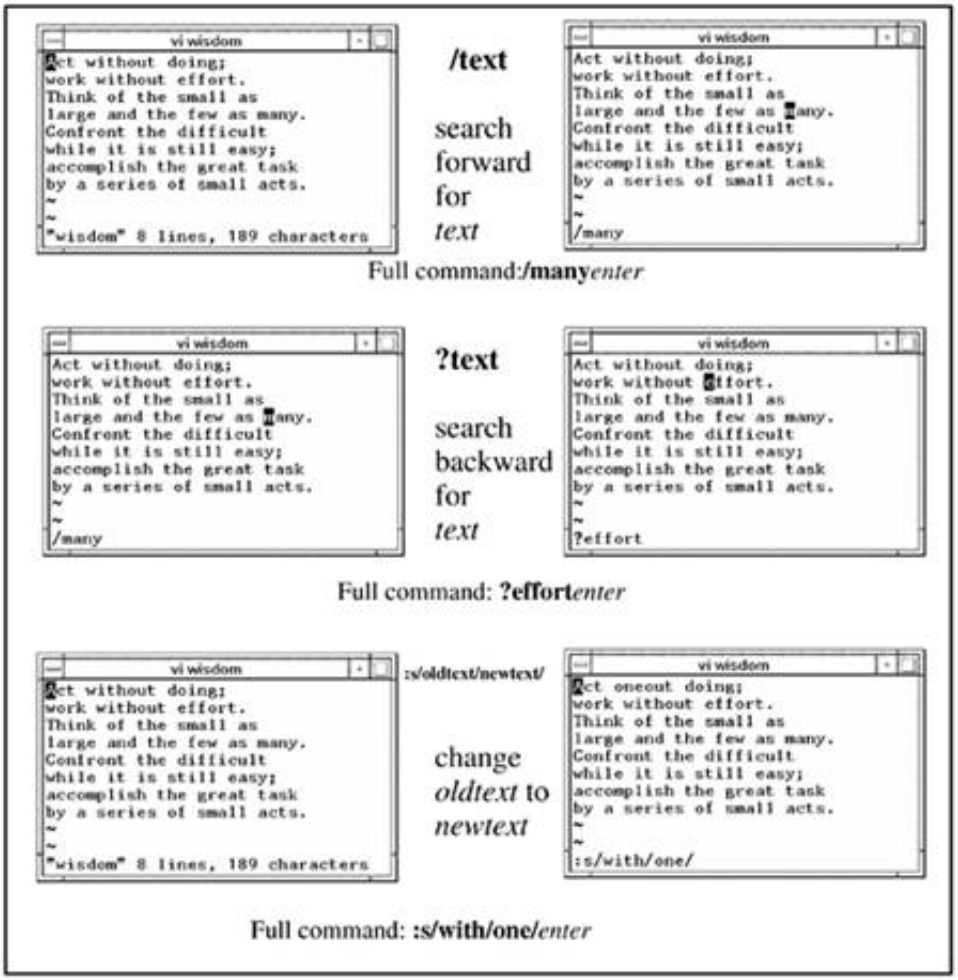
You have a lot of search and replace functionality in **vi**. Table 26-6 summarizes some of the more common search-and-replace functionality in **vi**:

Table 26-6. Search And Replace In vi

Command	Search and Replace Action
/*text*	Search for ***text*** going forward into the file.
?*text*	Search for ***text*** going backward into the file.
n	Repeat the search in the same direction as the original search.
N	Repeat the search in the opposite direction as the original search.
f*text*	Search for *text* going forward in the current line.
F*text*	Search for *text* going backward in the current line.
t*text*	Search for *text* going forward in the current line and stop at the character before *text.*
T*text*	Search for *text* going backward in the current line to character after *text.*
:set ic	Ignore case when searching.
:set noic	Make searching case-sensitive.
:s/*oldtext***/***newtext***/**	Substitute *newtext* for *oldtext*.
:*m*,*n***s/***oldtext***/***newtext***/**	Substitute *newtext* for *oldtext* in lines *m* through *n*.
&	Repeat the last **:s** command.
:g/*text1***/s/***text2***/***text3*	Find line containing *text1,* replace *text2* with *text3.*
:g/*text***/***command*	Run *command* on all lines that contain *text*.
:v/*text***/***command*	Run *command* on all lines that do not contain *text*.

Let's now look at some examples of searching and replacing text in **wisdom** in Figure 26-12:

Figure 26-12. Examples of Search and Replace in vi



You can perform advanced searches with **:g** and **:v**. You can find and display all the lines in a file containing **while** with the following command:

```
:g/while/p
```

The /p in this command line is the print command used with the ex editor. You could find all the lines in the file that contain *while* and delete those lines with the following command:

```
:g/while/d
```

You can also specify the specific line numbers for which you want the search to take place. The following command finds all the lines between 10 and 20 that contain *while* and prints the line number on which they appear:

```
:10,20g/while/nu
```

:g runs a command on the lines that contain the text for which we are searching, and **:v** runs a command on the lines that do not contain the specified text. The following three commands act on the lines that do not contain *while*, in the same way that the previous three act on the lines that do contain *while*:

```
:v/while/p
```

```
:v/while/d
```

```
:10,20v/while/nu
```

The first command prints lines that do not contain *while*. The second command deletes the lines on which *while* does not appear. The third command prints the line number between *10* and *20* on which *while* does not appear.

Copying Text in vi

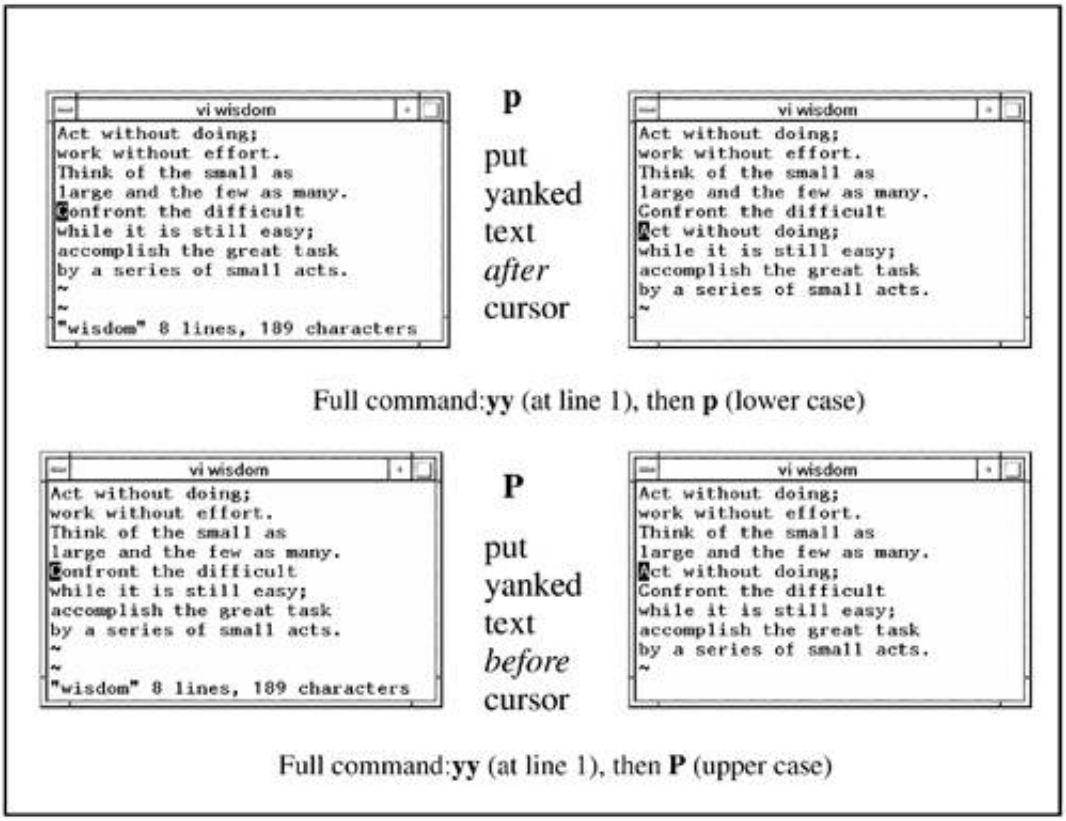
You can copy text in **vi**. Some commands for copying are shown in Table 26-7:

Table 26-7. Copying In vi

Command	Copy Action
yy	Yank the current line.
nyy	Yank **n** lines.
p (lower case)	Put the yanked text after the cursor.
p (upper case)	Put the yanked text before the cursor.
*"(a-z)n*__yy__	Copy *n* lines into the buffer named in parentheses. Omit *n* for the current line.
*"(a-z)n*__dd__	Delete *n* lines into the buffer named in parentheses. Omit *n* for the current line.
*"(a-z)*__p__	Put lines named in the buffer in parentheses after current line.
*"(a-z)*__P__	Put lines named in the buffer in parentheses before the current line.

Let's now look at some examples of copying text in **wisdom** in Figure 26-13:

Figure 26-13. Copying in vi



Undo and Repeat in vi

You can easily undo and repeat changes in **vi** with the commands shown in Table 26-8:

Table 26-8. Unso In vi

Command	Undo Action
u	Undo the last change.
U	Undo all changes to the current line.
. (period)	Repeat the last change.
, (comma)	Repeat, in reverse direction, last **f, F, t**, or **T** search command.
; (semi-colon)	Repeat last **f, F, t,** or **T** search command.
"*n***p**	Retrieve the last *n*th delete (a limited number of deletes are in the buffer, usually nine).
n	Repeat last **/** or **?** search command.
N	Repeat, in reverse direction, last **/** or **?** search command.

Save Text and Exit vi

You have a number of different ways to save files and exit **vi**, some of which are summarized in Table 26-9:

Table 26-9. Saving Text and Exiting vi

Command	Save and/or Quit Action
:w	Save the file but don't exit **vi**.
:w *file*	Save changes in *file* but don't quit **vi**.
:wq or **ZZ** or **:x**	Save the file and quit **vi**.
:q!	Quit **vi** without saving the file.
:e!	Re-edit the file, discarding changes since the last write.

Options in vi

There are many options that you can set and unset in **vi**. To set an option, you type **:set *option***. To unset an option, you type **:set no*option***. Table 26-10 summarizes some of the more commonly used options:

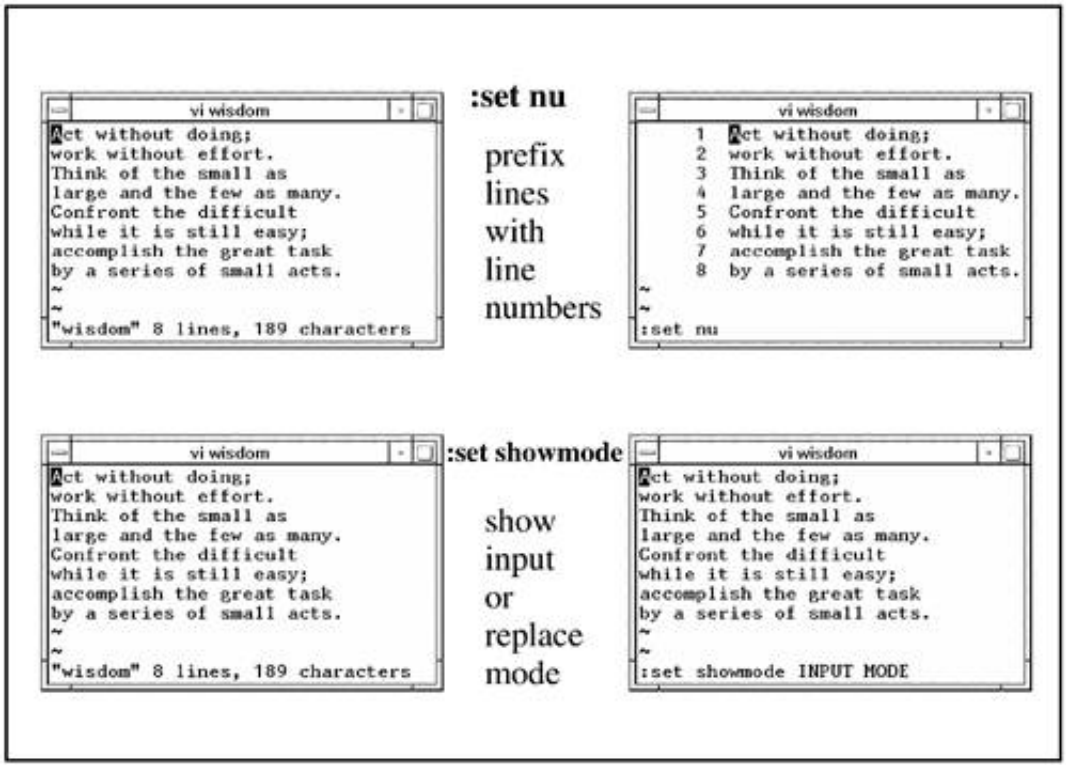
Table 26-10. Options In vi

Option	Action
:set all	Print all options.
:set no*option*	Turn off *option*.

:set nu	Prefix lines with line number.
:set showmode	Show whether input or replace mode.
:set noic	Ignore case when searching.
:set list	Show tabs (^I) and end of line ($).
:set ts=8	Set tab stops for text input.
:set window=*n*	Set number of lines in a text window to *n*.

Let's now prefix the lines with line numbers and show the input or replace mode in Figure 26-14:

Figure 26-14. Options in vi



Many additional options are available beyond those in Table 26-10. The following is a list of options produced on a UNIX system from the **:set all** command. You should issue this command when in **vi** to see the options available to you:

```
:set all
noautoindent
autoprint
noautowrite
nobeautify
directory=/var/tmp
nodoubleescape
noedcompatible
noerrorbells
noexrc
flash
hardtabs=8
noignorecase
keyboardedit
nokeyboardedit!
nolisp
nolist
magic
```

```
mesg
nomodelines
nonumber
nonovice
nooptimize
paragraphs=IPLPPPQPP LIpplpipnpbp
prompt
noreadonly
redraw
remap
report=5
scroll=11
sections=NHSHH HUuhsh+c
shell=/sbin/sh
shiftwidth=8
noshowmatch
noshowmode
noslowopen
tabstop=8
taglength=0
tags=tags /usr/lib/tags
tagstack
term=hp
noterse
timeout
timeoutlen=500
ttytype=hp
warn
window=23
wrapscan
wrapmargin=0
nowriteany
noshowmatch
noshowmode
```

Many of the options are preceded by a "no," indicating that the option is not set. You may want to list your options with **:set all** and then experiment with the options of interest to you to see the effect they will have on your **vi** session.

Status in vi

You can obtain a lot of useful status information with some simple commands in **vi**. You can display the current line number, number of lines in the file, file name, and other status information with the commands shown in Table 26-11:

Table 26-11. Status In vi

Option	Action
:.=	Print the current line number.
:=	Print the number of lines in the file.
^g	Show the file name, current line number, total lines in the file, and percent of file location.
:l	Use the letter "l" to display various special characters such as tab and newline.

Section Positioning and Placing Marks in Text

You can define sections of text to which you can move as well as mark text with characters and move to those marks. Table 26-12 summarizes positioning and marking in **vi**:

Table 26-12. Positioning and Marking In vi

Table 26-12. Positioning and Marking in vi

Option	Action
{	Insert **{** in first column to define section.
[[	Go back to beginning of section.
]]	Forward to beginning of next section.
m*(a-z)*	Mark current position with a letter such as **mz** for mark *z*.
'*(a-z)*	Move cursor to specified mark such as **'z** for move to *z*.

Joining Lines in vi

You can join one or more lines in **vi** using the commands shown in Table 26-13:

Table 26-13. Joining Lines In vi

Option	Action
J	Join the next line to the end of the current line.
*n***J**	Join the next *n* lines.

Cursor Placement and Adjusting the Screen

You can place the cursor anywhere in your file and adjust the screen in a variety of ways using the commands shown in Table 26-14:

Table 26-14. Cursor Placement and Adjusting the Screen In vi

Option	Action
H	Move cursor to top line of the screen.
*n***H**	Move cursor to *n* line from the top of the screen.
M	Move cursor to the middle of the screen.
L	Move cursor to the bottom line of the screen.
*n***L**	Move cursor to line *n* from the bottom of the screen.
^e (control-e)	Move screen up one line.
^y	Move screen down one line.
^u	Move screen up one-half page.
^d	Move screen down one-half page.
^b	Move screen up one page.
^f	Move screen down one page.
^l (letter l)	Redraw screen.
z - *return*	Make current line the top of screen.

*n***z** - *return*	Make *n* line the top of screen.
z.	Make current line the middle line.
*n***z.**	Make line *n* the middle line on screen.
z-	Make current line the bottom line.
*n***z-**	Make line *n* the bottom line on screen.

Shell Escape Commands

You can run a UNIX command without exiting **vi** by using shell escape commands. You could do something as simple as start a sub-shell with the **:sh** command. You could also run a command outside the file you are editing without exiting **vi**. Table 26-15 describes shell escape commands:

Table 26-15. Shell Escape Commands In vi

Option	Action
:! *command*	Execute shell command *command* such as **:! ls**.
:!!	Execute last shell command.
:r! *command*	Read and insert output from *command,* such as **:r! ls** to run **ls** and read contents.
:w !*command*	Send currently edited file to *command* as standard input and execute *command,* such as **:w ! grep all**.
:cd *directory*	Change the current working directory to *directory.*
:sh	Start a sub-shell and use **^d** (control-d) to return to **vi**.
:so *file*	Read and execute commands in the shell program *file.*



An example of using **:w** would be to send the file **wisdom** as a standard input to **grep,** looking for all lines that contain *all*, as in the following example:

```
:w ! grep all
Think of the small as
by a series of small acts.
```

You can issue the **:so** command to read and execute the commands in a file. Issuing the following command when in **vi** would run the commands in the file **file_with_commands**:

```
:so file_with_commands
```

This file contains the following two commands:

```
:set nu
:g/all/p
```

When we issue the earlier **:so** command, line numbers are shown with the **:set nu** command and the following lines containing *all* are printed:

```
Think of the small as
by a series of small acts.
```

Macros and Abbreviations

You are not limited to issuing individual **vi** commands. You can define strings of **vi** commands and define a key corresponding to this string that you can recall. When defining the keys for your macros, you can't use the following: **K V g q v * =** and function keys. There are also control keys you can't use, so stay away from control keys in general. Table 26-16 shows macros and abbreviations.

Table 26-16. Macros and Abbreviations In vi

Option	Action
:map *key command_seq*	Define *key* to run *command_seq,* such as **:map e ea** to append text whenever you use **e** to move to the end of a word.
:map	Display all defined macros on the status line.
:umap *key*	Remove the macro for *key.*
:ab *string1 string2*	Define an abbreviation such that when *string1* is inserted, replace it with *string2.* When inserting text type *string1,* press *escape* key and *string2* will be inserted.
:ab	Display all abbreviations.
:cd *directory*	Change the current working directory to *directory.*
:una *string*	Unabbreviate *string.*
	Avoid control keys, symbols, and don't use characters: **K V g q v * =** and function keys.

An example of using the **map** command would be to automatically add text when you move to the end, as shown with the following **map** command:

```
:map e ea
```

This command maps e to ea. When you go to the end of the next word with e, you are also placed in insert mode with a so that you can append new text immediately after the end of the word.

You can also abbreviate long sequences with ab. For instance, you could abbreviate *system administration* with *sa* with the following command:

```
:ab sa system administration
```

Now, whenever you insert text, type *sa* and then press the *escape* key to complete the insert; the string *system administration* appears. *sa* is an abbreviation for *system administration.*

Indenting Text

You can indent text in a variety of different ways. Table 26-17 shows some of the more commonly used indenting commands:

Table 26-17. Indenting Text In vi

Option	Action
^i (control i) or *tab*	While inserting text, insert on shift width. Shift width can be defined.
:set ai	Turn on auto-indentation.
:set sw=*n*	Set shift width to *n* characters.

n<<	Shift *n* lines left by one shift width.
n>>	Shift *n* lines right by one shift width. For example, **3>>** shifts the next three lines right by one shift width.

Before you adjust the shift width, you may want to issue **:set all** in order to see the current number of characters to which the shift width is set. It is usually eight characters by default. To set the shift width to 16 characters, issue the following command:

```
:set sw=16
```

You can then shift over the next three lines to the right by 16 characters each, with the following command:

```
3>>
```

The next three lines are then shifted right by 16 characters.

Shell Filters

You can send information from the file you are editing to a command and then replace the original text with the output of the command. Table 26-18 shows a shell filter:

Table 26-18. Shell Filters In vi

Option	Action
!*cursor_command command*	Send text from the current position to that described by *cursor_command* to the shell *command.* For example, use **!} grep admin** to take text from the current position to the end of the paragraph, run this text through **grep** looking for the word *admin*, and replace existing text with the output of **grep**.

Pattern Matching

Pattern matching allows you to find patterns within the file you are editing. You can then perform functions such as changing what you have found in some way. Table 26-19 shows some of the most common pattern-matching commands:

Table 26-19. Pattern Matching in vi

Option	Action
^ (caret)	Match the beginning of the line. To search for **Think** at only the beginning of the line, you would use: **/^Think** You can use this in combination with **$**, which matches to the end of the line, to delete all blank lines with: **:g/^$/d**.
$	Match end of line. To match **last.** only when it is followed by a newline character, you would use: **/last.$**
.	Match any single character.

\<	Match beginning of word.
\>	Match end of word.
[*string***]**	Match any single character in *string*. To find **mp, mP, Mp**, or **MP**, use: **/[mM][pP]** Change all occurrences of **input** or **Input** to **INPUT** with: **:%s/[Ii]nput/INPUT/g**
[^*string***]**	Match any character not in *string.*
[*a-p***]**	Match any character between *a* and *p.*
*	Match zero or more occurrences of previous character in expression.
****	Escape meaning of next character. To search for **[**, use the following: **/\[**
****	Escape the **** character.

You may find pattern matching a little confusing when you first start to use it, so I'll go through several simple examples to get you started. Keep in mind that many of the pattern-matching techniques described here also work outside **vi** in your shell.

Matching a Set

We'll begin with the *square bracket operator*. To match any of the single characters *m, f,* or *p*, you would use the following:

```
/[mfp]
```

A common pattern to match would be a word with the first letter in the word, either uppercase or lowercase. To match *input* or *Input*, you would use the following:

```
/[Ii]nput
```

After you match either *Input* or *input*, you could then change it to *INPUT* with the following command:

```
:%s/[Ii]nput/INPUT/g
```

You can use sequences of expressions to search for more than one character, as shown in the following example:

```
/[mM][pP]
```

This sequence will match *mp, mP, Mp*, or *MP*. You are, in effect, searching for any of the four two-character strings.

Matching a Range

You can also use the square bracket operator to match single characters within a range. To find an occurrence of any digit in a file, you could use either of the two following square bracket searches:

```
/[0123456789]
```

or

```
/[0-9]
```

The hyphen denotes a range within the square bracket. To find any character, either uppercase or lowercase, you could use the following:

```
/[a-zA-Z]
```

To search for characters that normally have a special meaning, such as **[**, you can ignore, or escape, the special meaning by preceding the special character with a **** (backslash). To search for **[** in **vi**, for instance, you would use the following sequence:

```
/\[
```

This search finds the first occurrence of **[**.

Beginning and End of Line Search

You can specify that you wish your pattern match to take place at only the beginning or end of a line. To specify a beginning of the line pattern match, use the **^** (caret) preceding your desired pattern, as shown in the following example:

```
/^Think
```

This matches *Think* only when it appears at the beginning of a line.

To specify an end-of-the-line pattern match, use a **$** (dollar sign) following your desired pattern, as shown in the following example:

```
/last.$
```

This matches *last.* only when it is followed by a newline. Manual Pages for Commands Used in Chapter 24Manual Pages for Commands Used in Chapter 24

[Team LiB]



Manual Pages for Commands Used in Chapter 26

There are manual pages in Appendix A for the following commands used in this chapter:

```
awk
grep
sed
```






Chapter 27. Introduction to the KornShell, C Shell, and Bash

Different Shells

Introduction to KornShell

Introduction to the C Shell

Introduction to Bash



Different Shells

HP-UX, like all UNIX variants, allows you to select among different shells. A lot of material is available on the POSIX shell, which is the default HP-UX shell in most cases. There is less information available on the other widely used shells, so I have decided to cover them in this chapter. The shell is important to you as a system administrator and user because it is your window into the system. I decided to include the KornShell and C shell in this chapter because so many new HP-UX users and system administrators use them as their shells even though the POSIX shell is the default on most systems. I included Bash because it is widely used in Open Source enviornments and many Open Source GNU tools are available for HP-UX 11i. The upcoming sections cover these three shells. The KornShell section also covers some important topics related to shells such as *File Name Expansion, Redirection, Background Jobs*, and *umask and Permissions*. Some of these topics have been covered in other chapters of the book, but these topics are closely related to your shell, so I decided to include them here as well.

Introduction to KornShell

Most UNIX variants allow you to select among several shells. Because of its versatility and ease of use, many system administrators configure the KornShell for new users. The KornShell was derived from the Bourne Shell and has much of the same functionality of the Bourne Shell. **ksh** is the program you run on most UNIX systems, including HP-UX, that supplies KornShell functionality. It is also often referred to as the K shell. I use **ksh** throughout this chapter. You can use the **ksh** in the following three ways:

- Interactively type commands on the command line.
- Group commonly executed sets of commands into command files that you can execute by typing the name of the file.
- Create KornShell programs using the structured programming techniques of the shell.

These three techniques are listed in the order in which you'll probably use them. First, you log in and use interactive commands. Then you group together commonly used commands and execute them with a single command. Finally, you may want to create sophisticated shell scripts.

For this reason, I describe these aspects of the KornShell in the order in which they are listed. The command file and programming aspects of the KornShell are covered as part of the Shell Programming Chapter 28.

The examples in this chapter are performed on a variety of different systems, not just HP-UX. This is because many system administrators use **ksh** on several different UNIX variants. You would probably find both your user setup and the operation of the KornShell on other systems similar to the description in this chapter. Much of the setup of any shell is performed by the system administrator, so you will surely find differences in your KornShell setup compared with what is shown in this chapter. In general, however, the operation of the KornShell is similar from one system to another.

Startup Files

The first activity you perform after you log into the system is to issue commands at the prompt. A command you may want to issue immediately is **ls -al**. Here is what I see on my system after executing this command, producing a long listing of all files:

```
martyp $ ls -al
total 22
drwxr-xr-x   2 martyp   staff        512 Mar 15 11:37.
drwxrwxr-x   4 root     sys          512 Mar  4 09:24 ..
-rw-r--r--   1 martyp   staff        124 Mar 15 11:36 .cshrc
-rw-r--r--   1 martyp   staff        562 Mar  4 09:24 .profile
-rw-------  1 martyp   staff      7056 Apr  6 07:24 .sh_history
martyp $
```

This produces a short list of files. Some of these files, such as **.profile** and **.cshrc**, are startup files for **ksh** and the C shell, respectively. Upon logging in to a system, you normally execute system startup files for your shell and then execute any local startup files that you have in your home directory. In this case, I have a minimal **ksh** startup file in my home directory - only a **.profile** that has very little in it. Virtually all the startup activity for this user comes from the system profile, usually **/etc/profile.** You can usually read **/etc/profile**, so you can see what your system administrator has set up for you and other users. You can then modify your local **.profile** to include a variety of functionality. Your local **.profile** is usually run at login, immediately after the system **/etc/profile**.

After running all the startup scripts associated with your **ksh** login, your environment is set up for you. Although our local **.profile** didn't do much other than set up our prompt, a lot of setup took place with the system files that were run.

Another file you have is called the *Environment* file. This file name is defined by the environment variable **ENV**. This file is usually **.kshrc** in your home directory. Using the environment file, you can define which options, aliases, and other information will be passed to subprocesses. By default, your existing

environment variables are passed to subprocesses. You can use the environment file to pass other information along as well.

You should take a look at your **ksh** startup files, including **/etc/profile, .profile**, and **.kshrc**.

We'll discuss much of the specific **ksh** functionality provided for you as part of your startup programs in the upcoming sections.

The History File, .sh_history

ksh keeps a history list of the commands you have issued. If you wish to reissue a command or view a command you earlier issued, you can use the history list. By default, **.sh_history** in your home directory is used as a history file. When your system administrator creates your **ksh** home directory, this file is probably present and used as your default history file.

By default, most systems save the 128 most recently executed commands. You can specify any number of commands to be included in the history list. The following line sets the history list to 200:

```
HISTSIZE=200
```

Most users make this entry in their home **.profile**. After you have added this line to your **.profile**, the history list is set to *200*.

Recalling from the History List

You can view the most recent commands issued, along with their corresponding line numbers, with the **history** command as shown in the following example:

```
martyp $ history
116     history 128
117     history 128 | more
118     history 200
119     alias
120     history 1
121     inv
122     env
123     env | more
124     history 1
125     more .profile
126     env | grep HIS
127     exit
128     env
129     env | grep HIS
130     more .profile
131     history
martyp $
```

Notice in this example that command number *127* is the **exit**, or command to log out, from the last session. Command number *128* is the **env** command I issued immediately upon establishing the next session.

We can also print the history list without line numbers using the *-n* option, as shown in the following example:

```
martyp $ history -n
        history 128 | more
        history 200
        alias
        history 1
        inv
        env
        env | more
        history 1
        more .profile
        env | grep HIS
        exit
        env
        env | grep HIS
        more .profile
```

```
        history
        history -n
martyp $
```

To produce a complete list of commands in the history list, we issue the following command:

```
martyp $ history 0
5       env
6       more .profile
7       set
8       env | grep CDP
9       env | grep cdp
10      echo $SHELLL
11      echo $SHELL
12      more .profile
13      ll
14      ls -al /etc/profile
15      more /etc/profile
16      more /etc/profile | grep PS
17      exhoecE
18      ECHO
19      echo $ENV
20      env | more
21      echo $ENV
22      env | more
23      env | grep ENV
24      env | grep env
25      more /usr/bin/env
26      ls -al
27      more /etc/.kshrc
28      find / -name .kshrc
29      more /etc/passwd
30      more /etc/passwd | grep /home
31      ll /home/oracle
32      ls -al /home/oracle
33      ll /home
34      ls -al /home
35      ls -al /home/ptc-nfs
36      ls -al /home/ptc-nfs | more
37      ls -al /root
38      cd /root/users
39      ls -al
40      ls -al verasu
41      ls -al rodtsu
42       cd rodtsu
43      more .kshrc
44      more .env
45      ls
46      ls -l
47      ls -a
48      more .kshrc
49      cd ..
50      ls -al
51      ls -al | more
52      exit
53      ls -al
54      more .sh*
55      history
56      env | grep his
57      env | grep HIS
58      env | grep IS
59      env | more
60       env | grep FC
61      env | grep EDI
62      history
63      echo $HISTIZE
64      echo $HISTSIZE
65      env | more
66      ls -al
67      history 128
68      history
69      history 1 104
70      exit
71      alias
72      more /etc/profile | grep alias
```

```
73      ll
74      la -al
75      ls -al
76      more .profile
77      ls -al
78      more .cshrc
79      alias ll="ls -al"
80      aliase
81      alias
82      ll
83      exit
84      alias
85      alias
86      history
87      fc -l
88      man fc
89      man fc
90      man fc
91      alias
92      ps
93      ps -ef
94      ps -efl
95      ps
96      ls -alF
97      ls -al
98      alias ls="ls -al"
99      alias
100     ls
101     alias
102     unalias ls
103     ls
104     ls -al
105     alias
106     history
107     alias
108     unalias history
109     history
110     fc -l
111     history
112     alias hisotry="fc -l"
113     history
114     alias history="fc -l"
115     history
116     history 128
117     history 128 | more
118     history 200
119     alias
120     history 1
121     inv
122     env
123     env | more
124     history 1
125     more .profile
126     env | grep HIS
127     exit
128     env
129     env | grep HIS
130     more .profile
131     history
132     history 0
martyp $
```

history 0 prints from the very first command, zero, to the present command. Because only 128 commands are saved by default, the first few oldest commands, zero through four, dropped off the history list. This means that we really started at command number five when we requested to start the list from command number zero.

To produce a list from command *100* to the present command, we issue the following:

```
martyp $ history 100
100     ls
101     alias
102     unalias ls
103     ls
104     ls -al
```

```
105     alias
106     history
107     alias
108     unalias history
109     history
110     fc -l
111     history
112     alias hisotry="fc -l"
113     history
114     alias history="fc -l"
115     history
116     history 128
117     history 128 | more
118     history 200
119     alias
120     history 1
121     inv
122     env
123     env | more
124     history 1
125     more .profile
126     env | grep HIS
127     exit
128     env
129     env | grep HIS
130     more .profile
131     history
132     history 0
133     history
134     history -n
135     history 100
martyp $
```

To list the current command and the *20* commands preceding it, we issue the following:

```
martyp $ history -20
116     history 128
117     history 128 | more
118     history 200
119     alias
120     history 1
121     inv
122     env
123     env | more
124     history 1
125     more .profile
126     env | grep HIS
127     exit
128     env
129     env | grep HIS
130     more .profile
131     history
132     history 0
133     history
134     history -n
135     history 100
136     history -20
martyp $
```

The *-20* goes back from the current command a full 20 commands. In this case, **history** is using the current command as the default place to begin producing the list. You can list the last 20 commands *preceding* the current command by specifying the range *-1 -20* as shown in the following example:

```
martyp $ history -1 -20
136     history -20
135     history 100
134     history -n
133     history
132     history 0
131     history
130     more .profile
129     env | grep HIS
128     env
127     exit
126     env | grep HIS
```

```
125     more .profile
124     history 1
123     env | more
122     env
121     inv
120     history 1
119     alias
118     history 200
117     history 128 | more
martyp $
```

Notice in this example that the current command, **history -1 -20**, is not shown in the list because the *-1* starts the list with the preceding command. This is effectively producing a range of commands to list.

You can also reverse the order of the last 20 commands by specifying *--20* first and then the *-1* as shown in the following example:

```
martyp $ history -20 -1
118     history 200
119     alias
120     history 1
121     inv
122     env
123     env | more
124     history 1
125     more .profile
126     env | grep HIS
127     exit
128     env
129     env | grep HIS
130     more .profile
131     history
132     history 0
133     history
134     history -n
135     history 100
136     history -20
137     history -1 -20
martyp $
```

You can also produce a list from the last time that a command was issued to the current command. The following example shows producing a list from the last **fc** command to the current command:

```
martyp $ history fc
110     fc -l
111     history
112     alias hisotry="fc -l"
113     history
114     alias history="fc -l"
115     history
116     history 128
117     history 128 | more
118     history 200
119     alias
120     history 1
121     inv
122     env
123     env | more
124     history 1
125     more .profile
126     env | grep HIS
127     exit
128     env
129     env | grep HIS
130     more .profile
131     history
132     history 0
133     history
134     history -n
135     history 100
136     history -20
137     history -1 -20
138     history -20 -1
139     history
140     history grep
```

```
141     history env
142     history
143     history fc
martyp $
```

You can also reverse this list using the *-r* option as shown in the following example:

```
martyp $ history -r fc
144     history -r fc
143     history fc
142     history
141     history env
140     history grep
139     history
138     history -20 -1
137     history -1 -20
136     history -20
135     history 100
134     history -n
133     history
132     history 0
131     history
130     more .profile
129     env | grep HIS
128     env
127     exit
126     env | grep HIS
125     more .profile
124     history 1
123     env | more
122     env
121     inv
120     history 1
119     alias
118     history 200
117     history 128 | more
116     history 128
115     history
114     alias history="fc -l"
113     history
112     alias hisotry="fc -l"
111     history
110     fc -l
martyp $
```

Having access to this history list is a great feature of the KornShell. You can view all the commands you have issued, including your typing errors, in any format you wish. In the next section, we start using the commands in the history list by recalling them.

Re-executing Commands with r

You can reissue a command with the **r** command. To reissue the last command, you simply type **r**. In the following example, I issue the command **history 5** to view the last five commands from the history list. I then type **r** to reissue the last command, and again the last five commands are listed:

```
martyp $ history -5
301     whoami
302     pwd
303     more /etc/passwd
304      ps -efl
305     cat .profile
306     history -5
martyp $ r
history -5
302     pwd
303     more /etc/passwd
304      ps -efl
305     cat .profile
306     history -5
307     history -5
martyp $
```

You can reissue a specific command number by typing **r**, a space, and the number of the command you wish to reissue. In the following example, I issue the **history** command number 302:

```
martyp $ history
294     history -5
295     whoami
296     pwd
297     more /etc/passwd
298     ps -efl
299     cat /etc/profile
300     history 5
301     whoami
302     pwd
303     more /etc/passwd
304      ps -efl
305     cat .profile
306     history -5
307     history -5
308     history -5
309     history
martyp $ r 302
pwd
/home/martyp
martyp $
```

You can also give the name of the command you wish to reissue. Let's say that you want to reissue the last **fc** command. You would simply give the letter "f" along with the **r** command, and the last command that started with *f* would be reissued as shown in the following example:

```
martyp $ history
308     history -5
309     history
310     pwd
311     whoami
312     pwd
313     cat /etc/profile
314     ls
315     fc -l
316     env
317     who
318     cat /etc/passwd
319     ls -al
320     history
321     more .profile
322     set
323     history
martyp $ r f
fc -l
309     history
310     pwd
311     whoami
312     pwd
313     cat /etc/profile
314     ls
315     fc -l
316     env
317     who
318     cat /etc/passwd
319     ls -al
320     history
321     more .profile
322     set
323     history
324     fc -l
martyp $
```

You can also perform some substitution with the **r** command. If you issued the **vi** command earlier and want to rerun **vi** but specify a different file name to edit, you can substitute a new file name. The following example shows reissuing an earlier **vi** command, except now we'll replace the original filename we edited, **.profile**, with the new filename we want to edit, **.kshrc**. In this example, we'll first run the **history** command, then reissue the **vi** command to edit **.kshrc**, and then we'll run the command again to see our revised **vi** command:

```
martyp $ history
```

```
316     env
317     who
318     cat /etc/passwd
319     ls -al
320     history
321     more .profile
322     set
323     history
324     fc -l
325     vi .profile
326     set
327     ls -al
328     env
329     who
330     more /etc/passwd | grep marty
331     history
martyp $ r vi .profile=.kshrc
martyp $ history
318     cat /etc/passwd
319     ls -al
320     history
321     more .profile
322     set
323     history
324     fc -l
325     vi .profile
326     set
327     ls -al
328     env
329     who
330     more /etc/passwd | grep marty
331     history
332     vi .kshrc
333     history
martyp $
```

This command substitutes **.kshrc** for **.profile** on the command line.

This leads us to more advanced command-line editing. You may want to recall a command earlier issued and edit the command line using **vi** editor commands. The next section covers command-line editing.

Fetching Commands Using vi Directives

You can edit entries in your history list with either the **vi** or **emacs** editor. I will cover the **vi** editor in this chapter, because it is the most widely used on UNIX systems. You can, however, perform all the same functions shown in this chapter with the **emacs** editor.

You edit the history list using **vi** the same way you edit a file using **vi**. You press the *escape* key and use standard **vi** keys for moving up and down, left and right, inserting deleting, and changing text. In this section, I cover fetching a command from the history list using **vi** commands and then I cover editing the command after you have restored it to the command line. After the command has been edited to your liking, you press *enter* to execute the command and place it at the very bottom of the command list.

Your editor is normally set in either your local **.profile** or **/etc/profile**. You can view these files to see what editor has been set for you by your system administrator. When I issue the **env** and **set** commands on my system, both report *EDITOR=vi*. This corresponds with my **/etc/profile** which has the editor set to **vi**.

Let's first take a look at fetching some previously issued commands using **vi**. The examples in this section use **vi** functionality that is described in both the **vi** chapter and on the **vi** quick reference card. The **k, j, G**, and search commands in this section operate in the same manner on the command line as they would if you were using **vi** to edit a file.

Pressing the *escape* key and a command allows you to perform various functions in the **ksh**. The following example shows producing the history list and issuing *escape* **k** to recall the previous command:

```
martyp $ history
125     man fc
126     set | more
127      exit
128     history
129     k
```

```
130     env | more
131     set | more
132     more .profile
133     more .profile | grep rof
134     more .profile | grep vi
135     more /etc/profile | grep vi
136     set |grep edit
137     set | grep vi
138     env | grep vi
139     history
140     who
martyp $ history
```

Issuing *escape* **k** recalled the last command issued, in this case the **who** command at line 140. Each time you press **k**, the preceding **ksh** command is fetched. This is the same as pressing *escape* and the **-** (hyphen) key. You can specify the number of commands you wish to go back in the history list with *escape* *n***k**, where *n* is the number of commands you want to go back in the history list. The following example shows issuing *escape* **5k** to fetch the fifth command back in the history list:




```
martyp $ history
125     man fc
126     set | more
127      exit
128     history
129     k
130     env | more
131     set | more
132     more .profile
133     more .profile | grep rof
134     more .profile | grep vi
135     more /etc/profile | grep vi
136     set |grep edit
137     set | grep vi
138     env | grep vi
139     history
140     who
martyp $ set |grep edit
```

Issuing *escape* **5k** fetched the fifth command back in the history list at line *136*. This is the same as having issued *escape* **5-**.

You can also move forward in the history list, using **j** in the same way we used **k** to move back in the history list. Let's issue *escape* **5k** to move back five commands in the history list to line *136*, and then we can use *escape* **3j** to move forward three commands to line *139*, as shown in the following example:

```
martyp $ history
125     man fc
126     set | more
127      exit
128     history
129     k
130     env | more
131     set | more
132     more .profile
133     more .profile | grep rof
134     more .profile | grep vi
135     more /etc/profile | grep vi
136     set |grep edit
137     set | grep vi
138     env | grep vi
139     history
140     history
martyp $ who
```

Issuing *escape* **3j** moved us forward to line *139*, which is the same as having issued *escape* **3-**.

We can also go back to the oldest command in the history list with *escape* **G**. You can go back to a specific

command number from the history list with *escape n***G**, where *n* is a specific command number from the history file. The following example shows issuing *escape* **126G** to go back to line *126* from the history list:

```
martyp $ history
125     man fc
126     set | more
127      exit
128     history
129     k
130     env | more
131     set | more
132     more .profile
133     more .profile | grep rof
134     more .profile | grep vi
135     more /etc/profile | grep vi
136     set |grep edit
137     set | grep vi
138     env | grep vi
139     history
140     who
martyp $ set | more
```

You can also recall lines from the history list based on searches. If, for instance, you want to recall the most recently issued command from the history list that has in it **profile**, you would issue **/profile**, which is the slash character followed by **profile**, to get the result shown in the following example:

```
martyp $ history
125     man fc
126     set | more
127      exit
128     history
129     k
130     env | more
131     set | more
132     more .profile
133     more .profile | grep rof
134     more .profile | grep vi
135     more /etc/profile | grep vi
136     set |grep edit
137     set | grep vi
138     env | grep vi
139     history
140     who
martyp $ more /etc/profile | grep vi
```

Searching for the most recent occurrence of **profile** produces line *135*. Many of the other searches in **vi** work at the command line. The **vi** chapter and **vi** quick reference card have more searching commands that you may want to try at the command line.

Editing on the Command Line Using vi Directives

After you have "fetched" the command you wish to edit or you have entered a command that needs editing, you can use your **vi** directives to make modifications on the command line.

The **ksh** puts you in input mode immediately upon fetching a command from the history list. This approach is the converse of the **vi** program that you use to edit files, which puts you in control mode by default. After you have fetched a command from the history list, you must press *escape* in order to move into control mode when using **vi** in **ksh**.

Let's now perform some simple modifications to commands we have fetched from the history list. We recall line number 286 from the history list with the sequence *escape* **286G**. After recalling this command, we insert *cat* at the beginning of the command with **i cat. i** is for insert text to the left of the current cursor position. The following example shows the result of both the fetch of line *286* and inserting **cat**:

```
martyp $ history
285     history
286     more /etc/profile | grep vi
287     who
288     whoami
289     who
290     ls -al
```

```
291     alias
292     pwd
293     ls -al /home
294     cat .profile
295     history
296     vi .profile
297     cat /etc/passwd
298     cat /etc/passwd | grep donna
299     pwd
300     history
martyp $ cat more /etc/profile | grep vi
```

You can see that **i cat** inserted *cat* at the very beginning of the line.

We can again recall line *286* and use **A** to append text to the very end of the line, as shown in the following example:

```
martyp $ history
285     history
286     more /etc/profile | grep vi
287     who
288     whoami
289     who
290     ls -al
291     alias
292     pwd
293     ls -al /home
294     cat .profile
295     history
296     vi .profile
297     cat /etc/passwd
298     cat /etc/passwd | grep donna
299     pwd
300     history
martyp $ more /etc/profile | grep vieditor
```



In this example, we used the sequence **A editor** to insert the word *editor* at the very end of the command line we had recalled.

We can recall this line and change *more* at the beginning of the line to *cat* by issuing the sequence **cw cat**, as shown in the following example:

```
martyp $ history
285     history
286     more /etc/profile | grep vi
287     who
288     whoami
289     who
290     ls -al
291     alias
292     pwd
293     ls -al /home
294     cat .profile
295     history
296     vi .profile
297     cat /etc/passwd
298     cat /etc/passwd | grep donna
299     pwd
300     history
martyp $ cat /etc/profile | grep vi
```

This changes *more* to *cat* on the command line.

To delete the word *more* without replacing it, we fetch line 286 and issue **dw**, as shown in the following example:

```
martyp $ history
285     history
286     more /etc/profile | grep vi
```

```
287     who
288     whoami
289     who
290     ls -al
291     alias
292     pwd
293     ls -al /home
294     cat .profile
295     history
296     vi .profile
297     cat /etc/passwd
298     cat /etc/passwd | grep donna
299     pwd
300     history
martyp $  /etc/profile | grep vi
```

If we are unhappy with the most recent command issued, we can undo the command by issuing **u**, which undoes the **dw** we used to remove *more* on the command line, as shown in the following example:




```
martyp $ history
285     history
286     more /etc/profile | grep vi
287     who
288     whoami
289     who
290     ls -al
291     alias
292     pwd
293     ls -al /home
294     cat .profile
295     history
296     vi .profile
297     cat /etc/passwd
298     cat /etc/passwd | grep donna
299     pwd
300     history
martyp $ more /etc/profile | grep vi
```

Issuing **u** puts back the *more* that we had removed with **dw**.

You may want to try using both of the **vi** commands I have covered here, as well as other commands to add, delete, change, search and replace, copy, and undo at the command line.

Aliases in KornShell

An alias is a name that you select for a frequently used command or series of commands. Many aliases are predefined for you.

The **alias** command, without any arguments, lists all aliases. This list includes both preset aliases as well as those you have set. The following command shows the preset aliases on the system on which I am working:

```
martyp $ alias
autoload='typeset -fu'
command='command '
functions='typeset -f'
history='fc -l'
hyper1=HHLIC='/opt/local/bristol/hyperhelp/licenses/hpptc1;export HHLIC'
hyper36=HHLIC='/opt/local/bristol/hyperhelp/licenses/hpptc36;export HHLIC'
hyper83=HHLIC='/opt/local/bristol/hyperhelp/licenses/hpptc83;export HHLIC'
hyper95=HHLIC='/opt/local/bristol/hyperhelp/licenses/hpptc95;export HHLIC'
integer='typeset -i'
local=typeset
nohup='nohup '
r='fc -e -'
```

```
stop='kill -STOP'
suspend='kill -STOP $$'
martyp $
```

Many preset aliases are shown as a result of typing the **alias** command without any options.

Many of these aliases are related to somewhat advanced use of the KornShell, such as job control. Others are useful to you right away. The *history* alias, for instance, lists the commands in the **.sh_history** file and precedes each entry by a line number. The *r* alias allows you to re-run the the last command that appears in the history list.

You are not limited to using only preset aliases. You can set your own aliases. The following example shows setting an alias, producing a list of aliases to see whether indeed our new alias has been set, and then running our alias:

```
martyp $ alias ls="ls -al"
martyp $ alias
autoload='typeset -fu'
command='command '
functions='typeset -f'
history='fc -l'
hyper1=HHLIC='/opt/local/bristol/hyperhelp/licenses/hpptc1;export HHLIC'
hyper36=HHLIC='/opt/local/bristol/hyperhelp/licenses/hpptc36;export HHLIC'
hyper83=HHLIC='/opt/local/bristol/hyperhelp/licenses/hpptc83;export HHLIC'
hyper95=HHLIC='/opt/local/bristol/hyperhelp/licenses/hpptc95;export HHLIC'
integer='typeset -i'
local=typeset
ls='ls -al'
nohup='nohup '
r='fc -e -'
stop='kill -STOP'
suspend='kill -STOP $$'
martyp $ ls
total 26
drwxr-xr-x   2 martyp   staff        512 Mar 15 11:37.
drwxrwxr-x   4 root     sys          512 Mar  4 09:24 ..
-rw-r--r--   1 martyp   staff        124 Mar 15 11:36 .cshrc
-rw-r--r--   1 martyp   staff        562 Mar  4 09:24 .profile
-rw-------   1 martyp   staff       9058 Apr 10 06:58 .sh_history
martyp $
```

The first command set an alias that executes **ls -al** whenever we type **ls**. I issued the **alias** command to see whether indeed the new alias would appear in the list of aliases. It appears right after *local* and right before *nohup* in the alphabetical list of aliases. The final command shows running **ls**, which produces the same listing that you would receive from running the **ls -al** command.

You don't have to keep an alias for the duration of your session after having set it. If you don't like an alias, you can use the **unalias** command to remove an alias.

To see **unalias** work, let's again produce a list of aliases, use **unalias** to unset the *history* alias, run the *history* command to see whether indeed it has been removed, and then run the **fc -l** command to which the *history* alias was mapped:

```
martyp $ alias
autoload='typeset -fu'
command='command '
functions='typeset -f'
history='fc -l'
hyper1=HHLIC='/opt/local/bristol/hyperhelp/licenses/hpptc1;export HHLIC'
hyper36=HHLIC='/opt/local/bristol/hyperhelp/licenses/hpptc36;export HHLIC'
hyper83=HHLIC='/opt/local/bristol/hyperhelp/licenses/hpptc83;export HHLIC'
hyper95=HHLIC='/opt/local/bristol/hyperhelp/licenses/hpptc95;export HHLIC'
integer='typeset -i'
local=typeset
nohup='nohup '
r='fc -e -'
stop='kill -STOP'
suspend='kill -STOP $$'
martyp $ unalias history
martyp $ history
ksh: history:  not found
martyp $ fc -l
131     ps
132     ls -alF
```

```
133     ls -al
134     alias ls="ls -al"
135     alias
136     ls
137     alias
138     unalias ls
139     ls
140     ls -al
141     alias
142     history
143     alias
144     unalias history
145     history
146     fc -l
martyp $
```

When we **unalias** *history* and then run *history,* **ksh** is unable to find it and produces the message *history: not found*. When we run **fc -l**, a list of the most recently issued commands with their corresponding numbers is produced. This makes it clear that when you run the *history* alias, you are actually running the **fc -l** command.

Command and Path Completion

ksh sometimes knows what you're thinking. You can type part of a command or pathname and **ksh** can complete the remainder for you. You can type part of a command or pathname and press the *escape* key to complete the command. If, for instance, you wish to issue the **uname** command to view the current system run level but can't remember the full command, you can type "un" and *escape,* followed by the "=" key, and the command is completed for you. In the following example, we'll change to the **/sbin** directory, list its contents, and then type **un***escape***=**:

```
martyp $ cd /sbin
martyp $ ls -al
total 11704
drwxrwxr-x   2 root     sys          512 Nov 27 14:55.
drwxr-xr-x  31 root     root        1024 Apr 19 03:24 ..
-r-xr-xr-x   1 bin      bin       200356 Sep  1  1998 autopush
lrwxrwxrwx   1 root     root          21 Nov 27 14:55 bpgetfile -> ../usr/sbin/e
-r-xr-xr-x   1 bin      bin       470436 Sep  1  1998 dhcpagent
-r-xr-xr-x   1 bin      bin       433064 Sep  1  1998 dhcpinfo
-r-xr-xr-x   1 bin      bin       253664 Sep  1  1998 fdisk
-r-xr-xr-x   1 bin      bin       762816 Sep  1  1998 hostconfig
-r-xr-xr-x   1 bin      bin       535900 Sep  1  1998 ifconfig
-r-xr-xr-x   1 root     sys       516484 Sep  1  1998 init
-r-xr-xr-x   2 bin      root      257444 Sep  1  1998 jsh
-r-xr-xr-x   1 bin      bin       224596 Sep  1  1998 mount
-r-xr-xr-x   1 root     sys         6935 Jan  1  1970 mountall
-rwxr--r--   3 root     sys         2689 Jan  1  1970 rc0
-rwxr--r--   1 root     sys         2905 Jan  1  1970 rc1
-rwxr--r--   1 root     sys         2491 Jan  1  1970 rc2
-rwxr--r--   1 root     sys         1948 Jan  1  1970 rc3
-rwxr--r--   3 root     sys         2689 Jan  1  1970 rc5
-rwxr--r--   3 root     sys         2689 Jan  1  1970 rc6
-rwxr--r--   1 root     sys         9412 Jan  1  1970 rcS
-r-xr-xr-x   2 bin      root      257444 Sep  1  1998 sh
-r-xr-xr-x   1 bin      bin       195300 Sep  1  1998 soconfig
lrwxrwxrwx   1 root     root          13 Nov 27 14:40 su -> ../usr/bin/su
-r-xr-xr-x   1 root     sys       473808 Sep  1  1998 su.static
-r-xr-xr-x   1 root     bin       288544 Sep  1  1998 sulogin
-rwxr--r--   1 root     sys         3138 Jan  1  1970 swapadd
-r-xr-xr-x   1 bin      bin        29736 Sep  1  1998 sync
-r-xr-xr-x   1 root     sys       435004 Sep  1  1998 uadmin
-r-xr-xr-x   1 bin      bin       213408 Sep  1  1998 umount
-r-xr-xr-x   1 root     sys         3292 Jan  1  1970 umountall
-r-xr-xr-x   1 bin      bin       193152 Sep  1  1998 uname
martyp $ un                                              ;typed unescape=
1) uname
martyp $ un
```

ksh determined that the only command that starts with "un" is **uname**. The **uname** command was listed and *un* was again put at the command prompt for me. You can't see the *escape***=** I typed after the first "un," so I made a comment to the side showing the full command.

Rather than list all files starting with "un," you can replace the current word with all files that match by typing **un***escape****,** as shown in the following example:

```
martyp $ cd /sbin
martyp $ ls -al
total 11704
drwxrwxr-x   2 root     sys          512 Nov 27 14:55.
drwxr-xr-x  31 root     root        1024 Apr 19 03:24 ..
-r-xr-xr-x   1 bin      bin       200356 Sep  1  1998 autopush
lrwxrwxrwx   1 root     root          21 Nov 27 14:55 bpgetfile -> ../usr/sbin/e
-r-xr-xr-x   1 bin      bin       470436 Sep  1  1998 dhcpagent
-r-xr-xr-x   1 bin      bin       433064 Sep  1  1998 dhcpinfo
-r-xr-xr-x   1 bin      bin       253664 Sep  1  1998 fdisk
-r-xr-xr-x   1 bin      bin       762816 Sep  1  1998 hostconfig
-r-xr-xr-x   1 bin      bin       535900 Sep  1  1998 ifconfig
-r-xr-xr-x   1 root     sys       516484 Sep  1  1998 init
-r-xr-xr-x   2 bin      root      257444 Sep  1  1998 jsh
-r-xr-xr-x   1 bin      bin       224596 Sep  1  1998 mount
-r-xr-xr-x   1 root     sys         6935 Jan  1  1970 mountall
-rwxr--r--   3 root     sys         2689 Jan  1  1970 rc0
-rwxr--r--   1 root     sys         2905 Jan  1  1970 rc1
-rwxr--r--   1 root     sys         2491 Jan  1  1970 rc2
-rwxr--r--   1 root     sys         1948 Jan  1  1970 rc3
-rwxr--r--   3 root     sys         2689 Jan  1  1970 rc5
-rwxr--r--   3 root     sys         2689 Jan  1  1970 rc6
-rwxr--r--   1 root     sys         9412 Jan  1  1970 rcS
-r-xr-xr-x   2 bin      root      257444 Sep  1  1998 sh
-r-xr-xr-x   1 bin      bin       195300 Sep  1  1998 soconfig
lrwxrwxrwx   1 root     root          13 Nov 27 14:40 su -> ../usr/bin/su
-r-xr-xr-x   1 root     sys       473808 Sep  1  1998 su.static
-r-xr-xr-x   1 root     bin       288544 Sep  1  1998 sulogin
-rwxr--r--   1 root     sys         3138 Jan  1  1970 swapadd
-r-xr-xr-x   1 bin      bin        29736 Sep  1  1998 sync
-r-xr-xr-x   1 root     sys       435004 Sep  1  1998 uadmin
-r-xr-xr-x   1 bin      bin       213408 Sep  1  1998 umount
-r-xr-xr-x   1 root     sys         3292 Jan  1  1970 umountall
-r-xr-xr-x   1 bin      bin       193152 Sep  1  1998 uname
martyp $ uname                                          ;typed unescape*
```

You can see in this example that **uname** is replaced right at the prompt with the only command that matched **un***escape******.

The next example shows using **um***escape***=** to get a list of all files that start with "um," then using **um***escape****** to get all files matched to "um," and finally, using **um***escape***** to replace the current word with the first file name that starts with "um":

```
martyp $ ls -al
total 11704
drwxrwxr-x   2 root     sys          512 Nov 27 14:55.
drwxr-xr-x  31 root     root        1024 Apr 19 03:24 ..
-r-xr-xr-x   1 bin      bin       200356 Sep  1  1998 autopush
lrwxrwxrwx   1 root     root          21 Nov 27 14:55 bpgetfile -> ../usr/sbin/e
-r-xr-xr-x   1 bin      bin       470436 Sep  1  1998 dhcpagent
-r-xr-xr-x   1 bin      bin       433064 Sep  1  1998 dhcpinfo
-r-xr-xr-x   1 bin      bin       253664 Sep  1  1998 fdisk
-r-xr-xr-x   1 bin      bin       762816 Sep  1  1998 hostconfig
-r-xr-xr-x   1 bin      bin       535900 Sep  1  1998 ifconfig
-r-xr-xr-x   1 root     sys       516484 Sep  1  1998 init
-r-xr-xr-x   2 bin      root      257444 Sep  1  1998 jsh
-r-xr-xr-x   1 bin      bin       224596 Sep  1  1998 mount
-r-xr-xr-x   1 root     sys         6935 Jan  1  1970 mountall
-rwxr--r--   3 root     sys         2689 Jan  1  1970 rc0
-rwxr--r--   1 root     sys         2905 Jan  1  1970 rc1
-rwxr--r--   1 root     sys         2491 Jan  1  1970 rc2
-rwxr--r--   1 root     sys         1948 Jan  1  1970 rc3
-rwxr--r--   3 root     sys         2689 Jan  1  1970 rc5
-rwxr--r--   3 root     sys         2689 Jan  1  1970 rc6
-rwxr--r--   1 root     sys         9412 Jan  1  1970 rcS
-r-xr-xr-x   2 bin      root      257444 Sep  1  1998 sh
-r-xr-xr-x   1 bin      bin       195300 Sep  1  1998 soconfig
lrwxrwxrwx   1 root     root          13 Nov 27 14:40 su -> ../usr/bin/su
-r-xr-xr-x   1 root     sys       473808 Sep  1  1998 su.static
-r-xr-xr-x   1 root     bin       288544 Sep  1  1998 sulogin
-rwxr--r--   1 root     sys         3138 Jan  1  1970 swapadd
```

```
-r-xr-xr-x   1 bin      bin        29736 Sep  1  1998 sync
-r-xr-xr-x   1 root     sys       435004 Sep  1  1998 uadmin
-r-xr-xr-x   1 bin      bin       213408 Sep  1  1998 umount
-r-xr-xr-x   1 root     sys         3292 Jan  1  1970 umountall
-r-xr-xr-x   1 bin      bin       193152 Sep  1  1998 uname
martyp $ um                                              ;typed umescape=
1) umount
2) umountall
martyp $ umount umountall                                 ;typed umescape*
martyp $ umount                                          ;typed umescape\
```

Let's now work with a command and arguments that will be file names. What if the characters you type are not unique, as they were with "un" in the **/sbin** directory? If the command or pathname is not unique, then **ksh** shows you the options for completing the command. The following example shows typing **ls r***escape***=** to get a list of commands that start with "r":

```
martyp $ ls -al
total 11704
drwxrwxr-x   2 root     sys          512 Nov 27 14:55.
drwxr-xr-x  31 root     root        1024 Apr 19 03:24 ..
-r-xr-xr-x   1 bin      bin       200356 Sep  1  1998 autopush
lrwxrwxrwx   1 root     root          21 Nov 27 14:55 bpgetfile -> ../usr/sbin/e
-r-xr-xr-x   1 bin      bin       470436 Sep  1  1998 dhcpagent
-r-xr-xr-x   1 bin      bin       433064 Sep  1  1998 dhcpinfo
-r-xr-xr-x   1 bin      bin       253664 Sep  1  1998 fdisk
-r-xr-xr-x   1 bin      bin       762816 Sep  1  1998 hostconfig
-r-xr-xr-x   1 bin      bin       535900 Sep  1  1998 ifconfig
-r-xr-xr-x   1 root     sys       516484 Sep  1  1998 init
-r-xr-xr-x   2 bin      root      257444 Sep  1  1998 jsh
-r-xr-xr-x   1 bin      bin       224596 Sep  1  1998 mount
-r-xr-xr-x   1 root     sys         6935 Jan  1  1970 mountall
-rwxr--r--   3 root     sys         2689 Jan  1  1970 rc0
-rwxr--r--   1 root     sys         2905 Jan  1  1970 rc1
-rwxr--r--   1 root     sys         2491 Jan  1  1970 rc2
-rwxr--r--   1 root     sys         1948 Jan  1  1970 rc3
-rwxr--r--   3 root     sys         2689 Jan  1  1970 rc5
-rwxr--r--   3 root     sys         2689 Jan  1  1970 rc6
-rwxr--r--   1 root     sys         9412 Jan  1  1970 rcS
-r-xr-xr-x   2 bin      root      257444 Sep  1  1998 sh
-r-xr-xr-x   1 bin      bin       195300 Sep  1  1998 soconfig
lrwxrwxrwx   1 root     root          13 Nov 27 14:40 su -> ../usr/bin/su
-r-xr-xr-x   1 root     sys       473808 Sep  1  1998 su.static
-r-xr-xr-x   1 root     bin       288544 Sep  1  1998 sulogin
-rwxr--r--   1 root     sys         3138 Jan  1  1970 swapadd
-r-xr-xr-x   1 bin      bin        29736 Sep  1  1998 sync
-r-xr-xr-x   1 root     sys       435004 Sep  1  1998 uadmin
-r-xr-xr-x   1 bin      bin       213408 Sep  1  1998 umount
-r-xr-xr-x   1 root     sys         3292 Jan  1  1970 umountall
-r-xr-xr-x   1 bin      bin       193152 Sep  1  1998 uname
martyp $ ls r                                             ;typed ls rescape=
1) rc0
2) rc1
3) rc2
4) rc3
5) rc5
6) rc6
7) rcS
martyp $ ls r
```

You can see from this example that typing **ls -r***escape***=** produced a list of seven files. Again, the **ls r** was placed for me at the next prompt. You can use the information left for you at the next prompt to perform additional **ksh** file name expansion. Let's again perform our **ls r***escape***=,** which produces the list of seven files beginning with *r*. Our next prompt will have **ls r** waiting for us, and this time we'll type *escape****,** which will produce a list of all seven files for us on the line:

```
martyp $ ls -al
total 11704
drwxrwxr-x   2 root     sys          512 Nov 27 14:55.
drwxr-xr-x  31 root     root        1024 Apr 19 03:24 ..
-r-xr-xr-x   1 bin      bin       200356 Sep  1  1998 autopush
lrwxrwxrwx   1 root     root          21 Nov 27 14:55 bpgetfile -> ../usr/sbin/e
-r-xr-xr-x   1 bin      bin       470436 Sep  1  1998 dhcpagent
-r-xr-xr-x   1 bin      bin       433064 Sep  1  1998 dhcpinfo
-r-xr-xr-x   1 bin      bin       253664 Sep  1  1998 fdisk
```

```
-r-xr-xr-x   1 bin      bin       762816 Sep  1  1998 hostconfig
-r-xr-xr-x   1 bin      bin       535900 Sep  1  1998 ifconfig
-r-xr-xr-x   1 root     sys       516484 Sep  1  1998 init
-r-xr-xr-x   2 bin      root      257444 Sep  1  1998 jsh
-r-xr-xr-x   1 bin      bin       224596 Sep  1  1998 mount
-r-xr-xr-x   1 root     sys         6935 Jan  1  1970 mountall
-rwxr--r--   3 root     sys         2689 Jan  1  1970 rc0
-rwxr--r--   1 root     sys         2905 Jan  1  1970 rc1
-rwxr--r--   1 root     sys         2491 Jan  1  1970 rc2
-rwxr--r--   1 root     sys         1948 Jan  1  1970 rc3
-rwxr--r--   3 root     sys         2689 Jan  1  1970 rc5
-rwxr--r--   3 root     sys         2689 Jan  1  1970 rc6
-rwxr--r--   1 root     sys         9412 Jan  1  1970 rcS
-r-xr-xr-x   2 bin      root      257444 Sep  1  1998 sh
-r-xr-xr-x   1 bin      bin       195300 Sep  1  1998 soconfig
lrwxrwxrwx   1 root     root          13 Nov 27 14:40 su -> ../usr/bin/su
-r-xr-xr-x   1 root     sys       473808 Sep  1  1998 su.static
-r-xr-xr-x   1 root     bin       288544 Sep  1  1998 sulogin
-rwxr--r--   1 root     sys         3138 Jan  1  1970 swapadd
-r-xr-xr-x   1 bin      bin        29736 Sep  1  1998 sync
-r-xr-xr-x   1 root     sys       435004 Sep  1  1998 uadmin
-r-xr-xr-x   1 bin      bin       213408 Sep  1  1998 umount
-r-xr-xr-x   1 root     sys         3292 Jan  1  1970 umountall
-r-xr-xr-x   1 bin      bin       193152 Sep  1  1998 uname
martyp $ ls r                                          ;typed ls rescape=
1) rc0
2) rc1
3) rc2
4) rc3
5) rc5
6) rc6
7) rcS
martyp $ ls rc0 rc1 rc2 rc3 rc5 rc6 rcS
                                    ;typed escape* to get seven files
```

escape* listed all seven files beginning with r and placed them on the command line.

In the next example we'll issue both the escape= and escape*. We'll then type **ls** escape_ (underscore), which puts the last word of the last command in the line, which is rcS:

```
martyp $ ls -al
total 11704
drwxrwxr-x   2 root     sys          512 Nov 27 14:55.
drwxr-xr-x  31 root     root        1024 Apr 19 03:24 ..
-r-xr-xr-x   1 bin      bin       200356 Sep  1  1998 autopush
lrwxrwxrwx   1 root     root          21 Nov 27 14:55 bpgetfile -> ../usr/sbin/e
-r-xr-xr-x   1 bin      bin       470436 Sep  1  1998 dhcpagent
-r-xr-xr-x   1 bin      bin       433064 Sep  1  1998 dhcpinfo
-r-xr-xr-x   1 bin      bin       253664 Sep  1  1998 fdisk
-r-xr-xr-x   1 bin      bin       762816 Sep  1  1998 hostconfig
-r-xr-xr-x   1 bin      bin       535900 Sep  1  1998 ifconfig
-r-xr-xr-x   1 root     sys       516484 Sep  1  1998 init
-r-xr-xr-x   2 bin      root      257444 Sep  1  1998 jsh
-r-xr-xr-x   1 bin      bin       224596 Sep  1  1998 mount
-r-xr-xr-x   1 root     sys         6935 Jan  1  1970 mountall
-rwxr--r--   3 root     sys         2689 Jan  1  1970 rc0
-rwxr--r--   1 root     sys         2905 Jan  1  1970 rc1
-rwxr--r--   1 root     sys         2491 Jan  1  1970 rc2
-rwxr--r--   1 root     sys         1948 Jan  1  1970 rc3
-rwxr--r--   3 root     sys         2689 Jan  1  1970 rc5
-rwxr--r--   3 root     sys         2689 Jan  1  1970 rc6
-rwxr--r--   1 root     sys         9412 Jan  1  1970 rcS
-r-xr-xr-x   2 bin      root      257444 Sep  1  1998 sh
-r-xr-xr-x   1 bin      bin       195300 Sep  1  1998 soconfig
lrwxrwxrwx   1 root     root          13 Nov 27 14:40 su -> ../usr/bin/su
-r-xr-xr-x   1 root     sys       473808 Sep  1  1998 su.static
-r-xr-xr-x   1 root     bin       288544 Sep  1  1998 sulogin
-rwxr--r--   1 root     sys         3138 Jan  1  1970 swapadd
-r-xr-xr-x   1 bin      bin        29736 Sep  1  1998 sync
-r-xr-xr-x   1 root     sys       435004 Sep  1  1998 uadmin
-r-xr-xr-x   1 bin      bin       213408 Sep  1  1998 umount
-r-xr-xr-x   1 root     sys         3292 Jan  1  1970 umountall
-r-xr-xr-x   1 bin      bin       193152 Sep  1  1998 uname
martyp $ ls r                                       ;typed ls rescape=
1) rc0
```

```
2) rc1
3) rc2
4) rc3
5) rc5
6) rc6
7) rcS
martyp $ ls rc0 rc1 rc2 rc3 rc5 rc6 rcS          ;typed escape* to get seven files
rc0  rc1  rc2  rc3  rc5  rc6  rcS
martyp $ ls  rcS                                 ;typed lsescape_ to get last word
rcS
martyp $
```

Now that we've seen how to get the last of the words to appear on the command line with *escape_*, how about the third word of the last command? You can specify the third word by typing *escape**3**_* or substitute for the "3" for any of the words on the last command line. The following example shows listing the third word of the last command:

```
martyp $ ls -al
total 11704
drwxrwxr-x   2 root     sys          512 Nov 27 14:55.
drwxr-xr-x  31 root     root        1024 Apr 19 03:24 ..
-r-xr-xr-x   1 bin      bin       200356 Sep  1  1998 autopush
lrwxrwxrwx   1 root     root          21 Nov 27 14:55 bpgetfile -> ../usr/sbin/e
-r-xr-xr-x   1 bin      bin       470436 Sep  1  1998 dhcpagent
-r-xr-xr-x   1 bin      bin       433064 Sep  1  1998 dhcpinfo
-r-xr-xr-x   1 bin      bin       253664 Sep  1  1998 fdisk
-r-xr-xr-x   1 bin      bin       762816 Sep  1  1998 hostconfig
-r-xr-xr-x   1 bin      bin       535900 Sep  1  1998 ifconfig
-r-xr-xr-x   1 root     sys       516484 Sep  1  1998 init
-r-xr-xr-x   2 bin      root      257444 Sep  1  1998 jsh
-r-xr-xr-x   1 bin      bin       224596 Sep  1  1998 mount
-r-xr-xr-x   1 root     sys         6935 Jan  1  1970 mountall
-rwxr--r--   3 root     sys         2689 Jan  1  1970 rc0
-rwxr--r--   1 root     sys         2905 Jan  1  1970 rc1
-rwxr--r--   1 root     sys         2491 Jan  1  1970 rc2
-rwxr--r--   1 root     sys         1948 Jan  1  1970 rc3
-rwxr--r--   3 root     sys         2689 Jan  1  1970 rc5
-rwxr--r--   3 root     sys         2689 Jan  1  1970 rc6
-rwxr--r--   1 root     sys         9412 Jan  1  1970 rcS
-r-xr-xr-x   2 bin      root      257444 Sep  1  1998 sh
-r-xr-xr-x   1 bin      bin       195300 Sep  1  1998 soconfig
lrwxrwxrwx   1 root     root          13 Nov 27 14:40 su -> ../usr/bin/su
-r-xr-xr-x   1 root     sys       473808 Sep  1  1998 su.static
-r-xr-xr-x   1 root     bin       288544 Sep  1  1998 sulogin
-rwxr--r--   1 root     sys         3138 Jan  1  1970 swapadd
-r-xr-xr-x   1 bin      bin        29736 Sep  1  1998 sync
-r-xr-xr-x   1 root     sys       435004 Sep  1  1998 uadmin
-r-xr-xr-x   1 bin      bin       213408 Sep  1  1998 umount
-r-xr-xr-x   1 root     sys         3292 Jan  1  1970 umountall
-r-xr-xr-x   1 bin      bin       193152 Sep  1  1998 uname
martyp $ ls r                                     ;typed ls rescape=
1) rc0
2) rc1
3) rc2
4) rc3
5) rc5
6) rc6
7) rcS
martyp $ ls rc0 rc1 rc2 rc3 rc5 rc6 rcS          ;typed escape* to get seven files
rc0  rc1  rc2  rc3  rc5  rc6  rcS
martyp $ ls  rc1                                ;typed lsescape3_ to get third word
```

Because the previous command included the **ls** command, the third word is not **rc2** but **rc1**, because **ls** was the first entry in the previous command.

The Table 27-1 summarizes the command and path completion used in the previous examples:

Table 27-1. Command and Path Completion in ksh

Word or Command	

(type *escape* before each)	Result
word*escape***=**	Displays a numbered list of file names beginning with **word**.
word*escape******	Replaces **word** with all files matched.
word*escape*****	Replaces **word** with the first file name that starts with **word**.
command word*escape***=**	Displays a numbered list of file names beginning with **word**.
command word*escape******	Replaces **word** with all files matched.
command word*escape***_**	Inserts the last word of the last command at the cursor position.
command word*escape***_*3**	Inserts the third word of the last command at the cursor position.

File Name Expansion

In your general activities working with **ksh**, you have to perform a lot of file-name-related work, including crafting shell scripts that deal with file names. An overview of file name expansion is useful to ensure that you're comfortable with this topic before you start writing shell scripts.

Table 27-2 lists some common filename expansion and pattern matching commands:

Table 27-2. File Name Expansion and Pattern Matching

Character(s)	Example	Description
*	1) **ls *.c**	Match zero or more characters
?	2) **ls conf.?**	Match any single character
[list]	3) **ls conf.[co]**	Match any character in list
[lower-upper]	4) **ls libdd.9873[5-6].sl**	Match any character in range
str{str1,str2,str3,...}	5) **ls ux*.{700,300}**	Expand str with contents of {}
~	6) **ls -a ~**	Home directory
~username	7) **ls -a ~gene**	Home directory of username

The following are more detailed descriptions of the examples shown in Table 27-2:

1. To list all files in a directory that end in ".c," you could do the following:

   $ **ls *.c**
   conf. SAM.c conf.c

2. To find all the files in a directory named "conf" with an extension of one character, you could do the following:

   $ **ls conf.?**
   conf.c conf.o conf.1

3. To list all the files in a directory named "conf" with only the extension "c" or "o," you could do the following:

   $ **ls conf.{co}**
   conf.c conf.o

4. To list files with similar names but with a specific field that covers a range, you could do the following:

```
$ ls libdd9873[5-6].sl
      libdd98735.sl libdd98736.sl
```

5. To list files that start with "ux" and have the extension "300" or "700," you could do the following:

```
$ ls ux*.{700,300}
      uxbootlf.700 uxinstfs.300
```

6. To list the files in your home directory, you could use ~:

```
$ ls -a ~
      .    .cshrc.org .login    .shrc.org
      ..   .exrc    .login.org    .cshrc
      .history
```

7. To list the files in the home directory of a user, you can do the following:

```
$ ls -a ~gene
      .      .history   splinedat under.des
      ..     .login    trail.txt xtra.part
      .chsrc   .login.org ESP-File
      .cshrc.org .profile Mail
      .exrc    .shrc.org opt
```

Many of these techniques are useful when working with **ksh** and writing shell scripts, so you want to become familiar with file name expansion.

Redirection (I/O Redirection)

UNIX is set up such that commands usually take their input from the keyboard, often called standard input, and usually send output to the screen, often called *standard output*. Commands also send error information to the screen. You do not always want input to come from standard input and output and errors to go to standard output. You are given a lot of control to override these defaults. This is called *redirection*. Table 27-3 shows many common forms of redirection.

As shown in Table 27-3, to redirect the output of a command from standard output to a file, you use ">". This works almost all of the time. If you have an environment variable called **noclobber** set, then redirecting to an existing file does not work (we'll cover environment variables shortly). The **noclobber** does not permit redirection to write over an existing file. If you try to write over an existing file, such as **/tmp/processes** below, you receive a message that the file exists:

```
# ps -ef > /tmp/processes
/tmp/processes: File exists
```

You can, however, use a "!" with redirection to force a file to be overwritten. Using ">!" forces a file to be overwritten, and ">>!" forces the output to be appended to the end of an existing file. Examples of these are shown in Table 27-3.

Table 27-3. Commonly Used Redirection Forms

Command or Assignment	Example	Description
<	**wc -l < .login**	Standard input redirection: execute **wc** (word count) and list number of lines (**-l**) in **.login**
>	**ps -ef > /tmp/processes**	Standard output redirection: execute **ps** and send output to file **/tmp/processes**

>>	**ps -ef >> /tmp/processes**	Append standard output: execute **ps** and append output to the end of file **/tmp/processes**
>!	**ps -ef >! /tmp/processes**	Standard output redirection and override **noclobber**: write over **/tmp/processes** even if it exists
>>!	**ps -ef >>! /tmp/processes**	Append standard output and override **noclobber**: append to the end of **/tmp/processes**
| (pipe)	**ps | wc -l**	Run **ps** and use the result as input to **wc**
0 - standard input		
1 - standard output		
2 - standard error	**cat program 2> errors**	**cat** the file **program** to standard output and redirect errors to the file **errors**
	cat program 2>> errors	**cat** the file **program** to standard output and append errors to the file **errors**
	find / -name '*.c' -print > cprograms 2>errors	**find** all files on the system ending in **.c**, place the list of files in **cprograms** in the current working directory, and send all errors (file descriptor 2) to the file **errors** in current working directory
	find / -name '*.c' -print > cprograms 2>&1	**find** all files on the system ending in **.c**, place the list of files in **cprograms** in the current working directory, and send all errors (file descriptor 2) to same place as file descriptor 1 (**cprograms**)

Environment Variables

An environment variable is a name associated with a string. The name is the variable and the string is its value. Environment variables are also available to sub-shells or processes spawned by the shell. In most cases, you see environment variables capitalized, which is the convention on most systems.

When you issue a command on the system, you usually enter a relative pathname, not an absolute pathname. The command you issue is found because the *PATH* variable points to the location of directories where commands are located. Without this *PATH* variable, you would have to type the absolute pathname of every command you issue. When you issue a command, the shell searches the directories listed in the *PATH* variable in the order in which they are listed. A good way to see many of the environment variables you have set is with the **env** command as shown below:

```
martyp $ env
_=/usr/bin/env
MANPATH=:/opt/local/ptc/sysadmin/scripts:/opt/local/ptc/man:/opt/local/altrasofn
_INIT_UTS_RELEASE=5.7
HZ=100
_INIT_UTS_MACHINE=sun4m
EPC=true
PATH=/usr/bin:/usr/ucb:/etc:.
WEB_SERVER=sioux.rose.hp.com
_INIT_UTS_VERSION=Generic
MODEL=SPARCstation-10
OS_REV=5.7
EDITOR=vi
_INIT_RUN_NPREV=0
CLASSPATH=.:/usr/java/lib:
LOGNAME=martyp
_INIT_UTS_NODENAME=sunsys
_INIT_UTS_ISA=sparc
MAIL=/var/mail/martyp
ERASE=^H
OS=solaris
```

```
PS1=$PWD
$LOGNAME $TOKEN
_INIT_PREV_LEVEL=S
HOST=sunsys
TESTEXPERT_HOME=/opt/local/svn/te33
TA_HOME=/opt/local/platinum/solaris2/testadvise
MA_HOME=/opt/local/platinum/solaris2/memadvise
CL_LICENSE_FILE=/opt/local/CenterLine/configs/license.dat
SHELL=/bin/ksh
PROFILE_DIR=/opt/local/ptc/sysadmin/profile.d
OSTYPE=solaris2
HOME=/home/martyp
_INIT_UTS_SYSNAME=SunOS
TERM=vt100
LD_LIBRARY_PATH=:/opt/local/parasoft/lib.solaris
MWHOME=/opt/local/mainsoft/mainwin/mw
FMHOME=/opt/local/adobe/frame
PWD=/home/martyp
TZ=US/Pacific
_INIT_RUN_LEVEL=3
CLEARCASE_BLD_UMASK=02
_INIT_UTS_PLATFORM=SUNW,SPARCstation-10
martyp $
```

As you can see, other environment variables in addition to *PATH* make working on your UNIX system easier. You may want to issue other commmands related to **ksh** variables. Table 27-4 summarizes some **ksh**-related variable commands you may want to issue on your system.

Table 27-4. ksh-Related Variable Commands

Command	Description
env	Lists all environment variables (exported). These variables are normally uppercase and passed to child processes.
set	Prints all local and exported set variables.
set -o	Lists all built-in variables that are set to *on* or *off*.
typeset	Displays all variables and associated attributes, functions, and integers.
typeset +	Displays only the names of variables.

You may want to try all these commands on your system to see what variables have been set for you.

If you want to know the value of a specific environment variable, you could use the **echo** command to see its value, as shown below for the environment variable *HOME*:

```
martyp $ echo $HOME
/home/martyp
```

Similarly, to view the operating system type on this specific computer, you could issue the following command:

```
martyp $ echo $OSTYPE
solaris2
```

The "$" preceding the environment variable name specifies that the value of the variable be sent to standard output. In this case, the value of the environment variable *HOME* is **/home/martyp**, which means that the current user has a home directory of **/home/martyp**.

You can define your own environment variables in **ksh** with the following syntax:

```
export NAME=value
```

Many users like to customize the **ksh** prompts. Most systems provide four **ksh** prompts by default. You can normally modify the first two prompts. The first, called **PS1**, is the primary prompt. The second, called **PS2**, appears after you have partially typed a command and pressed *enter*. **PS1** is normally set to a *$* and **PS2** to *>*, by default.

The following example sets **PS1** to our home directory, a space, the current history number, a space, and dollar sign:

```
PS1="`pwd` ! $ "
```

We'll leave the default for **PS2** as >. The following sequence shows issuing a command at our new **PS1** prompt and the default **PS2** prompt:

```
/home/martyp 187 $ print "This is new PS1
> but default PS2"
This is new PS1
but default PS2
/home/martyp 188 $
```

This example shows the new **PS1**, including the history entry incrementing from *187* to *188*, and the default **PS2** when the incomplete **print** command was issued.

You can also append to the end of an existing variable with the following format:

```
export NAME="$NAME:appended_information"
```

To add **/home/martyp/programs** to the existing *PATH* environment variable, for instance, you issue the following command:

```
export PATH="$PATH:/home/martyp/programs"
```

This appends the path **/home/martyp/programs** to the environment variable *PATH*.

A great deal of flexibility is available when working with **ksh**. You should view all your variables and update those that make your job easier. Some customization, such as updating **PS1** and your **PATH**, can also make your job easier.

Background Jobs and Job Control

When you run a command, as we have done so far in many examples, you don't get back the prompt until the command has completed. These commands have been run in the foreground. Some commands can take a long time to complete, in which case, you'll be waiting a long time for the prompt to return. As an alternative to waiting for the prompt to return, you can run the command in the background, meaning that it is running behind the scenes while you perform other work. Because UNIX is multi-tasking it is happy to run many commands in the background and still provide you with a prompt to issue yet more commands.

In order to run a command in the background, you simply add an ampersand (&) to the end of the command line. When you follow your command with an ampersand, the command is run in the background and the prompt is immediately returned.

Let's now run some commands on our Solaris system. We'll first run a command to find all of the files in **/usr** that end in ".c," which takes some time to complete. We'll preface the **find** string with the **time** command so that we know how long the command takes to complete:




```
martyp $ time find /usr -name *.c
            .
            .
            .
/usr/demo/link_audit/src/dumpbind.c
/usr/demo/link_audit/src/env.c
/usr/demo/link_audit/src/hash.c
/usr/demo/link_audit/src/perfcnt.c
/usr/demo/link_audit/src/symbindrep.c
/usr/demo/link_audit/src/truss.c
/usr/demo/link_audit/src/who.c
find: cannot read dir /usr/aset: Permission denied

real    1m21.04s
user    0m1.51s
```

```
sys     0m12.67s
```

This commad took roughly one minute and 21 seconds to complete. Because it was run in the foreground we were unable to issue any other commands while it was running, because we had to wait for the prompt to return.

An alternative to running the command in the foreground is to issue the command followed by an ampersand, in which case the job runs in the background and the prompt returns immediately, as shown in the following example:

```
martyp $ time find /usr -name *.c > cprogs 2>&1 &
[3]     16279
martyp $
real    2m10.20s
user    0m1.31s
sys     0m8.62s
```

The result of running this command in the background produces a job number in brackets and the process id, or PID, as the second number. All the outputs of this command, including errors, are written to the file **cprogs**. The prompt was immediately returned after we issued the command, and after it completed, the output of **time** was sent to the screen. We could have begun issuing additional commands immediately after issuing the command in the background.

You have control over both foreground jobs and background jobs. To suspend a foreground job, you type the "control" and "z" keys simultaneously, as shown in the following example:




```
martyp $ find /usr -name *.c
/usr/openwin/share/include/X11/Xaw/Template.c
/usr/openwin/share/src/dig_samples/DnD/main.c
/usr/openwin/share/src/dig_samples/DnD/owner.c
/usr/openwin/share/src/dig_samples/DnD/requestor.c
/usr/openwin/share/src/dig_samples/Tooltalk/olit_tt.c
/usr/openwin/share/src/dig_samples/Tooltalk/tt_callbacks.c
/usr/openwin/share/src/dig_samples/Tooltalk/tt_code.c
/usr/openwin/share/src/dig_samples/ce1/ce_map1.c
/usr/openwin/share/src/dig_samples/ce2/ce_simple.c
/usr/openwin/share/src/dig_samples/dnd_olit/olitdnd.c
/usr/openwin/share/src/dig_samples/dnd_xview1/xview_dnd.c
/usr/openwin/share/src/dig_samples/dnd_xview2/
xview_dnd2.c
/usr/openwin/share/src/dig_samples/selection_olit/olit_sel.c
/usr/openwin/share/src/dig_samples/tooltalk_simple/ttsend.c
/usr/openwin/share/src/olit/oldials/oldials.c
/usr/openwin/share/src/olit/olitbook/ch10/draw.c
^Z[3] + Stopped (SIGTSTP)        find /usr -name *.c
```

After *ctrl-z* is pressed, the **find** command is interrupted at the point at which you type *ctrl-z.* The command is suspended at this point, and you are shown the job number, in this case "3," and its status is listed as "Stopped". This command has only been suspended; it is not gone forever. You can start this process in the foreground with **fg**, or run it in the background with **bg**. Using **bg** runs the command as if you had followed it with an "&". It is started from the point at which you interrupted it. You do not have to supply a job number when you issue the **fg** or **bg** command, because the default is to perform the specified operation on the last job, which in this case is job number *3*.

Notice that in this example we have stopped job number *3*. This means that there are other jobs running with lower job numbers. You can use the **jobs** command to get a list of all jobs and their status. You can then control the jobs by issuing commands such as **fg** followed by a "%" and the job number to run the command in the foreground, or a **bg** followed by a "%" and the job number to run the command in the background. If you wish to terminate a job altogether, you can issue the **kill** command followed by a "%" and the job number.

In the process of creating the examples in this section, I have started and suspended many jobs. The following example shows listing all jobs with the **jobs** command, killing jobs *1* and *2* with **kill**, and running job 3 in the background:

```
martyp $ jobs
```

```
[3] + Stopped (SIGTSTP) find /usr -name *.c
[2] - Running         time find / -name gnu* > gnu 2>&1 &
[1]   Running            time find / -name *.c > cprogs 2>&1 &
martyp $ kill %1
[1]   Terminated         time find / -name *.c > cprogs 2>&1 &
martyp $ kill %2
[2] - Terminated      time find / -name gnu* > gnu 2>&1 &
martyp $ bg %3
[3]     find /usr -name *.c&
martyp $
```

Notice that an ampersand was added to job *3* when we requested that it be run in the background. Killing jobs 1 and 2 and running job 3 in the background returns the prompt so that you can perform additional work.

umask and Permissions

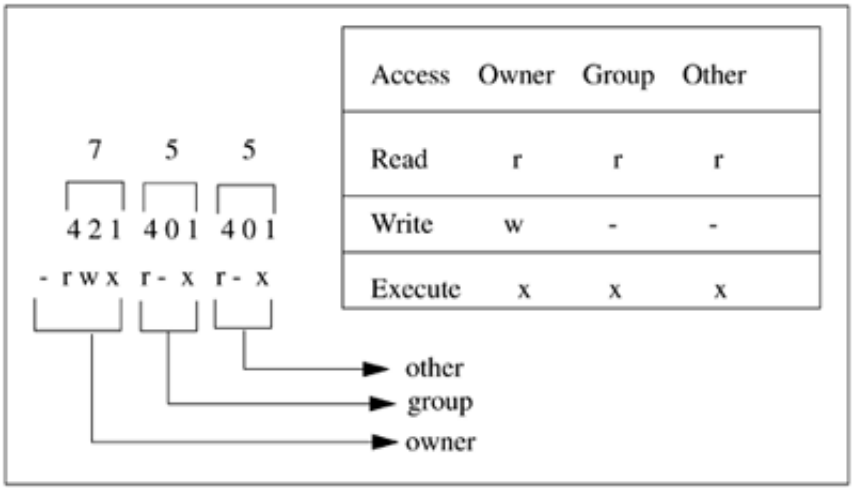
An additional topic to cover related to **ksh** is file permissions and the way they relate to **umask**. This is important because you may write shell programs, and the permissions control the access that others will have to these programs. **umask** is used to specify permission settings for new files and directories.

Let's start with an example of a long listing of a file. We'll use **ls -l** in the following example:

```
sys1 1: ls -l script1
-rwxr-xr-x  1  marty  users  120 Jul 26 10:20 script1
```

The access rights for this file are defined by the position of read (r), write (w), and execute (x) when the **ls -l** command is issued. Figure 27-1 shows the three groups of three access rights for this file:

Figure 27-1. Example of File Permissions



The owner of this file has read, write, and execute permissions on the file. The group to which the user belongs has read and execute permissions, and others also have read and execute permissions. The permissions on this file can be specified by the octal sum of each field, which is 755.

What happens if you craft a new shell script or any new file? What permission settings exist? You want to execute the shell script, so you need execute permission for the file. You can use **umask** to define the defaults for all your new files and directories.

By default, most systems start with a permission of 777 for directories and 666 for files. These mean that everyone has complete access to all directories you create and everyone has read and write access to all files you create. These defaults are modified with the value of **umask**.

You can view your **umask** in the following two ways:

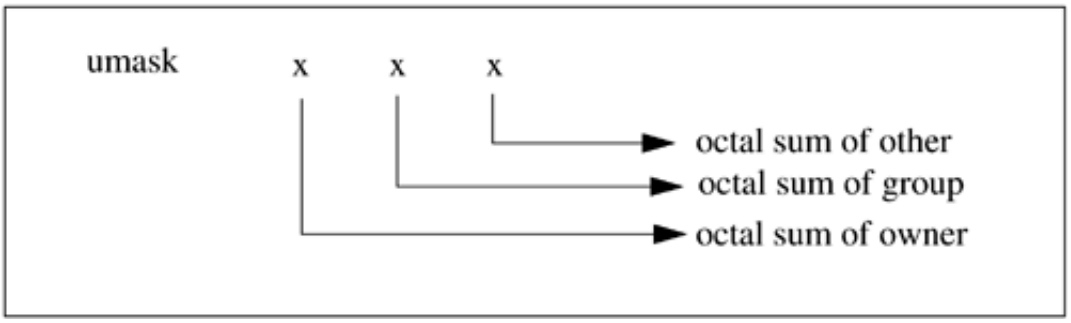
```
martyp $ umask
002
```

```
martyp $ umask -S
u=rwx,g=rwx,o=rx
martyp $
```

The first example displays the octal value of **umask**, which we'll cover shortly, and the second example shows the symbolic value of **umask**.

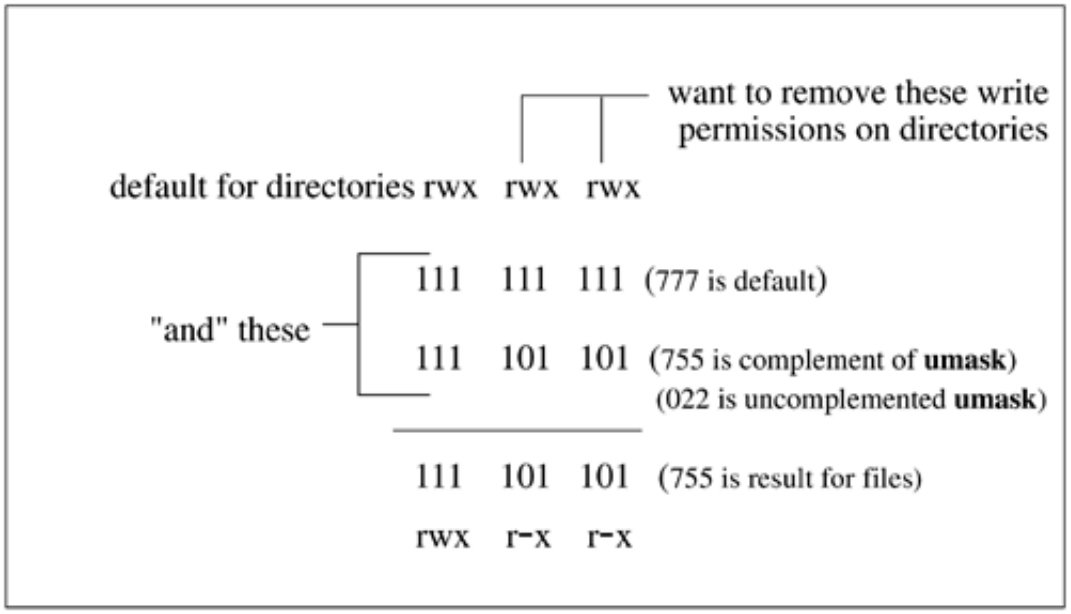
The **umask** is used to *disable* access. You start with a **umask** and use the fields to disable some level of access. The **umask** command uses three octal fields. The fields are the sum of the access codes for user, group, and other as shown in Figure 27-2:

Figure 27-2. umask Fields



The *complement* of the umask field is "*anded*" with the default setting to change the **umask**. You can set **umask** by specifying its value. In our earlier example we viewed **umask** two different ways. To set the **umask**, you simply issue **umask** and the desired value. Setting **umask** to *022*, for example, removes write permissions of directories for "group" and "other," as shown in Figure 27-3:

Figure 27-3. umask Example



umask 022 changes the directory permissions to 755 in this example.

Similarly, a **umask** of *022* changes the default permission of 666 for files to 644, which would be read-only for group and other.

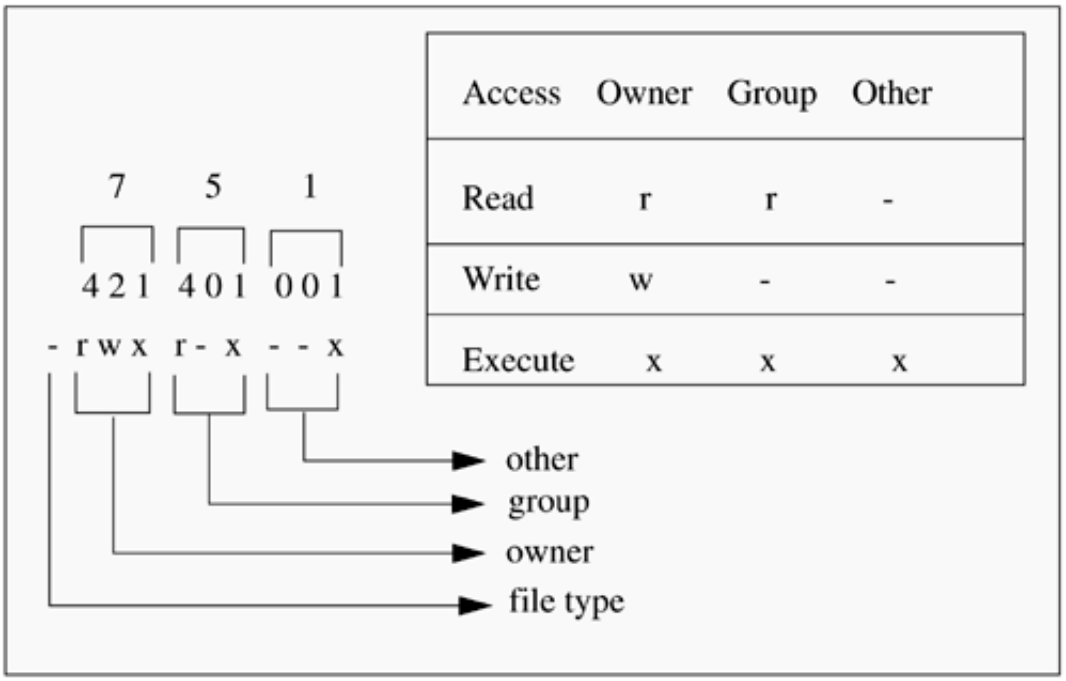
Change File Permissions with chmod

The **chmod** command is used to change the permissions on a file. Irrespective of what takes place with **umask** as just described, you can change a file's permissions at any time with **chmod**. You need to be the owner of the file or superuser to change a file's permissions with **chmod** in most cases. Let's start our discussion of **chmod** with the listing of the file **sort**:

```
$ ls -l sort
-rwxr-x--x   1 marty     users     120 Jul 26 10:20 sort
```

Figure 27-4 shows a breakdown of the permissions on **sort**:

Figure 27-4. Permissions of File sort

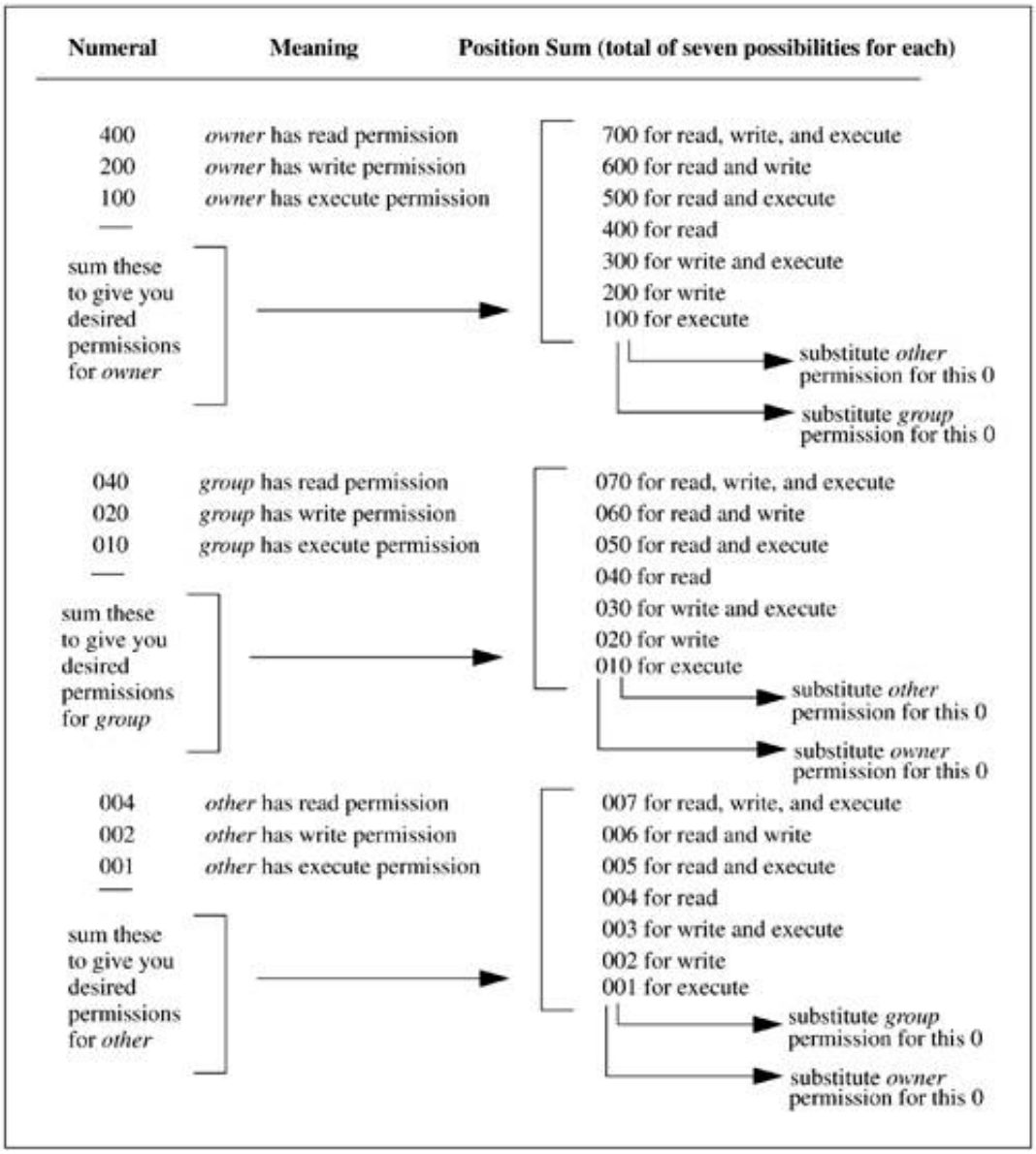


You have very little control over the type of file defined. You do, however, have a great deal of control over the permissions of this file if it belongs to you. The **chmod** command is used to change the permissions on a file or directory. If you are the owner of the file, you can have a field day changing the permissions on the file.

Two means exist by which you can change the permissions: symbolic and numeric. I focus first on the numeric mode, because the numbers involved are easy to manage, and I sometimes find that new UNIX users get hung up on the meanings of some of the symbols. I'll then cover the symbols and include the symbol meanings in the **chmod** summary.

First of all, what do I mean by numbers? Looking at the numbers for **sort**, we see permissions of 751: 7 for *owner* (hundred's position), 5 for *group* (ten's position), and 1 for *other* (one's position). Figure 27-5 helps with the meanings of the positions:

Figure 27-5. Numerical Permissions Summary



Selecting the desired permissions for *owner, group*, and *other,* you use the **chmod** command to assign those permissions to a file or directory. Some of these permission possibilities are infrequently used, such as execute only, because you usually need to have read access to a file in order to execute it; however, I included all possibilities in Figure 27-5 for completeness. In addition to the permission mode bits shown in Figure 27-5, there are also miscellaneous mode bits, which you don't need to be concerned with at this time.

If you decide that you would like to add write permission of the file **sort** for *group*, and remove all permissions for *other*, you would simply execute the **chmod** command with the appropriate numeric value. The following set of commands first list the existing permissions for **sort**, next change the permissions on **sort**, and finally, list the new permissions on **sort**:

```
$ ls -l sort
-rwxr-x--x   1 marty     users     120 Jul 26 10:20 sort

$ chmod 770 sort

$ ls -l sort
-rwxrwx---   1 marty     users     120 Jul 26 10:20 sort
```

The same set of commands to change the permissions using the symbolic mode would be:

```
$ ls -l sort
```

```
-rwxr-x--x   1 marty     users     120 Jul 26 10:20 sort

$ chmod g+w,o-x sort

$ ls -l sort
-rwxrwx---   1 marty     users     120 Jul 26 10:20 sort
```

In symbolic mode, you issue the **chmod** command and specify who will be affected by the change [user (u), group (g), other (o), or all (a)], the operation you wish to perform [add (+), delete (-), or replace (=)], and the permission you wish to specify [read (r), write (w), or execute (x)]. In the previous example using symbolic mode, write (w) permission is being added (+) for *group* (g), and execute (x) permission is being removed (-) for *other* (o).

The following is a summary of some of the more commonly used symbols of **chmod**:

chmod - Change permissions of specified files using the following symbolic mode list.

Symbol of who is affected:

u	User is affected.
g	Group is affected.
o	Other is affected.
a	All users are affected.

Operation to perform:

+	Add permission.
-	Remove permission.
=	Replace permission.

Permission specified:

r	Read permission.
w	Write permission.
x	Execute permission.
u	Copy user permissions.
g	Copy group permissions.
o	Copy other permissions.

Introduction to the C Shell

The C shell is similar to other shells in that it provides a user interface to UNIX. You can use the C shell in the following three ways:

- Interactively type commands on the command line.
- Group commonly executed sets of commands into command files that you can execute by typing the name of the file.
- Create C shell programs using the structured programming techniques of the C shell.

These three techniques are listed in the order in which you'll probably use them. First, you log in and use interactive commands. Then you group together commonly used commands and execute them with a single command. Finally, you may want to create sophisticated shell scripts.

In this chapter, I cover login and interactive commands and a lot of useful ways to use the C shell.

Most of the examples in this chapter are from Solaris and HP-UX systems. You probably will find both your user setup and the operation of the C shell on other systems similar to what is covered in this chapter. Much of the setup of any shell is performed by the system administrator, so you will surely find differences in your C shell setup compared with what is shown in this chapter. In general, however, the operation of the C shell is similar from one system to another.

Issuing Commands

The first activity you perform after you log in to the system is to issue commands at the prompt. A command you may want to issue immediately is **ls -al**. Here is what I see on my system after executing this:

```
sys1 7: ls -al
total 10
drwxr-x---   2 martyp2    users         96 May  5 09:34 .
drwxr-xr-x  10 root       root        1024 May  5 10:38 ..
-rw-r--r--   1 martyp2    users        814 May  5 09:34 .cshrc
-rw-r--r--   1 martyp2    users        347 May  5 09:34 .exrc
-rw-r--r--   1 martyp2    users        341 May  5 09:34 .login
-rw-r--r--   1 martyp2    users       446 May  5 09:34 .profile
sys1 8:
```

The C shell prompt consists of system name (sys1) followed by the command number and a colon. I cover the prompt shortly.

ls -al shows two files related to the C shell in this user area:

.cshrc and **.login**

Figure 27-6 shows the contents of **.cshrc**:

Figure 27-6 Sample .cshrc

```
# Default user .cshrc file (/usr/bin/csh initialization).

# Usage:  Copy this file to a user's home directory and edit it to
# customize it to taste.  It is run by csh each time it starts up.

# Set up default command search path:
#
# (For security, this default is a minimal set.)
```

```
     set path=( $path )

# Set up C shell environment:

    if ( $?prompt ) then        # shell is interactive.
        set history=20         # previous commands to remember.
        set savehist=20        # number to save across sessions.
        set system=`hostname`      # name of this system.
        set prompt = "$system \!: " # command prompt.

        # Sample alias:

        alias       status  '(date; bdf)'

        # More sample aliases:

        alias  d    dirs
        alias  pd   pushd
        alias  pd2  pushd +2
        alias  po   popda
        alias  m    more
     endif
```

Figure 27-7 shows the contents of **.login**:

Figure 27-7 Sample .login

```
# @(#) $Revision: 72.3 $


# Default user .login file ( /usr/bin/csh initialization )

# Set up the default search paths:
set path=( $path )

#set up the terminal
eval `tset -s -Q -m ':?hp' `
stty erase "^H" kill "^U" intr "^C" eof "^D" susp "^Z" hupcl ixon ixoff
tostop
tabs

# Set up shell environment:
set noclobber
set history=20
```

The .cshrc File

The sequence of events after login varies from one UNIX system to another. On many systems, the **.cshrc** is first read and executed by the C shell. You can modify the **.cshrc** file to specify the command-line prompt you wish to use, initialize the history list, and define aliases. The upcoming sections describe the way the **.cshrc** file shown in Figure 27-6 defines these. Let's first take a quick look at the **.login** file in the next section.

The .login File

On many UNIX systems, the **.login** file is read after the **.cshrc** file. There are only two issues related to setup present in the example shown. The first is the **tset** command, which sets the *TERM* environment variable. The **eval** preceding **tset** means that the C shell executes **tset** and its arguments without creating a child process. This allows **tset** to set environment variables in the current shell instead of a subshell, which would be useless. The **stty** command is used to set terminal I/O options. The two **set** commands are used to define shell variables, which I describe shortly. The **noclobber** does not permit redirection to write over an existing file. If you try to write over an existing file, such as **/tmp/processes** below, you receive a message that the file exists:




```
sys1 1:  ps  -ef  > /tmp/processes
/tmp/processes:  File exists
```

The ">" means to take the output of **ps** and rather than write it to your screen, write it to **/tmp/processes**. The file **/tmp/processes** will not be written over, however, with the output of **ps -ef** because **/tmp/processes** already exists and an environment variable called **noclobber** has been set. If **noclobber** is set, then redirecting output to this file will not take place. This is a useful technique for preventing existing files from being accidently overwritten. There are many forms of redirection that you'll find useful. Redirection is covered later in this chapter.

Initialize History List in .cshrc

The C shell can keep a history list of the commands that you have issued. If you wish to reissue a command or view a command you earlier issued, you can use the history list.

The commands issued are referred to by number, so you want to have a number appear at the command prompt. The following line in **.cshrc** provides a number following the system name:

```
set prompt = "$system \!: "
```

```
sys1 1:
```

We get into shell and environment variables shortly, but for now it is sufficient to know that **$system** corresponds to system name "sys1."

You can specify the number of previously issued commands that you want to save and view when you issue the **history** command. The following line in **.cshrc** sets the history list to 20:

```
set history = 20
```

The last 20 commands issued are displayed when you issue the **history** command.

The *savehist* variable allows you to save a specified number of history commands after logout. By default, when you log out, the history list is cleared. This variable has a value of 20, so that upon the next login there will be 20 commands from the previous session will be saved.

Command-Line History

You can view the history list a variety of different ways. Let's first view the last 20 commands by simply issuing the **history** command:

```
sys1 23: history
     4 whoami
     5 pwd
     6 find / -name login -print &
     7 hostname
     8 who
     9 more /etc/passwd
    10 history
    11 history 5
    12 echo $PATH
    13 more .login
    14 cat .login
    15 exit
    16 exit
    17 history -h
    18 pwd
    19 whoami
    20 cd /tmp
    21 cat database.log
    22 cd
```

```
    23 history
sys1 24:
```

We can also print the history list without line numbers, as shown in the following example:

```
sys1 24: history -h
pwd
find / -name login -print &
hostname
who
more /etc/passwd
history
history 5
echo $PATH
more .login
cat .login
exit
exit
history -h
pwd
whoami
cd /tmp
cat database.log
cd
history
history -h
sys1 25:
```

Next, let's print the history list in reverse order:

```
sys1 25: history -r
    25 history -r
    24 history -h
    23 history
    22 cd
    21 cat database.log
    20 cd /tmp
    19 whoami
    18 pwd
    17 history -h
    16 exit
    15 exit
    14 cat .login
    13 more .login
    12 echo $PATH
    11 history 5
    10 history
     9 more /etc/passwd
     8 who
     7 hostname
     6 find / -name login -print &
sys1 26:
```

We can also select the number of events we want to print from the history list. The following example prints only the last ten commands from the history list:

```
sys1 26: history 10
    17 history -h
    18 pwd
    19 whoami
    20 cd /tmp
    21 cat database.log
    22 cd
    23 history
    24 history -h
    25 history -r
    26 history 10
sys1 27:
```

You can see that you have a variety of ways to produce a list of commands previously issued. Table 27-5 summarizes the commands issued in this section:

Table 27-5. Command-Line History

Table 27-3: Command-Line History

Command	Description	Example
history	The history list is produced with each command numbered.	**history**
history -h	The history list is produced without line numbers.	**history -h**
history -r	The history list is produced in reverse order with each command numbered.	**history -r**
history *n*	The last *n* commands from the history list are produced with each command numbered.	**history 10**

Re-Executing Commands from the History List

You can re-execute commands from the history list using a variety of techniques. We'll re-execute commands from the history list using the most common techniques.

You can repeat the last command with **!!**, the second command with **!2**, and the last command that started with "c" with **!c**. Let's issue the **history** command to get a list of the last 20 commands and then re-execute some of them:

```
sys1 27: history
     8 who
     9 more /etc/passwd
    10 history
    11 history 5
    12 echo $PATH
    13 more .login
    14 cat .login
    15 exit
    16 exit
    17 history -h
    18 pwd
    19 whoami
    20 cd /tmp
    21 cat database.log
    22 cd
    23 history
    24 history -h
    25 history -r
    26 history 10
    27 history
sys1 28:
```

Let's first reissue the last command with **!!**:

```
sys1 28: !!
history
     9 more /etc/passwd
    10 history
    11 history 5
    12 echo $PATH
    13 more .login
    14 cat .login
    15 exit
    16 exit
    17 history -h
    18 pwd
    19 whoami
    20 cd /tmp
    21 cat database.log
    22 cd
    23 history
    24 history -h
    25 history -r
    26 history 10
    27 history
    28 history
```

```
sys1 29:
```

Let's now reissue the 19th command with **!19**:

```
sys1 29: !19
whoami
martyp2
sys1 30:
```

Let's now reissue the last command beginning with "p" with **!p**:

```
sys1 30: !p
pwd
/home/martyp2
sys1 31:
```

Table 27-6 includes some of the more commonly used history list recall commands:

Table 27-6. Recalling from the History List

Command	Description	Example
!N	Issue command **N**	**!2**
!!	Issue last command	**!!**
!-N	Issue **Nth** command from last command issued	**!-N**
!str	Issue last command starting with **str**	**!c**
!?str?	Issue last command that had **str** anyplace in command line	**!?cat?**
!{str}str2	Append **str2** to last command with **str1**	**!{cd} /tmp**
^str1^str2^	Substitute **str2** for **str1** in last command	**^cat^more^**

Aliases in .cshrc

An alias is a name that you select for a frequently used command or series of commands. Many aliases are predefined for you.

You can use the **.cshrc** file as a place where your aliases are stored and read every time you log in. You can also define aliases at the command-line prompt, but these are cleared when you log out.

The **alias** command, without any arguments, lists all aliases. This list includes both preset aliases as well as those you have set. The following **alias** command shows all preset aliases on the system on which I am working:

```
sys1 7: alias
d       dirs
m       more
pd      pushd
pd2     (pushd +2)
po      popd
status (date; bdf)
sys1 8:
```

You are not restricted to using only the preset aliases. To create your own alias, you first issue the **alias** command, the name of the alias, and then the command or commands that are executed when the alias is executed.

Let's now create a few simple aliases. The first creates an alias of "h" for the **history** command:

```
sys1 1: alias h history
sys1 2: h
        history
```

Every time you type **h**, the history command is executed.

The following example creates an alias of a command that contains spaces, so the command is surrounded by single quotes:

```
alias ls='ls -al'
alias
ls
```

The first command creates an alias for "ls" that executes the **ls -al** command. We then issued the **alias** command to see whether indeed the new alias appears in the list of aliases. Then we run **ls** to see whether **ls -al** is run.

You don't have to keep an alias for the duration of your session after having set it. If you don't like an alias, you can use the **unalias** command to remove an alias.

To see **unalias** work, let's again produce a list of aliases, use **unalias** to unset the **h** alias, and run the **h** command to see if, indeed, it has been removed:

```
unalias h
h
```

When we issued **unalias** to remove the **h** alias and then try to run *h,* we were told that **h** is not found.

```
sys1 3:alias procs 'echo "Number of processes are: \c";
ps -ef | wc -l'
                    # single quote on outside
                    # double quote on inside
```

When you run **procs**, you see the following:

```
sys1 4: procs
Number of processes are: 44
```

A lot of quoting takes place in this command line. To understand what is taking place on this line, consult Table 27-7 for help.

Table 27-7. Shell Quoting

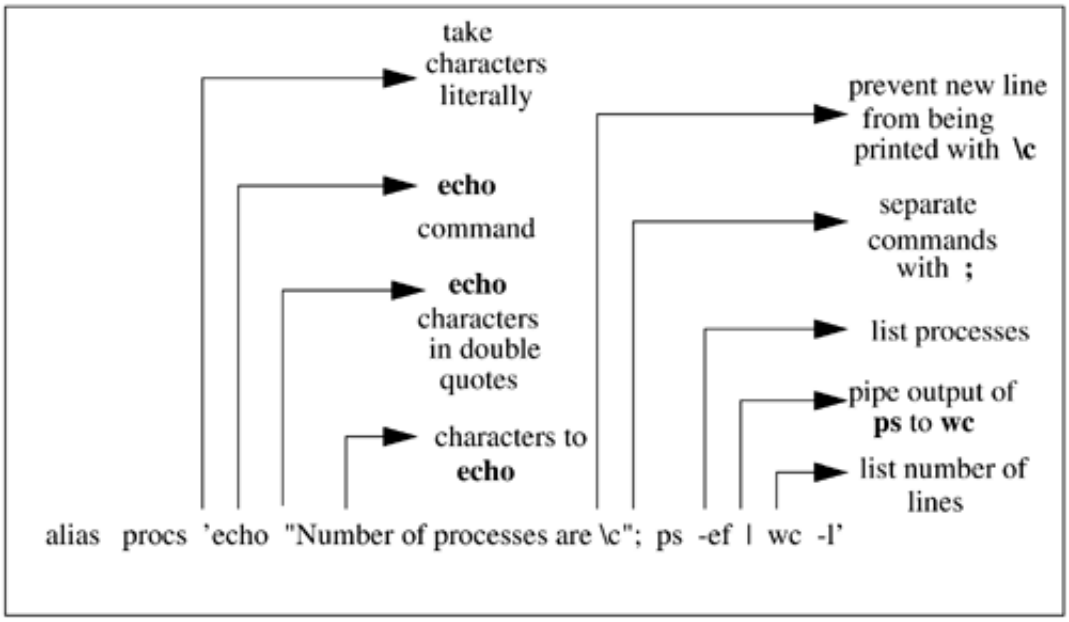
Character(s)	Description
'cmd'	Single quote means to take the string character literally
"str"	Double quote means to allow command and variable substitution
\c	Escape character that prevents everything following it from printing, including new line
'str'	Grave means to execute command and substitute output



Applying Table 27-7 to the earlier **procs** alias, we can see what comprises this alias. The alias begins with a single quote, which means to execute the command(s) within the single quotes. The first command is the **echo** command, which uses double quotes to specify the characters to **echo**. Embedded in the double quotes is the escape character **\c**, which prevents a new line from being printed. The semicolons separate commands. **ps** is then run to produce a list of processes, and the output is piped (**|**) to word count (**wc**) which produces a count of the number of lines. There are actually 43 processes running, because an extra line consisting of the **ps** headings is reported by **wc**.

As you can see in Figure 27-8, some of the quoting becomes tricky. An understanding of quoting is important if you wish to modify and reuse existing shell scripts or craft your own.

Figure 27-8. Quoting Example


take characters literally
prevent new line from being printed with \c
echo command
separate commands with ;
echo characters in double quotes
list processes
pipe output of ps to wc
characters to echo
list number of lines
alias procs 'echo "Number of processes are \c"; ps -ef | wc -l'

Introduction to Bash

With many developers using the Open Source development tools in HP-UX 11i, there is a lot of interest in Bash. Bash is widely used, although not required, when using GNU tools. In addition, the Linux-based UNIX operating systems I have used configure Bash as the default shell. Bash is covered in this section on a Linux system, however, the features of Bash covered exist on HP-UX as well. Bash possesses many of the fine features of other shells, and in fact derives its name from **B**ourne **A**gain **SH**ell, which is a dead giveaway that it possesses at least some of the features of the Bourne shell. Bash is similar to other shells in that it provides a user interface to UNIX. You can use the Bash shell in the following three ways:

- Interactively type commands on the command line.
- Group commonly executed sets of commands into command files that you can execute by typing the name of the file.
- Create Bash shell programs using the structured programming techniques of the shell.

These three techniques are listed in the order in which you'll probably use them. First, you log in and use interactive commands. Then you group together commonly used commands and execute them with a single command. Finally, you may want to create sophisticated shell scripts.

For this reason, I'll describe these aspects of the Bash shell in the order in which they are listed. The command file and programming aspects of the Bash shell are covered in Chapter 28. Bash is very similar to the KornShell, which is the shell used in the shell programming chapter. You can, therefore, use the shell programming chapter as an introduction to programming with Bash as well. Keep in mind, however, that differences always occur when programming with one shell rather than another.

Issuing Commands

The first activity you perform after you log into the system is to issue commands at the prompt. A command you may want to issue immediately is **ls -al**. Here is what I see on my system after executing this command to check my present working directory and producing a long listing of all files when logged in as root:

```
# pwd
# ls -al
total 46
drwxr-xr-x   5 root     root         1024 Nov 26 19:40 .
drwxr-xr-x  20 root     root         1024 Nov  8 20:10 ..
-rw-r--r--   1 root     root          964 Nov 26 19:40 .bash_history
-rw-r--r--   1 root     root          674 Feb  5  1997 .bashrc
-rw-r--r--   1 root     root          602 Feb  5  1997 .cshrc
-rw-r--r--   1 root     root        14815 Nov  8 20:09 .fvwmrc.menus.prep
-rw-r--r--   1 root     root          116 Feb  5  1997 .login
-rw-r--r--   1 root     root          234 Feb  5  1997 .profile
drwxr-xr-x   2 root     root         1024 Nov  8 14:10 .seyon
-rw-r--r--   1 root     root         4276 Nov  8 20:09 XF86Config
-r--r--r--   1 root     root        13875 Nov  8 20:05 XF86Config.bak
drwxrwxrwx   2 root     root         1024 Nov 26 19:40 book
drwxr-xr-x   5 root     root         1024 Nov 14 18:12 lg
-rw-r--r--   1 root     root            0 Nov 26 19:40 typescript
#
```

Among the files produced in the long listing of all files is a Bash startup file called **.bashrc** . The following shows the contents of the **.bashrc** file:

```
# cat .bashrc
# ~/.bashrc --
#   The individual per-interactive-shell startup file for bash

. /etc/profile
```

```
# try solve this tedious 'Backspace vs. Delete' problem...
if [ -z "$TERM" ]; then
  echo ".bashrc: TERM empty: this shouldn't happen!" 1>&2
  echo "   Please contact 'support@lst.de'" 1>&2
else
  case $TERM in
  linux*)
    stty erase '^?'
    ;;
  *)
    stty erase '^H'
    ;;
  esac
fi

# general environment settings
#export GROFF_TYPESETTER=latin1
#export LC_CTYPE=iso-8859-1
export LESSCHARSET=latin1
#export METAMAIL_PAGER=less

HISTSIZE=100

alias which='type -path'
alias h=history
alias j="jobs -l"
alias l="ls -Fax"
alias ll="ls -Alg"
alias pd=pushd
alias z=suspend

#
```

The .**bashrc** file has some interesting contents. Among them is a value for *HISTSIZE,* which we'll get into shortly, and a set of aliases. These aliases are "shortcuts" for long commands. When I issue the **ll** command, for instance, I am really issuing the **ls -Alg** command.

I may execute both the local **.profile** shown in the earlier long listing as well as **/etc/profile**. **/etc/profile** usually performs setup for all users who log into the system. The following is a listing of **/etc/profile**:

```
# cat /etc/profile
# /etc/profile
# System wide environment and startup programs
# Functions and aliases go in $HOME/.bashrc

PATH="/bin:/usr/bin:/opt/bin:/usr/X11R6/bin:/usr/openwin/bin:/usr/TeX/bin:/usr/local/bin"

umask 022

if [ 'id -gn' = 'id -un' ] && [ 'id -u' != 0 ]; then
  umask 002
fi

if [ -z "$UID" ]; then
  UID='id -u'
fi

if [ "$UID" = 0 ]; then
  PATH=/sbin:/usr/sbin:$PATH
else
  PATH=$PATH:
fi
USER='id -un'
LOGNAME=$USER

export PATH USER LOGNAME

HOSTNAME='/bin/hostname'
MAIL="/var/spool/mail/$USER"

export HOSTNAME MAIL

if [ -n "$BASH_VERSION" ]; then
  # (aliases now in $HOME/.bashrc, resp. /etc/skel/.bashrc)
  export PS1="[\u@\h \W]\\$ "
```

```
  export HISTSIZE=100
fi

#
```

We'll also cover some of the contents of **/etc/profile**.

Initializing the History List in .bashrc

The Bash shell can keep a history list of the commands you have issued. If you wish to reissue a command or view a command you earlier issued, you can use the history list.

You can specify any number of commands to be included in the history list. The following line in **.bashrc** sets the history list to 100:

set history = 100

One hundred commands will be saved in the history list. When you log, out the last 100 commands you have issued are stored in the history list. The next time you log in you can view these 100 commands; however, as you issue commands, the oldest commands fall off the history list. This fact is shown in the following example:

```
# history
    2  more history
    3   ll
    4  cd ..
    5  pwd
    6  cd ..
    7  ll
    8  cd ..
    9  ll
   10  ll log
   11  cd log
   12  more *
   13  l
   14  ll
   15  cd /
   16  ll
   17  cd
   18  XF86Setup
   19  XF86Setup
   20  startx
   21  ll
   22  pwd
   23  ll
   24  ll /
   25  XF86Setup
   26  ll
   27  startx
   28  find / -name XF86Config*
   29  cp /usr/X11R6/lib/X11/XF86Config.eg .
   30  ll
   31  XF86Setup
   32  XF86Setup
   33  startx
   34  ll
   35  mv XF86Config.eg XF86Config
   36  XF86Setup
   37  startx
   38  shutdown -h now
   39  man ls
   40  man ll
   41  man ls
   42  man file
   43  lsr
   44  man chmod
   45  man chmod
   46  shutdown -h now
   47  pwd
   48  ls -l
   49  pwd
   50  ls -a
```

```
   51  ls -al
   52  pwd
   53  ls -al
   54  more .profile
   55  more .bashrc
   56
   57  alias
   58  ll
   59  pwd
   60  script
   61  script
   62  scrit
   63  script
   64  more .bashrc
   65  more .bashrc
   66   ll
   67  more .profile
   68   ll
   69  more .bashrc | grep P
   70  more .profile | grep P
   71  env
   72  more /.profile
   73  more /etc/profile
   74  more /etc/profile | grep PS
   75  find / -name *profile* -print
   76  more .bashrc
   77   more .bashrc
   78  ll /etc/profi*
   79  cp /etc/profile /etc/profile.orig
   80  vi /etc/profile
   81  exit
   82  cp /etc/profile.orig /etc/profile
   83  history
   84
   85  exit
   86  history
   87  history
   88    ll
   89    ll
   90  history
   91  ll /etc/profi*
   92  ll /etc/profi*
   93   more .bashrc
   94    history
   95  ll
   96  cd /root
   97  ll
   98  history | more
   99   history | more
  100   exit
  101  history
```

Notice in this example that command number *100* is the **exit**, or command to log out, from the last session. Command number *101* is the **history** command I issued immediately upon establishing the next session.

Recalling from the History List

All these commands (**cp, more, find, ll**) are in the history list with their corresponding numbers. You can repeat the last command with **!!**, the 89th command with **!89**, and the last command that started with "m" with **!m**, all of which are shown in the following example:

```
# !!
history
    3   ll
    4  cd ..
    5  pwd
    6  cd ..
    7  ll
    8  cd ..
    9  ll
   10  ll log
   11  cd log
```

```
12  more *
13  l
14  ll
15  cd /
16  ll
17  cd
18  XF86Setup
19  XF86Setup
20  startx
21  ll
22  pwd
23  ll
24  ll /
25  XF86Setup
26  ll
27  startx
28  find / -name XF86Config*
29  cp /usr/X11R6/lib/X11/XF86Config.eg .
30  ll
31  XF86Setup
32  XF86Setup
33  startx
34  ll
35  mv XF86Config.eg XF86Config
36  XF86Setup
37  startx
38  shutdown -h now
39  man ls
40  man ll
41  man ls
42  man file
43  lsr
44  man chmod
45  man chmod
46  shutdown -h now
47  pwd
48  ls -l
49  pwd
50  ls -a
51  ls -al
52  pwd
53  ls -al
54  more .profile
55  more .bashrc
56
57  alias
58  ll
59  pwd
60  script
61  script
62  scrit
63  script
64  more .bashrc
65  more .bashrc
66   ll
67  more .profile
68   ll
69  more .bashrc | grep P
70  more .profile | grep P
71  env
72  more /.profile
73  more /etc/profile
74  more /etc/profile | grep PS
75  find / -name *profile* -print
76  more .bashrc
77   more .bashrc
78  ll /etc/profi*
79  cp /etc/profile /etc/profile.orig
80  vi /etc/profile
81  exit
82  cp /etc/profile.orig /etc/profile
83  history
84
85  exit
86  history
```

```
   87  history
   88    ll
   89    ll
   90  history
   91  ll /etc/profi*
   92  ll /etc/profi*
   93   more .bashrc
   94    history
   95  ll
   96  cd /root
   97  ll
   98  history | more
   99   history | more
  100   exit
  101  history
  102  history
# !89
  ll
total 44
-rw-r--r--   1 root     root          956 Nov 26 19:33 .bash_history
-rw-r--r--   1 root     root          674 Feb  5  1997 .bashrc
-rw-r--r--   1 root     root          602 Feb  5  1997 .cshrc
-rw-r--r--   1 root     root        14815 Nov  8 20:09 .fvwmrc.menus.prep
-rw-r--r--   1 root     root          116 Feb  5  1997 .login
-rw-r--r--   1 root     root          234 Feb  5  1997 .profile
drwxr-xr-x   2 root     root         1024 Nov  8 14:10 .seyon
-rw-r--r--   1 root     root         4276 Nov  8 20:09 XF86Config
-r--r--r--   1 root     root        13875 Nov  8 20:05 XF86Config.bak
drwxrwxrwx   2 root     root         1024 Nov 13 21:25 book
drwxr-xr-x   5 root     root         1024 Nov 14 18:12 lg
-rw-r--r--   1 root     root            0 Nov 26 19:36 typescript
# !m
more .bashrc
# ~/.bashrc --
#   The individual per-interactive-shell startup file for bash

. /etc/profile

# try solve this tedious 'Backspace vs. Delete' problem...
if [ -z "$TERM" ]; then
  echo ".bashrc: TERM empty: this shouldn't happen!" 1>&2
  echo "   Please contact 'support@lst.de'" 1>&2
else
  case $TERM in
  linux*)
    stty erase '^?'
    ;;
  *)
    stty erase '^H'
    ;;
  esac
fi

# general environment settings
#export GROFF_TYPESETTER=latin1
#export LC_CTYPE=iso-8859-1
[7m--More--(70%)[m
export LESSCHARSET=latin1
#export METAMAIL_PAGER=less

HISTSIZE=100

alias which='type -path'
alias h=history
alias j="jobs -l"
alias l="ls -Fax"
alias ll="ls -Alg"
alias pd=pushd
alias z=suspend

[root@nycald1 /root]#
Script done on Thu Nov 26 19:39:51 1998
```

Table 27-8 includes some of the more commonly used history list recall commands:

Table 27-8. Recalling from the History List

Command	Description	Example
!*N*	Issue command **N**	**!2**
!!	Issue last command	**!!**
!-*N*	Issue **Nth** command from last command issued	**!-N**
!*str*	Issue last command starting with **str**	**!c**
!?*str***?**	Issue last command that had **str** anyplace in command line	**!?cat?**
!{*str1***}***str2*	Append **str2** to last command with **str1**	**!{cd} /tmp**
^*str1***^***str2***^**	Substitute **str2** for **str1** in last command	**^cat^more^**

Editing on the Command Line

Using the history list is a great way of viewing and reissuing commands. Bash also supports command-line editing. You can use the up arrow key to move back one command in the history list. When you press the up arrow key, the last command from the history list appears on the command line. Every time you press the up arrow key, you move back one more command in the history list. When a command appears on the command line, you can press the "Enter" key to issue the command. You can modify the command by using the left and right arrow keys to move to a point in the command line and type additional information, or use the "backspace" and "delete" keys to remove information from the command line.

Aliases in .bashrc

An alias is a name that you select for a frequently used command or series of commands. You can use the **.bashrc** file as a place where your aliases are stored and read every time you log in. In the earlier **.bashrc** file, seven aliases were already set up. You can add additional aliases in the **.bashrc** file or define aliases at the command-line prompt, but these will be cleared when you log out.

Here is a list of the aliases that are already set up for us in the **.bashrc** file and an example of running the aliases **l** and **ll**:

```
# alias
alias h='history'
alias j='jobs -l'
alias l='ls -Fax'
alias ll='ls -Alg'
alias pd='pushd'
alias which='type -path'
alias z='suspend'
#
# l
./                ../                 .bash_history        .bashrc
.cshrc            .fvwmrc.menus.prep  .login               .profile
.seyon/           XF86Config          XF86Config.bak       book/
lg/               typescript
#
# ll
total 44
-rw-r--r--   1 root     root          970 Nov 26 21:35 .bash_history
-rw-r--r--   1 root     root          674 Feb  5  1997 .bashrc
-rw-r--r--   1 root     root          602 Feb  5  1997 .cshrc
-rw-r--r--   1 root     root        14815 Nov  8 20:09 .fvwmrc.menus.prep
-rw-r--r--   1 root     root          116 Feb  5  1997 .login
-rw-r--r--   1 root     root          234 Feb  5  1997 .profile
drwxr-xr-x   2 root     root         1024 Nov  8 14:10 .seyon
-rw-r--r--   1 root     root         4276 Nov  8 20:09 XF86Config
-r--r--r--   1 root     root        13875 Nov  8 20:05 XF86Config.bak
drwxrwxrwx   2 root     root         1024 Nov 26 19:41 book
```

```
drwxr-xr-x   5 root     root         1024 Nov 14 18:12 lg
-rw-r--r--   1 root     root            0 Nov 26 21:35 typescript
#
```



These are all very useful indeed, but let's now set up our own alias. Suppose that we want to know how many processes are running on the system. We'll create an alias called "procs" that does this for us. The **ps** command produces a list of processes. We'll issue **ps** and pipe (|) this output to **wc** with the "l" option to tell us how many lines are present. The pipe (|) directs the output of **ps** to be used as the input to **wc**. The **ps** command produces a list of processes, and **wc -l** gives us a count of the number of lines. Therefore, we'll know the total number of processes running. The following example first shows the output of **ps**, then our **alias** command, and finally the output produced by the **alias** command:

```
# ps
  PID TTY STAT  TIME COMMAND
  188   2 S    0:00 /sbin/getty tty2 VC linux
  189   3 S    0:00 /sbin/getty tty3 VC linux
  190   4 S    0:00 /sbin/getty tty4 VC linux
  191   5 S    0:00 /sbin/getty tty5 VC linux
  192   6 S    0:00 /sbin/getty tty6 VC linux
  619   1 S    0:00 login root
  620   1 S    0:00 -bash
  642   1 S    0:00 script
  643   1 S    0:00 script
  644  p0 S    0:00 bash -i
  656  p0 R    0:00 ps
#
# alias procs='echo "Number of processes are: ";ps | wc -l'
#
# procs
Number of processes are: 11
#
```

This alias works wonderfully. All we have to type is "procs" to see the number of processes running on our system.

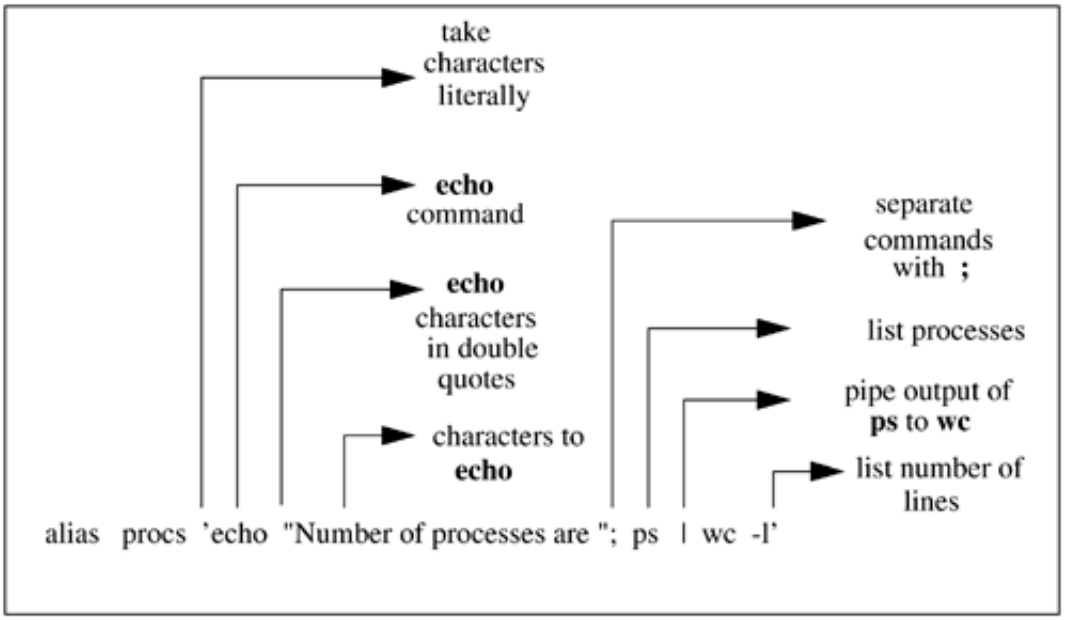
A lot of quoting takes place in this command line. To understand what is taking place on this line, consult Table 27-9:

Table 27-9. Shell Quoting

Character(s)	Description
'cmd'	Single quote means to take the string character literally
"str"	Double quote means allow command and variable substitution
\c	Escape character prevents everything following it from printing, including new line
'str'	Grave means to execute command and substitute output

Applying Table 27-9 to the earlier **procs** alias, we can see what comprises this alias. The alias begins with a single quote, which means to execute the command(s) within the single quotes. The first command is **echo**, which uses double quotes to specify the characters to **echo**. We could have added the escape character **\c**, which would have prevented a new line from being printed. The semicolons separate commands. **ps** is then run to produce a list of processes, and the output is piped (**|**) to word count (**wc**), which produces a count of the number of lines, as shown in Figure 27-9:

Figure 27-9. Quoting Example



As you can see in Figure 27-9, some of the quoting becomes tricky. An understanding of quoting is important if you wish to modify and reuse existing shell scripts or craft your own.

Command and Path Completion

Bash sometimes knows what you're thinking. You can type part of a command or pathname, and Bash can complete the remainder for you. You can type part of a command or pathname and use the "tab" key to complete the command. If, for instance, you wish to issue the **runlevel** command to view the current system run level, but can't remember the full command, you can type "run" and press the tab key and the command is completed for you, as shown in the following example:

```
# run<tab key>level
 N 3
```

Bash determined that the only command that starts with "run" is **runlevel** and completed the command.

As long as you issue the command or pathname to the extent that it is unique, then Bash completes it for you. If the command or pathname is not unique, then Bash shows you the options for completing the command. The following example shows typing "ru" and two tabs to get a list of commands that start with "ru":

```
# ru<tab key><tab key>
runlevel  rusers
```

You can see from this example that typing "ru" produced two possible commands - **runlevel** and **rusers**.

This great completion also works for path names. If you change directory to "/b," you get the following result:

```
# cd /b<tab key><tab key>
bin  boot
```

Because two directories at the root level begin with "b," Bash could not determine which of the two you wanted and listed both.



Chapter 28. Introduction to Shell Programming

Shell Programming

Steps to Create Shell Programs

ksh Programming

An HP-UX Logical Volume Manager Shell Program

C Shell Programming

Manual Pages for Commands Used in Chapter 28





Shell Programming

There is much more to a shell than meets the eye. The shell can do much more than the command-line interpreter everyone is used to using. UNIX shells actually provide a powerful interpretive programming language as well.

In this chapter, we'll cover **ksh** shell programming. I chose **ksh** because most **ksh** programming techniques work with the Bourne shell as well. There is a follow-on to **ksh** programming at the end of the chapter for **csh** because the **csh** employs some different programming techniques than the **ksh**.

We'll cover the most important **ksh** programming techniques. The climax of this chapter is a fairly sophisticated shell program to remove files and place them in a directory, rather than just permanently removing files from the system with **rm**. This program employs all the shell programming techiques covered in the chapter. This shell program, called **trash**, with some minor modifications, can be run in the Bourne shell as well.

The shell is one of the most powerful and veratile features on any UNIX system. If you can't find the right command to accomplish a task, you can probably build it quite easily using a shell script.

The best way to learn shell programming is by example. There are many examples given in this chapter. Some serve no purpose other than to demonstrate the current topic. Most, however, are useful tools or parts of tools that you can easily expand and adapt into your environment. The examples provide easy-to-understand prompts and output messages. Most examples show what is needed to provide the functionality we are after. They do not do a great deal of error checking. From my experience, however, it only takes a few minutes to get a shell program to do what you want; it can take hours to handle every situation and every error condition. Therefore, these programs are not very dressed up (perhaps a sport coat versus a tuxedo). I'm giving you what you need to know to build some useful tools for your environment. I hope that you will have enough knowledge and interest by the time we get to the end of this chapter to learn and do more.

Most of the examples in this chapter were performed on a Solaris system; however, shells are nearly identical going from one system to another, so you should be able to get the programs in this chapter running on your system quickly.



Steps to Create Shell Programs

When you craft a shell program, there are some initial steps you want to perform so that you have consistency among all of your programs. The following is a list of some important concepts to employ with your shell programs:

1. **Names of shell programs and their output** - You should give your shell programs and their output a meaningful name. If you call your shell programs *script1, script2,* and so on, these will not have any meaning for you and other potential users of the programs. If your shell program finds files of a particular type, then you can name the program *filefind* or some other descriptive name. Do the same with the output of shell programs. Don't call the output of a shell program *output1* if there is some descriptive name you can give it such as *filefind.out*. Also avoid naming shell programs and their output names that already exist. You may use commands such as **read** and **test**, so you may create confusion and conflicts if you were to give your program and their output the same names. The first shell program in this chapter shows today's date. If we were to name the program **date**, we would actually run the system **date** command as opposed to our shell program **date**. The reason is that in most cases, the system **date** command would be found before our shell program **date** file.

2. **Repository for shell programs** - If you plan to produce numerous shell programs, you may want to place them in a directory. You can have a directory in your home directory called **bin** or **shellprogs**, in which all programs are located. You can then update your path to include the directory name where your shell programs are located.

3. **Specify shell to execute program** - The very first line of the program specifies the shell to be used to execute the program. The path of the shell is preceeded by **#!**, which is a "magic number" that indicates that the path of the shell is about to follow. The shell programs in this chapter use the **ksh** path.

4. **Formatting of shell programs** - Do not underestimate the importance of proper formatting to enhance the readability of shell programs. You should indent areas of the program to indicate that commands are part of a group. Indenting only three or four spaces is fine for groups of commands. You may also want to set autoindent in **vi** so that the next line starts at the same point as the previous line (try **:set ai** in **vi** to see how this technique works). You can break up long lines by placing a **** (backslash) at the end of one line and continue the command on the next line.

5. **Comments** - Include detailed comments in your shell program. You can write paragraphs of comments to describe what an upcoming section of your program will accomplish. These will help you understand your program when you look at it months or years after having originally created it as well as help others understand the program. No limit exists on the number of comments you can include. Start comment lines with a # (pound sign).

6. **Make the shell program executable** - The **chmod** command covered in the previous chapters introducing various shells is used to make your program executable.

Following the previous list of recommendations will make creating and using shell programs more efficient. Now we need to cover the types of shell programs.

A shell program is simply a file containing a set of commands you wish to execute sequentially. The file needs to be set with execute permissions so that you can execute it just by typing the name of the script.

There are two basic forms of shell programs exist:

1. **Simple command files** - When you have a command line or a set of command lines that you use over and over, you can use one simple command to execute them all.

2. **Structured programs** - The shell provides much more than the ability to "batch" together a series of commands. It has many of the features that any higher-level programming language contains:

    - Variables for storing data
    - Decision-making controls (the **if** and **case** commands)
    - Looping abilities (the **for** and **while** loops)
    - Function calls for modularity

Given these two basic forms you can build everything from simple command replacements to much larger and more complex data manipulation and system administration tools.

[Team LiB]

ksh Programming

I have created a directory in my home directory called **shellprogs** to serve as a repository for my shell programs. In order to execute these programs without having to specify an absolute path, I have updated my path in **.profile** to include the following line:

```
export PATH=${PATH}:~shellprogs
```

This adds **/home/martyp/shellprogs** to my path assuming colon is the delimiter, which is usually the case. After updating **.profile** you can log out and log in to update your path. You can issue the command below to confirm that your path has indeed been updated to include the directory in which you keep your shell programs:

```
martyp $ echo $PATH
/usr/bin:/usr/ucb:/etc:.:/home/martyp/shellprogs
martyp $
```

Let's now go to the **shellprogs** directory and type a simple command file called **today**:

```
#!/bin/ksh
# This is a simple command file to display today's date.
echo "Today's date is"
date +%x
```

Before we look at each line in the program, let's run it to see the output:

```
martyp $ today
ksh: today: cannot execute
martyp $ ls -al
total 6
drwxrwxr-x   2 martyp   staff    512 May 21 09:53 .
drwxrwx---   4 martyp   staff    512 May 21 09:25 ..
-rw-rw-r--   1 martyp   staff    100 May 21 09:54 today
martyp $ chmod +x today
martyp $ ls -al
total 6
drwxrwxr-x   2 martyp   staff     512 May 21 09:53 .
drwxrwx---   4 martyp   staff     512 May 21 09:25 ..
-rwxrwxr-x   1 martyp   staff     100 May 21 09:54 today
martyp $ today
Today's date is
05/21/99
martyp $
```

We could not execute **today** because the file was created without execute permissions, which we confirm by performing a long listing. We then add execute permission to the file with **chmod +x**, which is confirmed with the next long listing. We are then able to execute **today** and view its results.

The **umask** discussion in each of the earlier shell chapters describes the defaults for new files. Almost invariably, new files are created without execute permission; therefore, you will have to update the permissions on the file to include execute.

Let's now walk through this simple shell program and analyze each line.

```
#!/bin/ksh
```

The first line specifies that the **ksh** will be used. If you are using Bash, C shell, or any other shell, you would specify its location in this manner. Some systems have multiple versions of shells running on them. It may be that a shell has been updated since the last release and some users may want to maintain the old version of the shell. For this reason, you want to be sure that the absolute path you specify is indeed that of the shell you wish to use. Note that the **#!** must be the very first two characters in the file.

Normally, when you run a shell program, the system tries to execute commands using the same shell that you are using for your interactive command lines. If we don't include this line, someone running a shell other than the **ksh** might have unexpected results when trying to run one of our programs.

As a good practice, you should include **#!shellname** as the first line of every shell program you write.

Let's now view the next line of our program:

```
# This is a simple command file to display today's date.
```

These are comments. Everything after a # in a command line is considered a comment (**#!** on the first line is the one very big exception). Keep in mind my early remarks about including comments liberally. It is a great pleasure to share a well commented shell program with a friend, knowing that you have adequately documented the program with comments.

Here is the next command:

```
echo "Today's date is"
```

The **echo** command generates prompts and messages in shell programs. See the **echo** manual entry to see all the options available with **echo** for formatting your output. We commonly enclose the string to be displayed in double quotes. In this case, we did because we needed to let the shell know that the apostrophe was part of the string and not a single quote that needs a match.

Next is the last command in our program:

```
date +%x
```

This executes the **date** command. There are indeed many options to the **date** command, some of which are not intuitive. In this case, we use one of the simplest forms of the command, which simply produces today's date.

Let's cover one more example of a command file that you may find useful. This program informs you of the present working directory and then produces a long listing of the files in the directory. The following is a listing of the shell program **myll**:

```
#!/bin/ksh
# This is a simple shell program that displays the current
# directory name before a long file listing (ll) of that
# directory.
# The script name is myll
echo "Long listing of directory:"
pwd
echo
ll -l
```

This program uses **ll**; you may need to use **ls -al**. The following is what **myll** looks like when it runs:

```
martyp $ myll
Long listing of directory:
/home/martyp/shellprogs

total 4
-rwxrwxr-x   1 martyp   staff     220 May 24 05:28 myll
-rwxrwxr-x   1 martyp   staff     100 May 21 09:54 today
martyp $
```

This name of the present working directory is **/home/martyp/shellprogs**. A long listing of the contents of this directory shows the two programs we have covered so far in this chapter.

Before we can produce more complex shell programs, we need to learn more about some of the programming features built into the shell. Let's start with shell variables.

Shell Variables

A shell variable is similar to a variable in any programming language. A variable is simply a name you give to a storage location. Unlike most languages, however, you never have to declare or initialize your variables; you just use them.

Shell variables can have just about any name that starts with a letter (uppercase or lowercase). To avoid confusion with special shell characters (like file name generation characters), keep the names simple and use just letters, numbers, and underscore (_).

To assign values to shell variables, you simply type the following:

```
name=value
```

Note that there are no spaces before and after the = character.

Here are some examples of setting shell variables from the command line. These examples work correctly:

```
$ myname=ralph
$ HerName=mary
```

This one does not work because of the space after "his":

```
$ his name=norton
his: not found
```

The shell assumes that "his" is a command and tries to execute it. The rest of the line is ignored.

This example contains an illegal character (+) in the name:

```
$ one+one=two
one+one=two: not found
```

A variable must start with a letter. A common mistake is to give a variable a name that makes perfect sense to you but does not start with a letter. The following example uses a variable that starts with a number:

```
$ 3word=hi
```

```
3word=hi: not found
```

The "3" causes a "not found" to be produced when we attempt to assign this variable.

Now that we can store values in our variables, we need to know how to use those values. The dollar sign ($) is used to get the value of a variable. Any time the shell sees a $ in the command line, it assumes that the characters immediately following it are a variable name. It replaces the *$variable* with its value. Here are some simple examples using variables at the command line:

```
$ myname=ralph
$ echo myname
myname

$ echo $myname
ralph
$ echo $abc123
```

In the first **echo** command, there is no $, so the shell ignores **myname**, and **echo** gets **myname** as an argument to be echoed. In the second **echo**, however, the shell sees the $, looks up the value of **myname,** and puts it on the command line. Now **echo** sees **ralph** as its argument (not **myname** or **$myname**). The final **echo** statement is similar, except that we have not given a value to **abc123** so the shell assumes that it has no value and replaces **$abc123** with nothing. Therefore, echo has no arguments and echos a blank line.

There may be times when you want to concatenate variables and strings. This is very easy to do in the shell:

```
$ myname=ralph
$ echo "My name is $myname"
My name is ralph
```

There may be times when the shell can become confused if the variable name is not easily identified in the command line:

```
$ string=dobeedobee
$ echo "$stringdoo"
```

We wanted to display "dobeedobee," but the shell thought the variable name was stringdoo, which had no value. To accomplish this we can use curly braces around the variable name to separate it from surrounding characters:

```
$ echo "${string}doo"
dobeedobeedoo
```

You can set variables in shell programs in the same way, but you might also like to do things such as save the output of a command in a variable so that we can use it later. You may want to ask users a question and read their response into a variable so that you can examine it.

Command Substitution

Command substitution allows us to save the output from a command (**stdout**) into a shell variable. To demonstrate this, let's take another look at how our "today" example can be done using command substitution.

```
#!/bin/ksh
d='date +%x'
echo "Today's date is $d"
```

The back quotes (') around the **date** command tell the shell to execute date and place its output on the command line. The output will then be assigned to the variable **d**. We'll name this updated script **today1** and run it:

```
$ today1
Today's date is 05/24/00
```

We could also have done this task without using the variable **d**. We could have just included the **date** command in the echo string, as shown in the **today2** script shown in the following example:

```
#!/bin/ksh
echo "Today's date is 'date +%x'"
```

When we run this program, we see exactly the same output as we did with **today1**:

```
$ today2
Today's date is 05/24/00
```

We'll use shell variables and command substitution extensively in some of the upcoming examples. Let's now cover reading user input.

Reading User Input

The most common way to get information from the user is to prompt him or her and then read the response. The **echo**

command is most commonly used to display the prompt; then the **read** command is used to read a line of input from the user (standard input). Words from the input line can be assigned to one or several shell variables.

Here is an example with comments to show you how **read** can be used:

```
#!/bin/ksh
# program: readtest
echo "Please enter your name: \c"      # the \c leaves cursor on
                                       # this line.

read name       # I have no $ because we are doing an assignment
                 # of whatever the user enters into name.

echo "Hello, $name"

echo "Please enter your two favorite colors: \c"

read color1 color2        # first word entered goes into color1
                           # remainder of line goes into color2

echo "You entered $color2 and $color1"
```

If we ran this program, it would look something like this:

```
$ readtest
Please enter your name: gerry
Hello, gerry
Please enter your two favorite colors: blue green
You entered green and blue
$
```

Notice how the **read** command assigned the two words entered for colors into the two respective color variables. If the user entered fewer words than the **read** command was expecting, the remaining variables are set to null. If the user enters too many words, all extra words entered are assigned into the last variable. This technique is how you can get a whole line of input into one variable. Here's an example of what happens whn you enter more than two colors:

```
$ readtest
Please enter your name: gerry
Hello, gerry
Please enter your two favorite colors: chartreuse orchid blue You entered orchid blue and chartreuse
$
```

The program took the last two colors entered and assigned them to *color2*. For this reason, you have to be careful of what a user may enter with the read command and how you map that information to variables.

You may have a built-in variable used with **read** called *REPLY.* With *REPLY,* a line of input is assigned to the variable *REPLY*. I tend not to use this often because I don't like to leave anything to chance. I normally explicitly name a variable to which input is assigned. The following example shows the listing and running of **readtest1**, which includes a line that contains only the **read** command, the response of which is assigned to *REPLY*:

```
martyp $cat readtest1

#!/bin/ksh
# program: readtest
echo "Please enter your name: \c"      # the \c leaves cursor on
                                       # this line.

read name       # I have no $ because we are doing an assignment
                 # of whatever the user enters into name.

echo "Hello, $name"

echo "Please enter your two favorite colors: \c"

read color1 color2        # first word entered goes into color1
                           # remainder of line goes into color2

echo "You entered $color2 and $color1"

echo "Where are you from?"

read                              # read response into $REPLY
echo "I'm sure $REPLY is great"


martyp $ readtest1
Please enter your name: MARTY
Hello, MARTY
Please enter your two favorite colors: RED BLUE
You entered BLUE and RED
Where are you from?
MANY DIFFERENT PLACES
I'm sure MANY DIFFERENT PLACES is great
```

```
martyp $
```

You can see in this example that the response I typed of "MANY DIFFERENT PLACES" was indeed read into *REPLY*.

Although **echo** is used throughout this chapter, you may also see **print** used to display lines on the screen.

Arguments to Shell Programs

Shell programs can have command-line arguments just like any regular command. Command-line arguments that you use when you invoke your shell program are stored in a special set of variables. These are called the *positional parameters*.

The first ten words on the command line are directly accessible in the shell program using the special variables **$0-$9**. This is how they work:

$0	The command name
$1	The first argument
$2	The second argument
$3	.
	.
	.
$9	The ninth argument

If you are not sure how many command line arguments you may get when your program is run, there are two other variables that can help:

$#	The number of command-line arguments
$*	A space-separated list of all the command-line arguments (which does *not* include the command name)

The variable **$*** is commonly used with the **for** loop (soon to be explained) to process shell script command lines with any number of arguments.

Let's now take the **myll** we worked with earlier and modify it to produce a long listing of the directory that we specify when we run the program. Figure 28-1 shows the modified **myll**:

Figure 28-1 myll Shell Program

```
#!/bin/ksh
# This is a simple shell program that takes one command line
# argument (a directory name) and then displays the full pathname
# of that directory before doing a long file listing (ll) on
# it.
#
# The script name is myll
cd $1
echo "Long listing of the 'pwd' directory:"
echo
ls -l
```

If we run **myll** with a directory name, the script changes directory, echoes the message containing the full path name (notice the command substitution), and then executes the **ls -l** command.

Note that the **cd** in the **myll** program will change only the working directory of the script; it does not affect the working directory of the shell from which we run **myll**.

```
martyp $ myll /tmp
Long listing of the /tmp directory:

total 2384
-rw-------   1 root     sys       265228 Feb 22 15:21 dtdbcache_:0
-rw-r--r--   1 root     sys        70829 Feb 23 10:44 g
-rw-r--r--   1 root     sys        13642 Feb 23 10:48 i
-rw-rw-rw-   1 root     root       14071 May 24 06:10 license_log
-rwxr-xr-x   1 chrisb   users        317 Apr 20 17:38 ls
-rw-rw-r--   1 root     sys         4441 Mar 25 14:37 mwaps7454
-rw-rw-rw-   1 anne     users       4341 May 20 13:56 ps23974
```

```
-rw-r--r--   1 rodt     users       4218 Apr 14 11:17 ps3358
-rw-r--r--   1 rodt     users       4763 Feb 24 07:23 ps6465
-rw-rw-r--   1 root     sys         4446 Mar 25 14:31 ps7036
-rw-rw-r--   1 root     sys         4442 Mar 25 14:35 ps7138
-rw-rw-r--   1 root     sys         4446 Mar 25 14:35 ps7215
-rw-rw-r--   1 root     sys         4498 Mar 25 14:36 ps7342
-rw-rw-r--   1 root     sys         4446 Mar 25 14:38 ps7622
-rw-rw-r--   1 root     sys         4615 Mar 25 15:30 ps7812
-rw-rw-r--   1 root     sys         5728 Feb 18 11:09 ps_data
-rw-r--r--   1 root     sys            0 Apr 26 10:50 sh20890.1
-rw-r--r--   1 root     sys            0 Apr 26 10:50 sh20891.1
-rw-r--r--   1 root     sys            0 Apr 26 10:51 sh20978.1
-rw-r--r--   1 root     sys            0 Apr 26 10:51 sh20979.1
-rw-r--r--   1 chrisb   users       5325 Mar 26 13:42 sman_9664
-rw-rw-r--   1 root     sys       295996 Mar  1 10:15 ups_data
drwx------   2 root     other         69 Mar  9 11:37 whatis.11526
drwx------   2 root     other         69 Mar  9 11:37 whatis.11686
drwx------   2 root     other         69 Mar  9 11:38 whatis.11853
drwx------   2 root     other         69 Mar  9 11:38 whatis.12014
-rw-r--r--   1 root     sys       354221 Feb 23 10:49 x
-rw-r--r--   1 chrisb   users          0 Feb 23 14:39 xx
martyp $
```

In this case, we could give **myll** no argument and it would still work properly. If we don't provide any command-line arguments, then **$1** will be null, so nothing goes on the command line after **cd**. This will make **cd** take us to our home directory and perform the **ll** there.

If we provide more than one argument, only the first is used and any others are ignored.

If we use a command-line argument, it *must* be a directory name; otherwise, the **cd** command fails and the script terminates with a "bad directory" error message. Later I will show how to test for valid directory names and file names so that you can work around potential errors.



A more complex example can be used to build new versions of the **ps** command. Below are two examples that use command line arguments and command substitution to help you with your process management.

The **psg** shell program in Figure 28-2 is handy for searching through what is typically a long process status listing to find only certain commands or user processes. These examples use **grep. grep** finds all lines that contain the pattern for which you are searching.

Figure 28-2 psg Shell Program

```
#!/usr/bin/sh
# Program name: psg
# Usage: psg some_pattern
#
# This program searches through a process status (ps -ef)
# listing for a pattern given as the first command-line
# argument.
procs='ps -ef'                  # Get the process listing
head='echo "$procs" | line'     # Take off the first line (the
                                # headings)
echo "$head"                    # Write out the headings
echo "$procs" | grep -i $1 | grep -v $0 # Write out lines
    # containing $1 but not this program's command line
# Note that $procs MUST be quoted or the newlines in the ps
# -ef listing will be turned into spaces when echoed. $head
# must also be quoted to preserve any extra white space.
```

Here's what **psg** looks like when it runs. In this example, we want to look at all the Korn shells running on the system.

```
martyp $ psg ksh
     UID   PID  PPID  C    STIME TTY      TIME CMD
    root  2954  2936  0   Feb 22 ?        0:01 /bin/ksh /usr/dt/bin/Xsession
    root  3002  2999  0   Feb 22 pts/2    0:01 -ksh -c      unset DT;     DISPLg
    root  3067     1  0   Feb 22 ?        0:00 /bin/ksh /usr/dt/bin/sdtvolcheckm
   jnola 11516 11514  0   May 11 pts/3    0:01 -ksh
  martyp 29291 29289  0 09:30:04 pts/4    0:02 -ksh
```



This program also works to find the terminal, process ID, parent process ID, start date, and any other information from **ps**.

As a user, you may start processes that you wish to stop. You may, for instance, start an application that does not come up on your display. You can identify the process with **psg** and then use the next program to stop the process, provided that you have the rights to stop the process.

The **gkill** shell program in Figure 28-3 searches through a **ps -ef** listing for a pattern (just like **psg**); then it kills all listed processes. The examples use the **cut** command, which allows you to specify a range of columns to retain.

Figure 28-3 gkill Shell Program

```
#!/usr/bin/sh
# Program name: gkill
# Usage: gkill some_pattern
# This program will find all processes that contain the
# pattern specified as the first command line argument then
# kills those processes.
# get the process listing
procs='ps -ef'
echo "The following processes will be killed:"
# Here we list the processes to kill. We don't kill this
# process
echo "$procs" | grep -i $1 | grep -v $0
# Allow the user a chance to cancel.
echo "\nPress Return to continue Ctrl-C to exit"
# If the user presses Ctrl-C the program will exit.
# Otherwise this read waits for the next return character and
# continue.
read junk
# find the pattern and cut out the pid field
pids='echo "$procs" | grep -i $1 | grep -v $0 | cut -c9-15'
# kill the processes
kill $pids
```



If we don't provide any command-line arguments, **grep** issues an error and the program continues. In the next section, we will learn how to check if **$1** is set and how to gracefully clean up if it's not.

Let's now start a process in the background, use the **psg** program to identify the process number, and then use **gkill** to stop the process:




```
martyp $ find / -name .c > cprogs 2>&1 &
[1]     29683
martyp $ psg find
     UID   PID  PPID  C    STIME TTY      TIME CMD
  martyp 29683 29579  7 13:54:19 pts/4    0:02 find / -name .c
martyp $ gkill 29683
The following processes will be killed:
  martyp 29683 29579 10 13:54:19 pts/4    0:03 find / -name .c

Press Return to continue Ctrl-C to exit

martyp $
```



When we start the **find** command, we are given its process number. To be sure that we don't **kill** the wrong process, I checked for all of the **find** programs running with **psg**. When I was sure of the process number to **kill**, I used **psg** to **kill** the process.

Although the shell programs in this section were simple, they employed many important shell programming techniques.

Let's now move to testing and branching, which are some of the most powerful aspects of shell programming.

Testing and Branching

Decision-making is one of the shell's most powerful features. You have two ways to check conditions and branch to a piece of code that can handle that condition.

For example, you may want to ask the user a question and then check whether the answer was yes or no. You may also want to check whether a file exists before you operate on it. In either case, you can use the **if** command to accomplish the task. Here are a few shell script segments that explain each part of the **if** command:

```
echo "Continue? \c"
read ans
if [ "$ans" = "n" ]
then
      echo "Goodbye"
      exit
fi
```

The **echo** and **read** provide a prompt and response, as usual. The **if** statement executes the next command, and if it succeeds, it executes any commands between the **then** and the **fi** (**if** spelled backwards).

Note that the *\c* in the **echo** command suppresses the new line that **echo** normally generates. This action leaves the cursor on the line immediately after the "Continue? " prompt. This is commonly used when prompting for user input.

The **test** command is the most common command to use with the **if** command. The ["$ans" = "n"] is the **test** command. It performs many types of file, string, and numeric logical tests, and if the condition is true, the test succeeds.

The syntax of the **test** command requires spaces around the **[ ]** or you will get a syntax error when the program runs. Also notice the double quotes around the response variable **$ans**. These are a strange anomaly with the **test** command. If the user presses only [[RETURN]] at the prompt without typing any other character, the value of **$ans** will be null. If we didn't have the quote marks around **$ans** in the **test** command, it would look like this when the value of **$ans** was substituted into the **test** command:

```
[ = "n" ]
```

This generates a "test: argument expected" error when you run the program. This is a very common mistake, and if you ever get this error, you should look for variables in your **test** commands with null values.

There is another form of the **if** command that is very common. It allows you to do one thing if a condition is met or do something else if not:

```
if [    ]              # if some condition is true
then
                       # do something
else
                       # otherwise do this
fi
```

There are many conditions that the **test** command can test. Table 28-1 shows some of the more common conditions for which you can test.

Table 28-1. test Command Conditions

String tests:	
["$a" == "string"]	True if $a is equal to "string"
["$a" != "string"]	True if $a is NOT equal to "string"
[-z "$a"]	True if $a is null (zero characters)
[-n "$a"]	True if $a is NOT null
Numeric tests:	
[$x -eq 1]	True if $x is equal to 1
[$x -ne 1]	True if $x is NOT equal to 1
[$x -lt 1]	True if $x is less than 1
[$x -gt 1]	True if $x is greater than 1

[$x -le 1]	True if $x is less than or equal to 1
[$x -ge 1]	True if $x is greater than or equal to 1
File tests:	
[-d $file]	True if $file is a directory
[-f $file]	True if $file is a file
[-s $file]	True if $file is a file with > 0 bytes
[-r $file]	True if $file is readable
[-w $file]	True if $file is writable
[-x $file]	True if $file is executable

Tests can be combined using *-a* to logically "AND" the tests together, *-o* to logically "OR" two tests, and *!* to "negate" a test. For example, this test statement is true only if the $**interactive** variable is set to true or **$file** is a directory:

```
[ "$interactive" = "TRUE" -o -d $file ]
```

This will be used in some of the upcoming example programs.

Here is a useful extension to the **gkill** program earlier shown. It checks to see that we have exactly one command-line argument before the program will attempt to do the processing. It uses a numeric test and the *$#* variable, which represents the number of command-line arguments. It should be inserted before any other lines of code in the **gkill** example given above:

```
# If we don't have exactly one command-line argument,

# write an error and exit.
if [ $# -ne 1 ]
then
    echo "Usage: $0 pattern"
    echo "Some pattern matching the processes to kill""
    echo "must be specified"
    exit 1 # Exit 1 terminates the program and tells the
           # calling shell that we had an error.
fi
```

Some other possible extensions to the **gkill** program might be to:

- Allow the user to specify a signal to use with the **kill** command. For example:

  ```
  gkill -9 ralph
  ```

  would find all of ralph's processes and then kill them with **kill -9**.

- Make sure that a valid message is printed if we can't find any processes to kill using the specified pattern.

This same type of command-line check is easily applied to the **psg** program to make sure you that have exactly one argument representing the pattern for which to search.

When you are reading user input, you may want to check if the user entered a value at all. If not, you would provide a reasonable default value. This is easily done with a variable modifier.

This example reads an answer ("ans") from the user and then checks its value using an **if** command:

```
echo "Do you really want to remove all of your files? \c"
read ans
if [ ${ans:-n} == y ]
then
    rm -rf *
fi
```

The **${ans:-n}** statement checks the value of **$ans**. If there is a value in **$ans,** use it in the command line. If the user simply pressed [[RETURN]] at the prompt, **$ans** will be null. In this case, **${ans:-n}** will evaluate to *n* when we do the comparison. Basically, in one small statement it says, "if the user did not provide an answer, assume they meant *n*."

There is another modifier that is often used:

```
${var:=default}
```

It returns the value of **var** if it is set; it returns the default if **var** is not set, and it will also assign the default as the value of **var** for future use.

Making Decisions with the case Statement

The **case** statement is another way to make decisions and test conditions in shell programs. It is most commonly used to check for certain patterns in command-line arguments. For example, if you wanted to determine whether the first command-line argument is an option (starts with a -), the **case** statement is the easiest way to do so. The **case** statement is also used to respond to different user input (such as asking the user to select a choice from a menu).

The **case** statement is probably one of the most complicated shell commands because of its syntax:

```
case pattern_to_match in
        pattern1)  cmdA
                   cmdB
                   ;;
        pattern2) cmdC
                   ;;
                ...
        *) cmdZ
                   ;;
esac
```

pattern_to_match is usually a shell variable that you are testing (like a command-line argument or a user response). If *pattern_to_match* matches *pattern1*, then commands *cmdA* and *cmdB* are executed. The **;;** separates this pattern's command list from the next pattern. In all cases, when **;;** is reached, the program jumps to the **esac** (**case** spelled backwards).

If *pattern_to_match* matches *pattern2*, then *cmdC* is executed and we jump to **esac**, the end of the **case** statement.

The * is provided so that if *pattern_to_match* did not match anything else, it will execute *cmdZ*. It's important to have a default action to handle the case where the user types an invalid entry.

For more robust pattern matching, any file name generation characters (*, [ ], ?) can be used to do special pattern matches. There is also a very useful way to check for multiple patterns in one line using the **|** symbol, which means logical "OR". Here's an example:

```
echo "Do you want to continue? (y/n) \c"
read ans
case $ans in
       y|Y) echo "Continuing"
               ...
            ;;
       n|N) echo "Done, Goodbye"
            exit
            ;;
       *) echo "Invalid input"
            ;;
esac
```

Here is another example where we are testing to see whether *$1* (the first command-line argument) is a valid option (a character we recognize that begins with a -):

```
case $1 in
        -l | -d) # Perform a listing
                 echo "All files in $HOME:\n"
                 ll -R $HOME | more
                 ;;
        -i) # -i means set to an interactive flag to true
            interactive="TRUE"
                 ;;
        *)  # Invalid input
            echo "$0: $1 is an invalid option"
            exit 1
            ;;
esac
```

A **case** statement is used in another example later in this chapter.

Looping

There are many times when you want to perform an action repeatedly. In the shell, there are two ways to do this:

1. The **for** loop takes a list of items and performs the commands in the loop once for each item in the list.
2. The **while** loop executes some commands (usually the **test** command) if that command executes successfully. (If the test condition is true, then the commands in the loop are executed, and the command is again executed to see whether we should loop again.)

The basic format of the **for** loop is:

```
for var in list_of_items
```

```
do
         cmdA
         cmdB
         cmdC
done
```

When the loop starts, the variable **var** has its value set to the first word in the *list_of_items* through which to loop. Then the three commands between the **do** and the **done** statements are executed. After the program reaches the **done** statement, it goes back to the top of the loop and assigns **var** to the next item in the list, executes the commands, and so on. The last time through the loop, the program continues with the next executable statement after the **done** statement.

The *list_of_items* can be any list of words separated by white space. You can type the words or use variables or command substitution to build the list. For example, let's say that we want to copy a new **.kshrc** file into the home directory of several users. A **for** loop is the easiest way to do so:

```
for name in ralph norton alice edith archie
do
          echo $name
          cp /tmp/.kshrc.new /users/$name/.kshrc
done
```

This example can be extended to copy certain files to several machines using the **rcp** command and verify that they got there using the **remsh** command:

```
for host in neptune jupiter mars earth sun
do
        echo $host
        rcp /etc/passwd /etc/hosts $host:/etc
        rcp /.profile $host:/.profile
        remsh $host ll /etc/passwd /etc/hosts /.profile
done
```

You can also process lists of files in the current directory using command substitution to generate the *list_of_items*:

```
for file in 'ls'
do
      if [ -r $file ]
      then
             echo "$file is readable
      fi
done
```

Note that **for file in *** would have done the same thing.

If you have a large list of things you would like to loop through and you don't want to type them on the command line, you can enter them in a file instead. Then, using the **cat** command and command substitution, you can generate the *list_of_items*:

```
for i in 'cat important_files'
do
    # do something with each of the files listed in the
    # important_files file.
done
```

The **for** loop, however, is most commonly used to process the list of command-line arguments (**$***):

```
for name in $*
do
     if [ ! -f $name -a ! -d $name ]
     then
       echo "$name is not a valid file or directory name"
     else
       # do something with the file or directory
     fi
done
```

The upcoming **trash** program contains a **for** loop that processes command-line arguments in a similar way.

The while Loop

The **while** loop has the following format:

```
while cmd1
do
      cmdA
      cmdB
      cmdC
done
```

cmd1 is executed first. If it executes successfully, then the commands between the **do** and the **done** statements are

executed. **cmd1** is then executed again; if successful, the commands in the loop are executed again; and so on. When **cmd1** fails, the program jumps past the **done** statement and resumes execution with the next executable statement.

Most of the time, the command executed in place of **cmd1** is the **test** command. You can then perform logical tests as described in the **if** section. If the test succeeds (is true), the commands in the loop are executed and the script tests the condition again. The **while** loop is useful if you have a fixed number of times you want the loop to run or if you want something to happen until some condition is met.



Let's now take a look at an example of using **while**. We'll run the **netstat** command once every 30 seconds a total of ten times, looking for the output of our primary LAN interface. In the case of the Solaris system on which we are running, this will be *le0*. Keep in mind that your primary LAN interface may have a different name, almost surely if you are not using a Solaris system. Let's first take a look at our short program.

This program displays the primary LAN interface (*le0*) statistics, which may be *lan0* or some other name depending on the UNIX variant you are using, using **netstat** ten times, once every 30 seconds:

```
#!/bin/ksh
i=1
while [ $i -le 10 ]
do
        print $i
        netstat -i | grep le0
        sleep 30
        let i=i+1
done
```

We increment *i* in a very simple fashion by adding one to it each time through the loop. We evaluate the value of *i* every time through the loop to determine whether it is less than or equal to (*le*) ten. If so, we run **netstat -i**.

Before we run this program, called **net1**, let's run the **netstat -i** command to view its output, and then run **net1**. Please keep in mind that the **netstat** command varies greatly among UNIX variants, so your output may look very different.

```
martyp $ netstat -i
Name  Mtu  Net/Dest      Address     Ipkts     Ierrs Opkts   Oerrs Collis Queue
lo0   8232 loopback      localhost   18927030  0     18927030 0     0      0
le0   1500 sunsys        sunsys      310417964 0     17193381 52064 7573173
sunsys:/home/martyp/shellprogs
martyp $ net1
1
le0   1500 sunsys        sunsys         310418018 0     17193388 52064 7573173
2
le0   1500 sunsys        sunsys         310418738 0     17193395 52064 7573173
3
le0   1500 sunsys        sunsys         310419579 0     17193401 52064 7573173
4
le0   1500 sunsys        sunsys         310420295 0     17193405 52064 7573173
5
le0   1500 sunsys        sunsys         310421099 0     17193446 52064 7573173
6
le0   1500 sunsys        sunsys         310421786 0     17193455 52064 7573173
7
le0   1500 sunsys        sunsys         310422425 0     17193462 52064 7573173
8
le0   1500 sunsys        sunsys         310423089 0     17193467 52064 7573173
9
le0   1500 sunsys        sunsys         310423749 0     17193471 52064 7573173
10
le0   1500 sunsys        sunsys         310424507 0     17193478 52064 7573173
sunsys:/home/martyp/shellprogs
martyp $
```

net1 produces just the output we want for *le0* every 30 seconds on our system, called *sunsys*. We also print out the value of *i* each time through the loop just so that we can see that we are incrementing it properly.

On HP-UX 11i systems the **netstat** command was modified to provide less information. The **lanadmin** command can be used to supply detailed information on *collisions* and *errors*. The following script runs **lanscan** to automatically produce a list of network interface cards on an 11i system and then produces a **lanadmin** output for each card.

```
#!/bin/sh
#@(#) $Header: $
#@(#) Description: Script to create network stats output with lanadmin
#@(#) $Source:  $
#@(#) $Locker: $
#@(#) $Log:  $
#@(#)
#  INPUT:
#  OUTPUT:
#  FILES ACCESSED:
#  NOTES:
```

```
#  AUTHOR:
#  REVISION HISTORY:
# Setup path
PATH=/bin:/usr/bin:/usr/sbin/:/usr/local/bin/;export PATH
#
# Determine OS Revision level
# run lanscan to determine network cards
#
OS_VERSION=$(uname -r | awk -F. '{print $2}')
if [ $OS_VERSION -lt 10 ]
then
  COMMAND1=/bin/landiag
else
  COMMAND1=/usr/sbin/lanadmin
fi
if [ -x $COMMAND1 ]
then
  COMMAND2=/usr/sbin/lanscan
  if [ -x  $COMMAND2 ]
  then
    if [ $OS_VERSION -lt 10 ]
    then
      for CARD in 'ls -1 /dev/lan* | awk -F "/" '{print $3}''
      do
        echo ""

$COMMAND1 -t <<- EOF 2> /dev/null | egrep "="
lan
name $CARD
display

quit
EOF
      done
#
# use field 3 of lanscan output to obtain card instance used for ppa below
#
    else
       for NMID in '$COMMAND2 | grep 0x | awk '{print $3}''
       do
         echo ""


$COMMAND1 -t <<- EOF 2> /dev/null | egrep "="
lan
ppa $NMID
display

quit
EOF
      done
    fi
  else
    print
    print
    print "The command \"$COMMAND2\" is not executable"
    print "No data from \"$COMMAND1\" will be collected"
    print
    print
  fi
else
    print
    print
    print "The command \"$COMMAND1\" is not executable"
    print
   print
fi
```

Field three of **lanscan** is used to provide the *ppa* information to **lanadmin** as the card instance for which a **lanadmin** output will be produced. The following is the **lanscan** output and result of running the script on a system with five network cards that are *ppa 0* through *1*:

```
# lanscan

Hardware Station           Crd Hdw   Net-Interface  NM  MAC       HP-DLPI DLPI
Path     Address           In# State NamePPA        ID  Type      Support Mjr#
0/0/0/0     0x001083FEDCB7 0   UP    lan0 snap0     1   ETHER     Yes     119
0/4/0/0/4/0 0x001083F72ED0 1   UP    lan1 snap1     2   ETHER     Yes     119
0/4/0/0/5/0 0x001083F72E9B 2   UP    lan2 snap2     3   ETHER     Yes     119
0/4/0/0/6/0 0x001083F77E22 3   UP    lan3 snap3     4   ETHER     Yes     119
0/4/0/0/7/0 0x001083F71E59 4   UP    lan4 snap4     5   ETHER     Yes     119
```

```
# networkscript.sh

PPA Number                     = 0
Description                   = lan0 Hewlett-Packard 10/100 TX Half-Duplex  TT = 1500
Type (value)                   = ethernet-csmacd(6)
MTU Size                       = 1500
Speed                          = 10000000
Station Address                = 0x1083fedcb7
Administration Status (value)  = up(1)
Operation Status (value)       = down(2)
Last Change                    = 10440
Inbound Octets                 = 0
Inbound Unicast Packets        = 0
Inbound Non-Unicast Packets    = 0
Inbound Discards               = 0
Inbound Errors                 = 0
Inbound Unknown Protocols      = 0
Outbound Octets                = 820
Outbound Unicast Packets       = 20
Outbound Non-Unicast Packets   = 0
Outbound Discards              = 0
Outbound Errors                = 0
Outbound Queue Length          = 0
Specific                       = 655367
Index                          = 1
Alignment Errors               = 0
FCS Errors                     = 0
Single Collision Frames        = 0
Multiple Collision Frames      = 0
Deferred Transmissions         = 0
Late Collisions                = 0
Excessive Collisions           = 0
Internal MAC Transmit Errors   = 0
Carrier Sense Errors           = 0
Frames Too Long                = 0
Internal MAC Receive Errors    = 0

PPA Number                     = 1
Description                   = lan1 Hewlett-Packard 10/100 TX Full-Duplex  TT = 1500
Type (value)                   = ethernet-csmacd(6)
MTU Size                       = 1500
Speed                          = 100000000
Station Address                = 0x1083f72ed0
Administration Status (value)  = up(1)
Operation Status (value)       = up(1)
Last Change                    = 11018
Inbound Octets                 = 2778542151
Inbound Unicast Packets        = 10008640
Inbound Non-Unicast Packets    = 14480929
Inbound Discards               = 0
Inbound Errors                 = 0
Inbound Unknown Protocols      = 12443000
Outbound Octets                = 3811379313
Outbound Unicast Packets       = 18378160
Outbound Non-Unicast Packets   = 50019
Outbound Discards              = 0
Outbound Errors                = 0
Outbound Queue Length          = 0
Specific                       = 655367
Index                          = 2
Alignment Errors               = 0
FCS Errors                     = 0
Single Collision Frames        = 0
Multiple Collision Frames      = 0
Deferred Transmissions         = 0
Late Collisions                = 0
Excessive Collisions           = 0
Internal MAC Transmit Errors   = 0
Carrier Sense Errors           = 0
Frames Too Long                = 0
Internal MAC Receive Errors    = 0

PPA Number                     = 2
Description                   = lan2 Hewlett-Packard 10/100 TX Full-Duplex  TT = 1500
Type (value)                   = ethernet-csmacd(6)
MTU Size                       = 1500
Speed                          = 100000000
Station Address                = 0x1083f72e9b
Administration Status (value)  = up(1)
Operation Status (value)       = up(1)
Last Change                    = 12223
Inbound Octets                 = 1053052283
Inbound Unicast Packets        = 660
Inbound Non-Unicast Packets    = 14478087
```

```
Inbound Discards                = 0
Inbound Errors                  = 0
Inbound Unknown Protocols       = 12442909
Outbound Octets                 = 5802138
Outbound Unicast Packets        = 43576
Outbound Non-Unicast Packets    = 43065
Outbound Discards               = 0
Outbound Errors                 = 0
Outbound Queue Length           = 0
Specific                        = 655367
Index                           = 3
Alignment Errors                = 0
FCS Errors                      = 0
Single Collision Frames         = 0
Multiple Collision Frames       = 0
Deferred Transmissions          = 0
Late Collisions                 = 0
Excessive Collisions            = 0
Internal MAC Transmit Errors    = 0
Carrier Sense Errors            = 0
Frames Too Long                 = 0
Internal MAC Receive Errors     = 0

PPA Number                      = 3
Description                    = lan3 Hewlett-Packard 10/100 TX Full-Duplex  TT = 1500
Type (value)                    = ethernet-csmacd(6)
MTU Size                        = 1500
Speed                           = 100000000
Station Address                 = 0x1083f77e22
Administration Status (value)   = up(1)
Operation Status (value)        = up(1)
Last Change                     = 13428
Inbound Octets                  = 943616591
Inbound Unicast Packets         = 9064639
Inbound Non-Unicast Packets     = 765175
Inbound Discards                = 0
Inbound Errors                  = 0
Inbound Unknown Protocols       = 39
Outbound Octets                 = 6454687
Outbound Unicast Packets        = 58769
Outbound Non-Unicast Packets    = 43040
Outbound Discards               = 0
Outbound Errors                 = 0
Outbound Queue Length           = 0
Specific                        = 655367
Index                           = 4
Alignment Errors                = 0
FCS Errors                      = 0
Single Collision Frames         = 0
Multiple Collision Frames       = 0
Deferred Transmissions          = 0
Late Collisions                 = 0
Excessive Collisions            = 0
Internal MAC Transmit Errors    = 0
Carrier Sense Errors            = 0
Frames Too Long                 = 0
Internal MAC Receive Errors     = 0

PPA Number                      = 4
Description                    = lan4 Hewlett-Packard 10/100 TX Full-Duplex  TT = 1500
Type (value)                    = ethernet-csmacd(6)
MTU Size                        = 1500
Speed                           = 100000000
Station Address                 = 0x1083f71e59
Administration Status (value)   = up(1)
Operation Status (value)        = up(1)
Last Change                     = 14633
Inbound Octets                  = 249984023
Inbound Unicast Packets         = 2628160
Inbound Non-Unicast Packets     = 765196
Inbound Discards                = 0
Inbound Errors                  = 0
Inbound Unknown Protocols       = 49
Outbound Octets                 = 3886863362
Outbound Unicast Packets        = 10894938
Outbound Non-Unicast Packets    = 425625
Outbound Discards               = 0
Outbound Errors                 = 0
Outbound Queue Length           = 0
Specific                        = 655367
Index                           = 5
Alignment Errors                = 0
FCS Errors                      = 0
Single Collision Frames         = 0
```

```
Multiple Collision Frames       = 0
Deferred Transmissions          = 0
Late Collisions                 = 0
Excessive Collisions            = 0
Internal MAC Transmit Errors    = 0
Carrier Sense Errors            = 0
Frames Too Long                 = 0
Internal MAC Receive Errors     = 0
```

Immediately before **lanscan** is run, our script prints the *ppa* number so that we know the network interface card for which the **lanscan** output is produced.

For all five network cards in our example there are no collisions or errors, as shown near the end of each **lanadmin** output, however, these fields are produced by **lanadmin**.

The **while** loop can also be used to process command-line arguments one at a time, using the number of command-line arguments and the **shift** command:

```
while [ $# -ne 0 ]
do
    case $1 in
    -*) # $1 must be an option because it starts with -
        # Add it to the list of options:
        opts="$opts $1"
        ;;
     *) # $1 must be an argument. Add it to the list of
        # command-line arguments:
        args="$args $1"
        ;;
  esac
  shift
done
```

The **shift** command shifts the remaining arguments in **$*** to the left by one position and decrements **$#**. What was the first argument (**$1**) is now gone forever; what was in **$2** is now in **$1**, etc. In the process of shifting command-line arguments, **$#** is also decremented to accurately reflect the number of arguments left in **$***.

You may want some commands to run until the user stops the program or until some stop condition is met. An infinite **while** loop is the best way to do so. For example, let's say that we are prompting users for some input and we will continue to prompt them until they give us valid input:

```
while true
do
    # prompt users and get their response
    echo "Enter yes or no: \c"
    read ans

    # Check whether the response is valid
    if [ "$ans" == "yes" -o "$ans" == "no" ]
    then
    # If it is valid, stop the looping
    break
 else
    # Otherwise print an error message and try it again
    # from the top of the loop
    echo "Invalid input, try again!\n"
  fi
done
# Now that we have valid input, we can process the user's
# request
  .
  .
  -
```

true is a special command that always executes successfully. The loop does not terminate unless the user stops the program by killing it or until a **break** command is executed in the loop. The **break** command will stop the loop.

Shell Functions

As you write shell programs, you will notice that there are certain sets of commands appear in many places within a program. For example, several times in a script, you may check user input and issue an appropriate message if input is invalid. It can be tedious to type the same lines of code in your program numerous times. It can be a nuisance if you later want to change these lines.

Instead, you can you can put these commands into a shell function. Functions look and act like a new command that can be used inside the script. Here's an example of a basic shell function:

```
# This is a function that may be called from anywhere within
# the program. It displays a standard usage error message
# and then exits the program.
```

```
print_usage()
{
    echo "Usage:"
    echo "To trash files: $0 [-i] files_to_trash..."
    echo "Display trashed files: $0 -d"
    echo "Remove all trashed files: $0 -rm"
    echo "Print this message: $0 -help"
    exit 1
}
```

print_usage is now a new command in your shell program. You can use it anywhere in this script.

Shell functions also have their own set of positional parameters (**$1-$9, $#**, and **$***), so you can pass them arguments just like any other command. The only nuance is that **$0** represents the name of the shell program, not the name of the function.

Earlier, we talked about arguments. When you type the name of the shell script, you can supply arguments that are saved in the variables **$1** through **$9**. The first ten words on the command line are directly accessible in the shell program using the special variables **$0 - $9**. The following shows how they work:

$0	The command name
$1	The first argument
$2	The second argument
$3	.
	.
	.
$9	The ninth argument

If you are not sure how many command-line arguments you may get when your program is run, there are two other variables that can help:

$#	The number of command-line arguments
$*	A space-separated list of all of the command-line arguments (which does *not* include the command name).

The variable **$*** is commonly used with a **for** loop to process shell script command lines with any number of arguments.

Figure 28-4 is a fairly complex program that exercises all the concepts we have covered so far. It is a **trash** program that removes files from their original locations. Instead of removing them permanently, it places them in a trash can in your home directory. This is a fairly robust program, but I'm sure that you can think of many extensions as you read through it.

Figure 28-4 trash Shell Program

```
#!/bin/ksh
# for Bourne use /bin/sh
# Program name: trash
# Usage:
#  To trash files:   trash [-i] file_names_to_trash ...
#  Display trashed files:    trash -d
#  Remove all trashed files: trash -rm
#  Print a help message:     trash -help

# This program takes any number of directory or file name
# arguments. If the argument is a file it will be removed
# from its current place in the file system and placed in the
# user's trash directory ($HOME/.trash). If the argument is a
# directory name the program will ask if the user really
# wants to trash the whole directory.
#
# This program also takes an -i (interactive) option. Like
# the rm command, if the -i is the first argument on the
# command line, the program stops and asks if each file
# named in the remaining arguments should be trashed.

#
# The -d (display) option shows the contents of the
# user's trashed files.
#
```

```
# The -help option displays a usage message for the user.
# The -rm (remove) option interactively
# asks the user if each file or directory in the trash
# directory should be removed permanently.
#
# The -h, -d and -rm options may not be used with
# any other command line arguments.

# Possible extensions:
# - Enhance the -rm option to remove a list of files
# from the trash directory from the command line.
# - Create a program to be run by cron once nightly to empty
# everyone's trash directory.

# This is a function that may be called from anywhere within
# the program. It displays a standard usage error message
# then exits the program.
print_usage()
{
  echo "Usage:"
  echo "To trash files: $0 [-i] file_names_to_trash ..."
  echo "Display trashed files:    $0 -d"
  echo "Remove all trashed files: $0 -rm"
  echo "Print this message:       $0 -help"
exit 1
}
# Make sure we have at least one command-line argument before
# we start.
if [ $# -lt 1 ]
then
    print_usage
fi

# If this flag is true then we need to do interactive
# processing.
interactive="FALSE"

# This is the name of the trash can.
trash_dir="$HOME/.trash"

# Make sure the trash directory exists before we go any
# further.
if [ ! -d $trash_dir ]
then
    mkdir $trash_dir
fi
# Sort out the command-line arguments.
case $1 in
   -help) # Print a help message.
      print_usage
      ;;
-d | -rm) # a -d or -rm were given
    # If it was not the only command-line argument
    # then display a usage message and then exit.
    if [ $# -ne 1 ]
    then
        print_usage
    fi

    # Otherwise do the task requested.
    if [ $1 == "-d" ]
    then
        echo "The contents of $trash_dir:\n"
       ls -l -R $trash_dir | more
    else
        # remove all files from the trash directory
        rm -rf $trash_dir/*
        # get any dotfiles too
        rm -rf $trash_dir/.[!.]*
    fi

    # Now we can exit successfully.
    exit 0
    ;;
-i) # If the first argument is -i ask about each file as it
    # is processed.
    interactive="TRUE"
    # Take -i off the command line so we know that the
    # rest of the arguments are file or directory names.

    shift
    ;;
```

```
 -*)# Check for an option we don't understand.
    echo "$1 is not a recognized option."
    print_usage
    ;;
    esac

# Just for fun we'll keep a count of the files that were
# trashed.
count=0

for file in $*
do
 # First make sure the file or directory to be renamed exists.
 # If it doesn't, add it to a list of bad files to be written
 # out later. Otherwise process it.
 if [ ! -f $file -a ! -d $file ]
 then
    bad_files="$bad_files $file"
 else
# If we are in interactive mode ask for confirmation
# on each file. Otherwise ask about directories.
if [ "$interactive" = "TRUE" -o -d $file ]
then
   # Ask the user for confirmation (default answer is no).
   if [ -d $file ]
   then
      echo "Do you want to trash the dir $file ? (y/n) n\b\c"
   else
      echo "Do you really want to trash $file ? (y/n) n\b\c"
   fi
   read doit

   # If the user answered y then do the move.
   # Otherwise print a message that the file was not touched.
   if [ "${doit:-n}" = y ]
   then
      mv -i $file $trash_dir
      echo "$file was trashed to $trash_dir"
      let count=count+1
   # for Bourne use: count='expr $count + 1'
   else
      echo "$file was not trashed"
   fi

  else # We are not in interactive mode, so just do it.
      mv -i $file $trash_dir
      let count=count+1
   #for Bourne use: count='expr  $count + 1'
  fi
fi
done

echo "$0: trashed $count item(s)"

if [ -n "$bad_files" ]
then
    echo "The following name(s) do not exist and \c"
    echo "could not be trashed:"
    echo "$bad_files"
fi

exit 0
```

Let's now run the **trash** program. The following example shows issuing just the program named **trash**, the next example shows issuing **trash -help**, the next example shows issuing **trash -i junk** to interactively remove the file **junk**, and the last example shows issuing **trash -d** to display files that have been removed with **trash** and are in the **/home/martyp/trash** directory:

```
martyp $ trash
Usage:
To trash files: trash [-i] file_names_to_trash ...
Display trashed files:    trash -d
Remove all trashed files: trash -rm
Print this message:       trash -help
martyp $ trash -help
Usage:
To trash files: trash [-i] file_names_to_trash ...
Display trashed files:    trash -d
Remove all trashed files: trash -rm
Print this message:       trash -help
martyp $ trash -i junk
Do you really want to trash junk ? (y/n) y
mv: overwrite /home/martyp/.trash/junk (yes/no)? yes
```

```
junk was trashed to /home/martyp/.trash
trash: trashed 1 item(s)
martyp $ trash -d
The contents of /home/martyp/.trash:

/home/martyp/.trash:
total 1364
-rw-------   1 martyp   staff     684808 May 30 05:31 core
-rwxrwxr-x   1 martyp   staff        631 May 30 06:45 file1
-rwxrwxr-x   1 martyp   staff         45 May 31 06:04 junk
martyp $
```

Notice that when we removed the file **junk, trash** asked us whether we wanted to overwrite a file by the same name that had been earlier removed with **trash** and placed in the **/home/martyp/trash** directory.

This program employs every concept we have covered in this shell programming chapter so far. You may want to take a close look at this program so that you can use these techniques in the programs you craft. I have also included comments for the lines that need to be changed to make this program work in the Bourne shell. The KornShell and Bourne shell are very similar, so you can use most of the same techniques when writing programs for these two shells.

awk in Shell Programs



awk is a very powerful symbolic programming language and data manipulation tool.



Simply stated, **awk** searches for patterns in lines of input (from standard input or from a file). For each line that matches the specified pattern, it can perform some very complex processing on that line. The code to actually process matching lines of input is a cross between a shell script and a C program.

Data manipulation tasks that would be very complex with combinations of **grep, cut,** and **paste** are very easily done with **awk**. Because **awk** is a programming language, it can also perform mathematical operations or check the input very easily, a task that is normally difficult with shells. It can even perform floating-point math.

The basic form of an **awk** program looks like this:

```
awk '/pattern_to_match/ { program to run }' input_file_names
```

Notice that the entire program is enclosed in single quotes. If no input file names are specified, **awk** reads from standard input (as from a pipe).

The *pattern_to_match* must appear between the **/** (slash) characters. The pattern is actually called a *regular expression*. Some common regular expression examples are shown shortly.

The program to execute is written in **awk** code, which looks something like C. The program is executed whenever a line of input matches the *pattern_to_match*. If */pattern_to_match/* does not precede the program in **{ }**, then the program is executed for every line of input.

awk works with fields of the input lines. Fields are words separated by white space. The fields in **awk** patterns and programs are referenced with **$**, followed by the field number. For example, the second field of an input line is **$2**. If you are using an **awk** command in your shell programs, the fields (**$1, $2**, etc.) are not confused with the shell script's positional parameters, because the **awk** variables are enclosed in single quotes and the shell ignores them.

But I don't want to just talk about it. Let's take a look at some examples.

This simple example lists just the terminals that are active on your system in which the terminal name is the second field of a **who** listing:

```
who | awk '{ print $2 }'
```

Here is an example of running **who** and then running this one-line command:

```
martyp $ who
thomfsu    console      Feb 22 15:21  (:0)
martyp     pts/3      May 31 06:03  (atlm0216.atl.hp.com)
martyp $ who  | awk '{print $2}'
console
pts/3
martyp $
```

This output shows only the active terminals on the system.

Note that **cut** could have done this also, but you would have had to know exactly which columns the terminal name occupied in the **who** output, as shown below:

```
martyp $ who
thomfsu    console      Feb 22 15:21    (:0)
martyp           pts/3                 May 31 06:03
(atlm0216.atl.hp.com)
martyp $ who | cut -c12-20
console
pts/3
martyp $
```

If the user or terminal name is longer than normal in any line, this command does not work. The **awk** example will work because it looks at fields, not columns.



Some trivia to wow your friends with at your next cocktail party: **awk** is the first letter of the last names of its authors - Alfred Aho, Peter Weinberger, and Brian Kernighan.

An HP-UX Logical Volume Manager Shell Program

In the section of Chapter 3 covering "Adding Disks," I provided an example of the HP-UX steps required to manually add a volume group consisting of five primary disks and five alternate disks. This manual procedure was done for one of ten volume groups. The remaining nine volume groups would have to be added using the same manual procedure, changing the names of the disks and volume groups as part of the manual procedure. Since manually performing these steps is highly error-prone, this is an ideal procedure to automate with a shell program.

To begin automating this procedure, let's take a look at the file that contains the physical disks to be used for the primary and alternate paths on the XP 256. The XP 256 is an advanced storage device that has in it the capability to failover to an alternate controller should the primary controller fail. The same sets of disks are connected to the primary and alternate controller, but the disks are given two different sets of device files. One set is for the disks when connected to the primary controller and the second set is for the same disks when connected to the alternate controllers. This is the same concept that you may have encountered if you are a ServiceGuard user. There is a set of disks connected through two different paths, so you must define the disks with different names depending on whether they are connected through the primary or alternate path. The following is a listing of the file **pri** containing the primary disks in groups of five:

```
c9t0d0 c9t0d1 c9t0d2 c8t0d0 c8t0d1
c7t0d0 c7t0d1 c7t0d2 c10t0d0 c10t0d1
c9t0d3 c9t0d4 c9t0d5 c8t0d3 c8t0d4
c7t0d3 c7t0d4 c7t0d5 c10t0d3 c10t0d4
c9t0d6 c9t0d7 c9t1d0 c8t0d6 c8t0d7
c7t0d6 c7t0d7 c7t1d0 c10t0d6 c10t0d7
c9t1d1 c9t1d2 c9t1d3 c8t1d1 c8t1d2
c7t1d1 c7t1d2 c7t1d3 c10t1d1 c10t1d2
c9t1d4 c9t1d5 c9t1d6 c8t1d4 c8t1d5
c7t1d4 c7t1d5 c7t1d6 c10t1d4 c10t1d5
```

Notice that in this listing the disks have been grouped in fives. Each group of five disks will constitute a volume group. There are a total of 10 groups of five disks that will be placed in volume groups *vgu01* through *vgu10*.

There will also be an alternate group of five disks. The alternate disks will be used in the event of a disk controller failover as described earlier. The following is a listing of the file **alt**, containing a list of alternate disks in groups of five:

```
c8t8d0  c8t8d1  c8t8d2  c9t8d0 c9t8d1
c10t8d0 c10t8d1 c10t8d2 c7t8d0 c7t8d1
c8t8d3  c8t8d4  c8t8d5  c9t8d3 c9t8d4
c10t8d3 c10t8d4 c10t8d5 c7t8d3 c7t8d4
c8t8d6  c8t8d7  c8t9d0  c9t8d6 c9t8d7
c10t8d6 c10t8d7 c10t9d0 c7t8d6 c7t8d7
c8t9d1  c8t9d2  c8t9d3  c9t9d1 c9t9d2
c10t9d1 c10t9d2 c10t9d3 c7t9d1 c7t9d2
c8t9d4  c8t9d5  c8t9d6  c9t9d4 c9t9d5
c10t9d4 c10t9d5 c10t9d6 c7t9d4 c7t9d5
```

There are a total of 10 groups of alternate disks shown in this listing that will also be part of the volume groups *vgu01* through *vgu10*. Using these primary and alternate disks that have been setup on the XP 256, we'll setup the appropriate volumes on the host system. In this example the host system is an HP-UX system.

Let's now cover the steps to manually create one of these volume groups. First, we'll create a physical volume for each of the disks in the volume group with the **pvcreate** command. Next we'll create a directory for the volume group with **mkdir**, then create a device special file for the volume group within the directory with **mknod**. This will setup the directory and special file required for the first of the 10 volume groups. Next, we'll create the volume group in which the five primary disks will be contained using **vgcreate**. We'll specify the first disk when we create the volume group and then include the other disks in the volume group with **vgextend**. Then we will extend the volume group with **vgextend** to include the five alternate disks. The final step is to create a single logical volume for the entire volume group. You might want to create several logical volumes within a volume group but in our example, we need only one logical volume that consumes the entire capacity of the volume group, which is 8755 physical extents. The following procedure is the list of manual steps to create the first volume group, which we'll then automate with a shell program:

```
# pvcreate /dev/rdsk/c9t0d0    # run for each of the 5 pri disks
```

```
# mkdir /dev/vgu01              # make dir for first vol group

# mknod /dev/vgu01/group -c 64 0x01000
                              # create special file with major
                                 and minor numbers shown
# vgcreate /dev/vgu01 /dev/dsk/c9t0d0
                              # Place first disk in volume group

# vgextend /dev/vgu01 /dev/dsk/c9t0d1
                              # extend volume with remaining four
                                 primary disks disks

# vgextend /dev/vgu01 /dev/dsk/c8t8d0
                              # extend volume group to include five
                                 alternte disks
# lvcreate -l 8755 /dev/vgu01  # creates lvol1 (lvol1 by default)
                                 that consumes all 8755 extents
```

We completed the procedure for only one disk and there are nine additional disks in this volume group. In addition, there are another nine volume groups for which this procedure must be completed. That is a total of an additional 99 disks for which various commands must be run. There is a lot of room for error with that much typing involved, so this is an ideal process to automate.

Since there are a primary set of disks and an alternate set of disks, we'll write a short program to automate each procedure. The following program performs all of the steps required to create a physical volume for each disk, create a volume group, and include the primary disks in it:

```
#!/bin/ksh
set -x                  ;set tracing on

  vgnum=$1              ;first item on each line is the volume group no.
  shift                 ;shift to get to first disk


  for i in $*           ;run pvcreate for every disk name in first line
  do
     pvcreate /dev/rdsk/$i
  done


  reada                 ;pause program to view what has been run


  mkdir /dev/vgu$vgnum                        ;mkdir for volume group
  mknod /dev/vgu$vgnum/group c 0x$(vgnum)0000  ;mknod for volume group
  vgcreate /dev/vgu$vgnum /dev/dsk/$1          ;vgcreate 1st disk in vg


  shift             ;shift over to second disk

  for i in $*       ;extend volume group to include remaining four disks
  do
     vgextend /dev/vgu$vgnum /dev/dsk/$i
  done


  lvcreate -l 8755 /dev/vgu$vgnum     ;create single log vol for entire vg
```

I use *set -x* in this file to turn on execution tracing. I always do this when first debugging a shell program so you can see the lines in the shell program as they are executed. The line being executed will appear with a "+" in front of it, followed by the what you would normally see when the program is run. The *read a* is a way of pausing the program to wait for input so I can review what has been run to that point of the program.

In order for this program to run, we have to slightly modify the file containing the primary disk devices and add the volume group number to the beginning of each line. In addition I decided to call the program from the file that has the primary disks in it and operate on one line of disks at a time. The following listing

shows the updated file containing the name of the shell program in the previous listing (**vg.sh**), followed by the volume group number and then the list of five primary disks names for each volume group:

```
#vg.sh 01 c9t0d0 c9t0d1 c9t0d2 c8t0d0 c8t0d1
#read a
#vg.sh 02 c7t0d0 c7t0d1 c7t0d2 c10t0d0 c10t0d1
#read a
#vg.sh 03 c9t0d3 c9t0d4 c9t0d5 c8t0d3 c8t0d4
#read a
#vg.sh 04 c7t0d3 c7t0d4 c7t0d5 c10t0d3 c10t0d4
#read a
#vg.sh 05 c9t0d6 c9t0d7 c9t1d0 c8t0d6 c8t0d7
#read a
#vg.sh 06 c7t0d6 c7t0d7 c7t1d0 c10t0d6 c10t0d7
#read a
#vg.sh 07 c9t1d1 c9t1d2 c9t1d3 c8t1d1 c8t1d2
#read a
#vg.sh 08 c7t1d1 c7t1d2 c7t1d3 c10t1d1 c10t1d2
#read a
#vg.sh 09 c9t1d4 c9t1d5 c9t1d6 c8t1d4 c8t1d5
#read a
#vg.sh 10 c7t1d4 c7t1d5 c7t1d6 c10t1d4 c10t1d5
```

The *read a* between lines of this file will pause and wait for you to enter a *Return* before the next line will be executed. I did this in case I decided to run several lines and I wanted to check the results between the execution of lines.

We can now uncomment the first line of the file and type the file name **pri** and it will call **vg.sh** and run the program (you have to give appropriate permissions to the files and make sure that both **vg.sh** and **pri** are executable). I like to run such files one line at a time and check the volume groups as they are created. The script is written to run one line at a time, but is easily modifiable to run all 10 lines.

We need to do much less work with the alternate disk names. The physical volumes have already been created, and the volume group and single logical volume already setup in **vg.sh**. We'll create another script called **vga.sh**, "a" for alternate, in which we'll extend the volume group to include the alternate name for each disk. This script is shown in the listing below:

```
#!/bin/ksh
set -x              ;set tracing on

  vgnum=$1          ;first item on each line is the volume group number
  shift             ;shift to get to first disk


  for i in $*       ;extend vol group to include all five disks on line

     vgextend /dev/vgu$vgnum /dev/dsk/$i

  done
```

This script performs only the task of extending the volume group *vgnum* to include all five disks that appear on the line. Much like the file **pri**, the file **alt** will call the script **vga.alt**, as shown in the following listing:

```
#vga.sh 01 c8t8d0  c8t8d1  c8t8d2  c9t8d0 c9t8d1
#vga.sh 02 c10t8d0 c10t8d1 c10t8d2 c7t8d0 c7t8d1
#vga.sh 03 c8t8d3  c8t8d4  c8t8d5  c9t8d3 c9t8d4
#vga.sh 04 c10t8d3 c10t8d4 c10t8d5 c7t8d3 c7t8d4
#vga.sh 05 c8t8d6  c8t8d7  c8t9d0  c9t8d6 c9t8d7
#vga.sh 06 c10t8d6 c10t8d7 c10t9d0 c7t8d6 c7t8d7
#vga.sh 07 c8t9d1  c8t9d2  c8t9d3  c9t9d1 c9t9d2
#vga.sh 08 c10t9d1 c10t9d2 c10t9d3 c7t9d1 c7t9d2
#vga.sh 09 c8t9d4  c8t9d5  c8t9d6  c9t9d4 c9t9d5
#vga.sh 10 c10t9d4 c10t9d5 c10t9d6 c7t9d4 c7t9d5
```

You would uncomment the line for which you want to run the script. Again, you could run all ten lines, but I like to check what has taken place after each line has been run. You could add the *read a* between lines of this file if you wanted to run several lines and have a pause between them to check the results.

These two scripts automate a lot of typing. There are 100 disks for which commands must be run as well as other Logical Volume Manager commands. This is the type of HP-UX system administration task that is ideally suited for shell programming.

I completed the steps that had to be run for the additional disks to complete the work such as the **vgcreate** for the addtional four disks and the **vgextend** for an additional nine disk devices. I included only the first

disk in the examples so you could see the initial steps that had to be taken.

We don't have to setup any RAID levels within the primary or alternate volume because this is being done internally to the XP 256.

The following **vgdisplay** listing shows the disks we setup for volume group *vgu01* with both the groups of five primary and alternate disks:

```
# vgdisplay /dev/vgu01 -v


VG Name                     /dev/vgu01
VG Write Access             read/write
VG Status                   available
Max LV                      255
Cur LV                      1
Open LV                     1
Max PV                      16
Cur PV                      5
Act PV                      5
Max PE per PV               1751
VGDA                        10
PE Size (Mbytes)            4
Total PE                    8755
Alloc PE                    8755
Free PE                     0
Total PVG                   0
Total Spare PVs             0
Total Spare PVs in use      0

   --- Logical volumes ---
   LV Name                      /dev/vgu01/lvol1
   LV Status                    available/syncd
   LV Size (Mbytes)             35020
   Current LE                   8755
   Allocated PE                 8755
   Used PV                      5


   --- Physical volumes ---
   PV Name                      /dev/dsk/c9t0d0
   PV Name                      /dev/dsk/c8t8d0Alternate Link
   PV Status                    available
   Total PE                     1751
   Free PE                      0

   PV Name                      /dev/dsk/c9t0d1
   PV Name                      /dev/dsk/c8t8d1Alternate Link
   PV Status                    available
   Total PE                     1751
   Free PE                      0

   PV Name                      /dev/dsk/c9t0d2
   PV Name                      /dev/dsk/c8t8d2Alternate Link
   PV Status                    available
   Total PE                     1751
   Free PE                      0

   PV Name                      /dev/dsk/c8t0d0
   PV Name                      /dev/dsk/c9t8d0Alternate Link
   PV Status                    available
   Total PE                     1751
   Free PE                      0

   PV Name                      /dev/dsk/c8t0d1
   PV Name                      /dev/dsk/c9t8d1Alternate Link
   PV Status                    available
   Total PE                     1751
   Free PE                      0
```

There are some points of interest to cover in this **vgdisplay**. The first is that there is a primary and alternate path to the same disk as we had defined them earlier. For instance, the first disk in the volume group has a primary path name of *c9t0d0* and an alterate path name of *c8t8d0*. Next, both the volume group *vgu01* and the only logical volume in it *lvol1* consist of a total of 8755 PE or physical extents (the size

of a volume group is PE x PE size or 8755 x 4MB in our case).

We should also check one logical volume on *vgu01* called *lvol1*. We can check the parameters of this logical volume with the **lvdisplay** command, as shown in the following example:

```
# lvdisplay -v /dev/vgu01/l*

--- Logical volumes ---
LV Name                     /dev/vgu01/lvol1
VG Name                     /dev/vgu01
LV Permission               read/write
LV Status                   available/syncd
Mirror copies               0
Consistency Recovery        MWC
Schedule                    parallel
LV Size (Mbytes)            35020
Current LE                  8755
Allocated PE                8755
Stripes                     0
Stripe Size (Kbytes)        0
Bad block                   on
Allocation                  strict
IO Timeout (Seconds)        default

   --- Distribution of logical volume ---
   PV Name            LE on PV  PE on PV
   /dev/dsk/c9t0d0    1751      1751
   /dev/dsk/c9t0d1    1751      1751
   /dev/dsk/c9t0d2    1751      1751
   /dev/dsk/c8t0d0    1751      1751
   /dev/dsk/c8t0d1    1751      1751

   --- Logical extents ---
   LE    PV1                 PE1   Status 1
   00000 /dev/dsk/c9t0d0     00000 current
   00001 /dev/dsk/c9t0d0     00001 current
   00002 /dev/dsk/c9t0d0     00002 current
   00003 /dev/dsk/c9t0d0     00003 current
   00004 /dev/dsk/c9t0d0     00004 current
   00005 /dev/dsk/c9t0d0     00005 current
   00006 /dev/dsk/c9t0d0     00006 current
   00007 /dev/dsk/c9t0d0     00007 current
   00008 /dev/dsk/c9t0d0     00008 current
   00009 /dev/dsk/c9t0d0     00009 current
   00010 /dev/dsk/c9t0d0     00010 current
   00011 /dev/dsk/c9t0d0     00011 current
   00012 /dev/dsk/c9t0d0     00012 current
   00013 /dev/dsk/c9t0d0     00013 current
   00014 /dev/dsk/c9t0d0     00014 current
   00015 /dev/dsk/c9t0d0     00015 current
   00016 /dev/dsk/c9t0d0     00016 current
   00017 /dev/dsk/c9t0d0     00017 current
   00018 /dev/dsk/c9t0d0     00018 current
   00019 /dev/dsk/c9t0d0     00019 current
   00020 /dev/dsk/c9t0d0     00020 current
   00021 /dev/dsk/c9t0d0     00021 current
   00022 /dev/dsk/c9t0d0     00022 current
   00023 /dev/dsk/c9t0d0     00023 current
   00024 /dev/dsk/c9t0d0     00024 current
   00025 /dev/dsk/c9t0d0     00025 current
   00026 /dev/dsk/c9t0d0     00026 current
   00027 /dev/dsk/c9t0d0     00027 current
   00028 /dev/dsk/c9t0d0     00028 current
   00029 /dev/dsk/c9t0d0     00029 current
   00030 /dev/dsk/c9t0d0     00030 current
   00031 /dev/dsk/c9t0d0     00031 current
   00032 /dev/dsk/c9t0d0     00032 current
              -
              -
              -
   08733 /dev/dsk/c8t0d1     01729 current
   08734 /dev/dsk/c8t0d1     01730 current
   08735 /dev/dsk/c8t0d1     01731 current
   08736 /dev/dsk/c8t0d1     01732 current
   08737 /dev/dsk/c8t0d1     01733 current
```

```
   08738 /dev/dsk/c8t0d1    01734 current
   08739 /dev/dsk/c8t0d1    01735 current
   08740 /dev/dsk/c8t0d1    01736 current
   08741 /dev/dsk/c8t0d1    01737 current
   08742 /dev/dsk/c8t0d1    01738 current
   08743 /dev/dsk/c8t0d1    01739 current
   08744 /dev/dsk/c8t0d1    01740 current
   08745 /dev/dsk/c8t0d1    01741 current
   08746 /dev/dsk/c8t0d1    01742 current
   08747 /dev/dsk/c8t0d1    01743 current
   08748 /dev/dsk/c8t0d1    01744 current
   08749 /dev/dsk/c8t0d1    01745 current
   08750 /dev/dsk/c8t0d1    01746 current
   08751 /dev/dsk/c8t0d1    01747 current
   08752 /dev/dsk/c8t0d1    01748 current
   08753 /dev/dsk/c8t0d1    01749 current
   08754 /dev/dsk/c8t0d1    01750 current
```

This listing has been abbreviated where the three dots are shown. Only the beginning of the first disk and end of the last disk are shown. The **lvdisplay** does indeed show the five primary disks of which the logical volume is comprised.

The final step is to place a file system on the logical volume we have setup in *vgu00*. This is a task for which SAM is ideally suited. I completed this procedure using SAM in Chapter 8, so you can refer to it if you want to see the SAM screen shots.

The next section is a short introduction to C shell programming.

[Team LiB]

PREVIOUS NEXT

C Shell Programming

Although shell programming techniques apply to all shells, generally speaking, there are some differences exist between the C shell and the KornShell. If you are using the C shell, I want you to get off to a quick start, so I'll cover the basics of C shell programming in this section. I'll cover each shell programming technique briefly and use basic examples to help reinforce each technique. In all of the following shell programs, any line beginning with a "**#**" is a comment. This is true except for the very first line of a shell program, in which the shell the script is written for is executed. In all of the following programs, the C shell is executed with **#!/bin/csh**, which is the path of the C shell on the Solaris system used in the examples.

Command Substitution

The shell variables earlier covered can be used to save the output from a command. You can then use these variables when issuing other commands. The following shell program executes the **date** command and saves the results in the variable **d**. The variable **d** is then used within the **echo** command in the program **cdate**:

```
#!/bin/csh
# program "today" that provides the date
set d='date +%x'
echo "Today's date is $d"
```

When we run **cdate**, the following is produced:

```
martyp $ cdate
Today's date is 06/01/00
martyp $
```

The "+%x" in the above example produces the current date. Command substitution of this type is used in several upcoming shell scripts.

Reading User Input

Two common methods help you read user input to shell programs. The first is to prompt the user for information, and the second is to provide arguments to shell programs.

To begin, I'll cover prompting a user for information. A character, word, or sentence can be read into a variable. The following example first shows prompting the user for a word, and then a sentence:

```
#!/bin/csh
echo "Please enter your name:"
set name = $<
echo "hello, $name"
echo "Please enter your favorite quote:"
set quote = $<
echo "Your favorite quote is:"
echo $quote
```

Here is an example of running this program:

```
martyp $ userinput
Please enter your name:
Marty
hello, Marty
Please enter your favorite quote:
Creating is the essence of life.
Your favorite quote is:
Creating is the essence of life.
martyp $
```

Using this technique, you can prompt a user for information in a shell program. This technique is used in an upcoming program.

You can also enter command-line arguments. When you type the name of the shell script, you can supply arguments that are saved in the variables **$1** through **$9**. The first ten words on the command line are directly accessible in the shell program using the special variables **$0 - $9**. This is how they work:

$0	The command name
$1	The first argument
$2	The second argument
$3	.
	.
	.
$9	The ninth argument

If you are not sure how many command-line arguments you may get when your program is run, there are two other variables that can help:

$#	The number of command-line arguments
$*	A space-separated list of all the command-line arguments (which does *not* include the command name)

The variable **$*** is commonly used with the **for** loop (soon to be explained) to process shell script command lines with any number of arguments.

The following script changes to the specified directory (**$1**) and searches for the specified pattern (**$2**) in the specified file (**$3**):

```
#!/bin/csh
# search
# Usage: search directory pattern file
echo " "
cd $1             # change to search dir and
grep -n "$2" $3   # search for $2 in $3
echo " "          # print line
endif
```



grep is used to search a file for a pattern and print the line in which the pattern was found. **awk**, which was earlier covered, can be used to pick out a specific field within a line.

Here is an example of the **search** program:




```
martyp $ search /home/martyp/shellprogs awk ifstat

12:# as one command so it can be easily piped to awk.
18:awk 'BEGIN { printf "%10s%10s%10s%10s%10s\n", "ipkts",
38:' # End of the awk program.

martyp $
```

In this example, we run **search** in the directory **/home/martyp/shellprogs**, looking for the pattern **awk** in the file **ifstat**. The result of this search produces three lines in the file **ifstat**, in which **awk** appears.

These are lines number 12, 18, and 38.

In the next section, we'll expand this program somewhat to include testing and branching.

Testing and Branching

There are many kinds of decision-making that your shell programs can perform. **if** provides the flexibility to make decisions and take the appropriate action. Let's expand the **search** script to verify that three arguments have been provided:

```
#!/bin/csh
# search
# Usage: search directory pattern files

if ($#argv != 3) then            # if < 3 args provided

        echo "Usage: search directory pattern files"
                                 # then print Usage
else
        echo " "                 # else print line and
        cd $1                    # change to search dir
        grep -n "$2" $3          # search for $2 in $3
        echo " "                 # print line
endif
```

This program is called **search1**. We run this program using the same arguments as we did with the **search** program; however, **search1** is enhanced to provide a *usage* message if we don't provide arguments when we run it. The following example shows running **search1**:

```
martyp $ search1
Usage: search directory pattern files
martyp $ search1 /home/martyp/shellprogs awk llsum

12:# drwxrwxrwx 2 gerry  aec  24  Mar 21 18:25   awk_ex
15:# awk field numbers:
18:awk ' BEGIN { x=i=0; printf "%-16s%-10s%8s%8s\n",\

martyp $
```



On the first attempt to run **search1**, we provided no arguments. The program checked to see whether we provided fewer than three arguments and produced the *Usage* message, because it found fewer than three arguments. Upon seeing the *Usage* message, it became clear how to use the program and we provided the required three arguments on the next attempt to run the program. In this example, we run **search1** in the directory **/home/martyp/shellprogs**, looking for the pattern **awk** in the file **llsum**. The result of this search produces three lines in the file **llsum**, in which **awk** appears. These are lines number 12, 15, and 18.

Here are four commonly used forms of **if**:

1)
```
if (expression) command
```

2)
```
if (expression) then
                command(s)
endif
```

3)
```
if (expression) then
                command(s)
        else
                command(s)
endif
```

4)
```
if (expression) then
                command(s)
      [else if expression) then
                command(s)]
```

```
                .
                .
                .
        [else
                command(s)]
endif
```

There are many operators that can be used in the C shell to compare integer values, such as the < used in the previous example. Here is a list of operators:

>	greater than
<	less than
>=	greater than or equal to
<=	less than or equal to
==	equal to
!=	not equal to

Looping

The C shell supports a number of techniques to support looping, including:

1. The **foreach** loop, which takes a list of items and performs the commands in the loop once for each item in the list.
2. The **while** loop, which executes a command (such as the **test** command) if the command executes successfully.

The format of the **foreach** loop is

```
foreach name (list)
        command(s)
end
```

The following example uses a **foreach** loop to test whether or not the systems in the **/etc/hosts** file are connected to the local host.

```
#!/bin/csh
#Program name: csh_hostck

#This program will test connectivity to all other hosts in
#your network listed in your /etc/hosts file.

# It uses the awk command to get the names from the hosts file
#and the ping command to check connectivity.

#Note that we use /bin/echo because csh echo doesn't support
#escape chars like \t or \c which are used in the
#foreach loop.

#Any line in /etc/hosts that starts with a number represents
#a host entry. Anything else is a comment or a blank line.

#Find all lines in /etc/hosts that start with a number and
#print the second field (the hostname).

set hosts='awk '/^[1-9]/ { print $2 }' /etc/hosts'
                # grave on outside, single quote on inside

     /bin/echo "Remote host connection status:"

foreach sys ($hosts)
     /bin/echo "$sys - \c"
                # send one 64 byte packet and look for
```

```
                # the"is alive" message in
                # the output that indicates success.
                # messages vary between UNIX variants.

     ping $sys 64 1 | grep "is alive" > /dev/null
     if ( $status == 0 ) then
               echo "OK"
     else
               echo "DID NOT RESPOND"
     endif
end
```



The crazy-looking line with **awk** is used to obtain the name of remote hosts from the **/etc/hosts** file. The **foreach** loop takes all of the items in the list, the hosts in this case, and checks the status of each.

The **hosts** file on this system has three entries: the localhost, the LAN interface, and a DNS system. When we run the program in the following example, we expect to see a result for the testing of all three entries:

```
martyp $ csh_hostck
Remote host connection status:
localhost - OK
sunsys - OK
dnssrv1 - OK
martyp $
```

All three entries in the **hosts** file have been evaluated with **ping** and produce a status of *OK*. When hardcoding information into scripts, such as the path of **ping** and the result you get from the **ping** command, please keep in mind that these may vary among different UNIX variants. One of the reasons you want to liberally comment your shell programs is to make them easy to modify under such circumstances.

You could use the **while** loop to execute commands for some number of iterations. The **while** loop is in the following format:

```
while (expression)
       command(s)
end
```



The following program, called **netcheck**, runs **netstat** at the desired interval, and prints out the heading once and the status of *le0* nine times:

```
#!/bin/csh
# program to run netstat at every specified interval
# Usage: netcheck interval

set limit=9               # set limit on number times
                          # to run netstat

echo " "
netstat -i | grep Name    # print netstat line with headings
set count=0
while ($count<$limit)     # if limit hasn't reached
                          # limit run netstat
          netstat -i | grep le0
          sleep $1        # sleep for interval
                          # specified on command line
          @ count++       # increment limit
end
echo "count has reached $limit, run netcheck again to see le0 status"
```

Here is an example run of the **netcheck** program:

```
martyp $ netcheck 3
```

```
Name  Mtu  Net/Dest     Address       Ipkts     Ierrs Opkts    Oerrs Collis Queue
le0   1500 sunsys        sunsys         314374989  0     17252200 52135  7580906
le0   1500 sunsys        sunsys         314375038  0     17252203 52135  7580906
le0   1500 sunsys        sunsys         314375114  0     17252206 52135  7580906
le0   1500 sunsys        sunsys         314375185  0     17252209 52135  7580906
le0   1500 sunsys        sunsys         314375257  0     17252212 52135  7580906
le0   1500 sunsys        sunsys         314375332  0     17252215 52135  7580906
le0   1500 sunsys        sunsys         314375444  0     17252218 52135  7580906
le0   1500 sunsys        sunsys         314375508  0     17252221 52135  7580906
le0   1500 sunsys        sunsys         314375588  0     17252224 52135  7580906
count has reached 9, run netcheck again to see le0  status
martyp $
```



The output of **netcheck** produces nine **netstat** outputs at the three-second interval we had specified. Keep in mind that you may have to modify such information as the name of the LAN interface when you use this program on your system.

This program increments the expression with the following:

```
@ count++
```

If the expression is true, then the command(s) will execute. The @count++ is an assignment operator in the form of:

```
@ variable_name operator expression
```

In this case, the variable is first assigned with "=" and is later auto incremented (++). There are a number of operations that can be performed on the variable, as described in Table 28-2:

Table 28-2. Assignment Operators

Operation	Symbol	Example with count = 100	Result
store value	=	@count=100	100
auto increment	++	@count++	101
auto decrement	--	@count--	99
add value	+=	@count+=50	150
subtract value	-=	@count-=50	50
multiply by value	*=	@count*=2	200
divide by value	/=	@count/2	50

There are also comparison operators, such as the "<" used in the example, as well as arithmetic, bitwise, and logical operators. As you craft more and more shell scripts, you will want to use all these operators.

There are a set of test conditions related to files that are useful when writing shell scripts that use files. Using the format **- operator filename**, you can use the tests in Table 28-3.

Table 28-3. Operator File Name Tests

Operator	Meaning
r	read access

w	write access
x	execute access
o	ownership
z	zero length
f	file, not a directory
d	directory, not a file

The following program, called **filetest,** uses these operators to test the file **.profile**. Because **.profile** is not executable, of zero length, or a directory, I would expect **filetest** to find these false.

Here is a long listing of **.profile**:

```
martyp $ ls -al .profile
-rw-r--r--   1 martyp   staff        594 May 21 09:29 ../.profile
martyp $
```

Here is a listing of the shell script **filetest**:

```
#!/bin/csh
#  Program to test file $1

if (-e $1) then
             echo "$1 exists"
             else
             echo "$1 does not exist"
endif

if (-z $1) then
             echo "$1 is zero length"
             else
             echo "$1 is not zero length"
endif

if (-f $1) then
             echo "$1 is a file"
             else
             echo "$1 is not a file"
endif

if (-d $1) then
             echo "$1 is a directory"
             else
             echo "$1 is not a directory"
endif

if (-o $1) then
             echo "you own $1 "
             else
             echo "you don't own $1 "
endif

if (-r $1) then
             echo "$1 is readable"
             else
             echo "$1 is not readable"
endif

if (-w $1) then
             echo "$1 is writable"
             else
             echo "$1 is not writable"
endif

if (-x $1) then
             echo "$1 is executable"
             else
             echo "$1 is not executable"
endif
```

This is a somewhat extreme example of testing a file; however, I wanted to include many of the file tests.

Here is the output of **filetest** using **.profile** as input:

```
martyp $ filetest /home/martyp/.profile
/home/martyp/.profile exists
/home/martyp/.profile is not zero length
/home/martyp/.profile is a file
/home/martyp/.profile is not a directory
you own /home/martyp/.profile
/home/martyp/.profile is readable
/home/martyp/.profile is writable
/home/martyp/.profile is not executable
martyp $
```

The result of having run **filetest** on **.profile** produces the file test results that we expect.

The next section covers a way of making decisions with **switch**.

Decision Making with switch

You can use **switch** to make decisions within a shell program. You can use **switch** to test command-line arguments or interactive input to shell programs as shown in the upcoming example. If, for example, you wanted to create a menu in a shell program and you needed to determine which option a user selected when running this shell program, you can use **switch**.

The syntax of **switch** looks like the following:

```
switch (pattern_to_match)

  case pattern1
             commands
             breaksw

  case pattern2
            commands
            breaksw

  case pattern 3
           commands
           breaksw

    default
           commands
           breaksw
endsw
```

pattern_to_match is the user input that you are testing, and if it is equal to *pattern1,* then the commands under *pattern1* are executed. If *pattern_to_match* and *pattern2* are the same, then the commands under *pattern2* will be executed, and so on. If no match occurs between *pattern_to_match* and one of the case statement patterns, then the default is executed. The following program allows you to pick from between two scripts on its menu. These are the two shell programs we crafted earlier in this chapter. You can expand this script to include as many of your programs as you wish. This example uses **switch**:

```
#!/bin/csh
# Program pickscript to run some of
# the C shell scripts we've created
# Usage: pickscript
echo " ---------------------------------------------"
echo "                  Sys Admin Menu              "
echo "----------------------------------------------"
echo " "
echo " 1            netcheck for network interface  "
echo " "
echo " 2            hostck to check connection      "
echo "                to hosts in /etc/hosts        "
echo "                                              "
echo " ---------------------------------------------"
echo " "
echo " Please enter your selection -> \c"

set pick = $<      # read input which is number of script
```

```
echo " "
switch ($pick)     # and assign to variable pick

             case 1         # if 1 was selected execute this
             $HOME/cshscripts/netcheck 5
             breaksw

             case 2         # if 2 was selected, execute this
             $HOME/cshscripts/hostck
             breaksw

             default
             echo "Please select 1 or 2 next time"
             breaksw

endsw
```

This program allows us to select from between two scripts to run. Let's take a look at an example of running this program:

```
martyp $ pickscript
 ------------------------------------------
                  Sys Admin Menu
------------------------------------------

 1          netcheck for network interface

 2          hostck to check connection
              to hosts in /etc/hosts

 ------------------------------------------

 Please enter your selection
1


Name  Mtu  Net/Dest      Address        Ipkts  Ierrs Opkts  Oerrs Collis Queue
le0   1500 sunsys       sunsys         314996747 0     17261251 52135 7580952
le0   1500 sunsys       sunsys         314996862 0     17261256 52135 7580952
le0   1500 sunsys       sunsys         314997189 0     17261266 52135 7580952
le0   1500 sunsys       sunsys         314997319 0     17261269 52135 7580952
le0   1500 sunsys       sunsys         314997420 0     17261272 52135 7580952
le0   1500 sunsys       sunsys         314997630 0     17261275 52135 7580952
le0   1500 sunsys       sunsys         314997774 0     17261278 52135 7580952
le0   1500 sunsys       sunsys         314997904 0     17261281 52135 7580952
le0   1500 sunsys       sunsys         314998020 0     17261284 52135 7580952
count has reached 9, run netcheck again to see lan0 status
martyp $
```

We selected option *1* when we ran **pickscript**. You can use this program as the basis for running the many shell programs you may write.

Debugging C Shell Programs

When you begin C shell programming, you'll probably make a lot of simple syntax-related errors. Using the *-n* option to **csh**, you can have the C shell check the syntax of your program without executing it. I also use the *-v* option to produce a verbose output. This can sometimes lead to too much information, so I start with *-v* and if there is too much feedback results, I eliminate it.

The following example is the earlier **search1** program, which includes a check that three arguments have been provided. When checking to see that **$#argv** is equal to 3, I left off the right parenthesis. Here is the listing of the program and a syntax check showing the error:

```
martyp $ cat search1
#!/bin/csh
# search
# Usage: search directory pattern files

if ($#argv !3 3 then             # if < 3 args provided

        echo "Usage: search directory pattern files"
                               # then print Usage
```

```
else
        echo " "                # else print line and
        cd $1                   # change to search dir
        grep -n "$2" $3         # search for $2 in $3
        echo " "                # print line
endif

martyp $ csh -nv search1
if ( $#argv != 3 then
Too many ('s
martyp $
```

The **csh -nv** has performed a syntax check with verbose output. First, the line in question is printed and then an error message that tells you what is wrong with the line. In this case, it is clear that I have left off a right parenthesis.

After fixing the problem, I can run the program with the *-x*, which causes all commands to be echoed immediately before execution. The following example shows a run of the **search** program:

```
martyp $ csh -xv search1 shellprogs grep csh_hostck

if ( $#argv != 3 ) then
if ( 3 != 3 ) then

echo " "
echo

cd $1
cd /home/martyp/shellprogs
grep -n "$2" $3
grep -n grep csh_hostck
25:     /usr/sbin/ping $sys 64 1 | grep "is alive" > /dev/null
echo " "
echo

endif
endif
martyp $
```



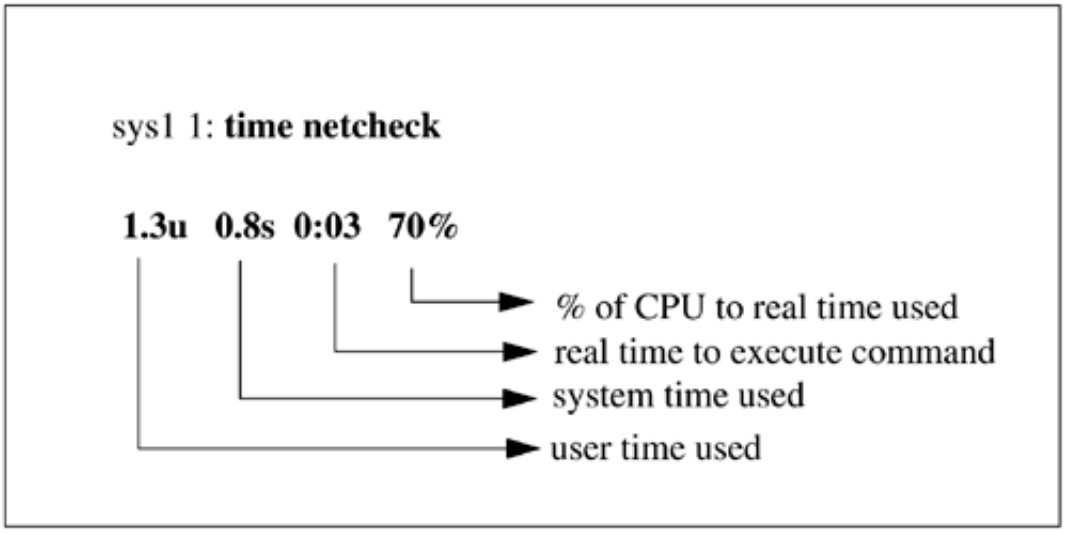
You can follow what is taking place on a line-by-line basis. The line beginning with 25 is the line in the file **csh_hostck** that has **grep** in it, that is, the output you would have received if the program had been run without the *-xv* options.

I would recommend performing the syntax check (*-n*) with a new shell program, and then echo all commands with the *-x* option only if you get unexpected results when you run the program. The debugging options will surely help you at some point when you run into problems with the shell programs you craft.

How Long Does It Take?

You can use the **time** command to see a report of the amount of time your shell program takes to run. The output of **time** is different for many UNIX variants. You may want to view the manual page for **time** to see what output you can expect. A typical output of **time** when in **csh** is shown in Figure 28-5:

Figure 28-5. time Example (different among UNIX variants)



Because some of the scripts you write may consume a substantial amount of system resources, you may want to consider investigating some of the job-control capabilities of the C shell. The simplest job control you can use is to run scripts in the background so that the priority of the script is low. By issuing the script name followed by the **&** (ampersand), you will run the script in the background. If you run several scripts in the background, you can get the status of these by issuing the **jobs** command. This is a more advanced C shell topic, but depending on the level of complexity of scripts you write, you may want to look into job control.

[Team LiB]

PREVIOUS NEXT

Manual Pages for Commands Used in Chapter 28

There are manual pages in Appendix A for the following commands used in this chapter:

awk
find
grep
nestat
ps

Appendix A. Select HP-UX Online Manual Pages

Manual Pages for Commands

awk

boot

crontab

find

grep

hpux

ifconfig

ignite

inittab

iostat

isl

netstat

pdc

ps

sed

shutdown

"sw" command summaries

top

vmstat

Manual Pages for Commands

The following section contains copies of the manual pages for commands used throughout the book. This makes a quick reference for you to use when issuing the commands commonly used during your system administration day. The manual pages, more commonly referred to as man pages, are listed in full detail. The man pages for Virtual Partitions (vPars) commands are in Appendix B.

awk



awk - Pattern-processing language.

```
awk(1)                                                          awk(1)

NAME
     awk - pattern-directed scanning and processing language

SYNOPSIS

     awk [-Ffs] [-v var=value] [program | -f progfile ...] [file ...]

DESCRIPTION
     awk scans each input file for lines that match any of a set of
     patterns specified literally in program or in one or more files
     specified as -f progfile.  With each pattern there can be an
     associated action that is to be performed when a line in a file
     matches the pattern.  Each line is matched against the pattern portion
     of every pattern-action statement, and the associated action is
     performed for each matched pattern.  The file name - means the
     standard input.  Any file of the form var=value is treated as an
     assignment, not a filename.  An assignment is evaluated at the time it
     would have been opened if it were a filename, unless the -v option is
     used.

     An input line is made up of fields separated by white space, or by
     regular expression FS.  The fields are denoted $1, $2, ...; $0 refers
     to the entire line.

   Options
     awk recognizes the following options and arguments:

          -F fs          Specify regular expression used to separate
                         fields.  The default is to recognize space and tab
                         characters, and to discard leading spaces and
                         tabs.  If the -F option is used, leading input
                         field separators are no longer discarded.

          -f progfile    Specify an awk program file.  Up to 100 program
                         files can be specified.  The pattern-action
                         statements in these files are executed in the same
                         order as the files were specified.

          -v var=value   Cause var=value assignment to occur before the
                         BEGIN action (if it exists) is executed.

   Statements
     A pattern-action statement has the form:

          pattern { action }

     A missing { action } means print the line; a missing pattern always
     matches.  Pattern-action statements are separated by new-lines or
     semicolons.

     An action is a sequence of statements.  A statement can be one of the
     following:

          if(expression) statement [else statement]
          while(expression) statement
          for(expression;expression;expression) statement
          for(var in array) statement
```

```
      do statement while(expression)
      break
      continue
      {[statement ...]}
      expression                  # commonly var=expression
      print[expression-list] [> expression]
      printf format [, expression-list] [> expression]
      return [expression]
      next           # skip remaining patterns on this input line.
      delete array [expression]          # delete an array element.
      exit [expression]      # exit immediately; status is expression.
```

Statements are terminated by semicolons, newlines or right braces. An empty expression-list stands for $0. String constants are quoted (""), with the usual C escapes recognized within. Expressions take on string or numeric values as appropriate, and are built using the operators +, -, *, /, %, ^ (exponentiation), and concatenation (indicated by a blank). The operators ++, --, +=, -=, *=, /=, %=, ^=, **=, >, >=, <, <=, ==, !=, and ?: are also available in expressions. Variables can be scalars, array elements (denoted x[i]) or fields. Variables are initialized to the null string. Array subscripts can be any string, not necessarily numeric (this allows for a form of associative memory). Multiple subscripts such as [i,j,k] are permitted. The constituents are concatenated, separated by the value of SUBSEP.

The print statement prints its arguments on the standard output (or on a file if >file or >>file is present or on a pipe if |cmd is present), separated by the current output field separator, and terminated by the output record separator. file and cmd can be literal names or parenthesized expressions. Identical string values in different statements denote the same open file. The printf statement formats its expression list according to the format (see printf(3)).

Built-In Functions

The built-in function close(expr) closes the file or pipe expr opened by a print or printf statement or a call to getline with the same string-valued expr. This function returns zero if successful, otherwise, it returns non-zero.

The customary functions exp, log, sqrt, sin, cos, atan2 are built in. Other built-in functions are:

blength[([s])]	Length of its associated argument (in bytes) taken as a string, or of $0 if no argument.
length[([s])]	Length of its associated argument (in characters) taken as a string, or of $0 if no argument.
rand()	Returns a random number between zero and one.
srand([expr])	Sets the seed value for rand, and returns the previous seed value. If no argument is given, the time of day is used as the seed value; otherwise, expr is used.
int(x)	Truncates to an integer value
substr(s, m[, n])	Return the at most n-character substring of s that begins at position m, numbering from 1. If n is omitted, the substring is limited by the length of string s.
index(s, t)	Return the position, in characters, numbering from 1, in string s where string t first occurs, or zero if it does not occur at all.
match(s, ere)	Return the position, in characters, numbering from 1, in string s where the extended regular expression ere occurs, or 0 if it does not. The variables RSTART and RLENGTH are set to the position and length of the matched string.
split(s, a[, fs])	Splits the string s into array elements a[1], a[2], ..., a[n], and returns n. The separation

is done with the regular expression fs, or with the field separator FS if fs is not given.

sub(ere, repl [, in])
Substitutes repl for the first occurrence of the extended regular expression ere in the string in. If in is not given, $0 is used.

gsub
Same as sub except that all occurrences of the regular expression are replaced; sub and gsub return the number of replacements.

sprintf(fmt, expr, ...)
String resulting from formatting expr ... according to the printf(3S) format fmt

system(cmd)
Executes cmd and returns its exit status

toupper(s)
Converts the argument string s to uppercase and returns the result.

tolower(s)
Converts the argument string s to lowercase and returns the result.

The built-in function getline sets $0 to the next input record from the current input file; getline < file sets $0 to the next record from file. getline x sets variable x instead. Finally, cmd | getline pipes the output of cmd into getline; each call of getline returns the next line of output from cmd. In all cases, getline returns 1 for a successful input, 0 for end of file, and -1 for an error.

Patterns

Patterns are arbitrary Boolean combinations (with ! || &&) of regular expressions and relational expressions. awk supports Extended Regular Expressions as described in regexp(5). Isolated regular expressions in a pattern apply to the entire line. Regular expressions can also occur in relational expressions, using the operators ~ and !~. /re/ is a constant regular expression; any string (constant or variable) can be used as a regular expression, except in the position of an isolated regular expression in a pattern.

A pattern can consist of two patterns separated by a comma; in this case, the action is performed for all lines from an occurrence of the first pattern though an occurrence of the second.

A relational expression is one of the following:

```
expression matchop regular-expression
expression relop expression
expression in array-name
(expr,expr,...) in array-name
```

where a relop is any of the six relational operators in C, and a matchop is either ~ (matches) or !~ (does not match). A conditional is an arithmetic expression, a relational expression, or a Boolean combination of the two.

The special patterns BEGIN and END can be used to capture control before the first input line is read and after the last. BEGIN and END do not combine with other patterns.

Special Characters

The following special escape sequences are recognized by awk in both regular expressions and strings:

Escape	Meaning
\a	alert character
\b	backspace character
\f	form-feed character
\n	new-line character
\r	carriage-return character
\t	tab character
\v	vertical-tab character
\nnn	1- to 3-digit octal value nnn
\xhhh	1- to n-digit hexadecimal number

Variable Names

Variable names with special meanings are:

FS	Input field separator regular expression; a space character by default; also settable by option -Ffs.
NF	The number of fields in the current record.
NR	The ordinal number of the current record from the start of input. Inside a BEGIN action the value is zero. Inside an END action the value is the number of the last record processed.
FNR	The ordinal number of the current record in the current file. Inside a BEGIN action the value is zero. Inside an END action the value is the number of the last record processed in the last file processed.
FILENAME	A pathname of the current input file.
RS	The input record separator; a newline character by default.
OFS	The print statement output field separator; a space character by default.
ORS	The print statement output record separator; a newline character by default.
OFMT	Output format for numbers (default %.6g). If the value of OFMT is not a floating-point format specification, the results are unspecified.
CONVFMT	Internal conversion format for numbers (default %.6g). If the value of CONVFMT is not a floating-point format specification, the results are unspecified.
SUBSEP	The subscript separator string for multi-dimensional arrays; the default value is " 34"
ARGC	The number of elements in the ARGV array.
ARGV	An array of command line arguments, excluding options and the program argument numbered from zero to ARGC-1. The arguments in ARGV can be modified or added to; ARGC can be altered. As each input file ends, awk will treat the next non-null element of ARGV, up to the current value of ARGC-1, inclusive, as the name of the next input file. Thus, setting an element of ARGV to null means that it will not be treated as an input file. The name - indicates the standard input. If an argument matches the format of an assignment operand, this argument will be treated as an assignment rather than a file argument.
ENVIRON	Array of environment variables; subscripts are names. For example, if environment variable V=thing, ENVIRON["V"] produces thing.
RSTART	The starting position of the string matched by the match function, numbering from 1. This is always equivalent to the return value of the match function.
RLENGTH	The length of the string matched by the match function.

Functions can be defined (at the position of a pattern-action statement) as follows:

```
function foo(a, b, c) { ...; return x }
```

Parameters are passed by value if scalar, and by reference if array name. Functions can be called recursively. Parameters are local to the function; all other variables are global.

Note that if pattern-action statements are used in an HP-UX command line as an argument to the awk command, the pattern-action statement must be enclosed in single quotes to protect it from the shell. For example, to print lines longer than 72 characters, the pattern-action statement as used in a script (-f progfile command form) is:

```
length > 72
```

The same pattern action statement used as an argument to the awk command is quoted in this manner:

```
awk 'length > 72'
```

EXTERNAL INFLUENCES

Environment Variables

LANG Provides a default value for the internationalization variables that are unset or null. If LANG is unset or null, the default value of "C" (see lang(5)) is used. If any of the internationalization variables contains an invalid setting, awk will behave as if all internationalization variables are set to "C". See environ(5).

LC_ALL If set to a non-empty string value, overrides the values of all the other internationalization variables.

LC_CTYPE Determines the interpretation of text as single and/or multi-byte characters, the classification of characters as printable, and the characters matched by character class expressions in regular expressions.

LC_NUMERIC Determines the radix character used when interpreting numeric input, performing conversion between numeric and string values and formatting numeric output. Regardless of locale, the period character (the decimal-point character of the POSIX locale) is the decimal-point character recognized in processing awk programs (including assignments in command-line arguments).

LC_COLLATE Determines the locale for the behavior of ranges, equivalence classes and multi-character collating elements within regular expressions.

LC_MESSAGES Determines the locale that should be used to affect the format and contents of diagnostic messages written to standard error and informative messages written to standard output.

NLSPATH Determines the location of message catalogues for the processing of LC_MESSAGES.

PATH Determines the search path when looking for commands executed by system(cmd), or input and output pipes.

In addition, all environment variables will be visible via the awk variable ENVIRON.

International Code Set Support

Single- and multi-byte character code sets are supported except that variable names must contain only ASCII characters and regular expressions must contain only valid characters.

DIAGNOSTICS

```
     awk supports up to 199 fields ($1, $2, ..., $199) per record.

EXAMPLES
     Print lines longer than 72 characters:

          length > 72

     Print first two fields in opposite order:

          { print $2, $1 }

     Same, with input fields separated by comma and/or blanks and tabs:

          BEGIN { FS = ",[ \t]*|[ \t]+" }
                { print $2, $1 }

     Add up first column, print sum and average:

                   { s += $1 }"
          END      { print "sum is", s, " average is", s/NR }

     Print all lines between start/stop pairs:

          /start/, /stop/

     Simulate echo command (see echo(1)):

          BEGIN    {                              # Simulate echo(1)
                   for (i = 1; i < ARGC; i++) printf "%s ", ARGV[i]
                   printf "\n"
                   exit }

AUTHOR
     awk was developed by AT&T, IBM, OSF, and HP.

SEE ALSO
     lex(1), sed(1).
     A. V. Aho, B. W. Kernighan, P. J. Weinberger: The AWK Programming
     Language, Addison-Wesley, 1988.

STANDARDS CONFORMANCE
     awk: SVID2, SVID3, XPG2, XPG3, XPG4, POSIX.2
```

[Team LiB]

PREVIOUS NEXT

boot



boot - Bootstrap process overview.

```
boot(1M)                    Series 700/800 Only                    boot(1M)

NAME
     boot - bootstrap process

DESCRIPTION

     The Series 700 and 800 bootstrap process involves the execution of
     three software components:

          -  pdc (see pdc(1M),

          -  isl (see isl(1M), and

          -  hpux (see hpux_800(1M)).

     After the processor is RESET, pdc, the processor-dependent code
     (firmware), performs a self-test and initializes the processor.  It
     then loads and transfers control to isl, the operating-system-
     independent initial system loader.  isl, in turn, loads and transfers
     control to the hpux utility, the HP-UX-specific bootstrap loader.
     hpux then downloads the HP-UX kernel object file from an HP-UX file
     system and transfers control to the loaded kernel image.

SEE ALSO
     hpux(1M), hpux_800(1M), isl(1M), pdc(1M).
```

crontab



crontab - Schedule jobs that are executed automatically on a regular basis.

```
crontab(1)                                                    crontab(1)

NAME
     crontab - user job file scheduler

SYNOPSIS

     crontab [file]

     crontab -e

     crontab -l

     crontab -r

DESCRIPTION
     The crontab command manages a crontab file for the user.  You can use
     a crontab file to schedule jobs that are executed automatically by
     cron (see cron(1M)) on a regular basis.  The command has four forms:

          crontab [file]    Create or replace your crontab file by copying
                            the specified file, or standard input if file
                            is omitted or - is specified as file , into the
                            crontab directory, /var/spool/cron/crontabs.
                            The name of your crontab file in the crontab
                            directory is the same as your effective user
                            name.

          crontab -e        Edit a copy of your crontab file, or create an
                            empty file to edit if the crontab file does not
                            exist. When editing is complete, the file will
                            be copied into the crontab directory as your
                            crontab file.

          crontab -l        List your crontab file.

          crontab -r        Remove your crontab file from the crontab
                            directory.

     The entries in a crontab file are lines of six fields each.  The
     fields are separated by spaces or tabs.  The lines have the following
     format:

          minute  hour  monthday  month  weekday  command

     The first five are integer patterns that specify when the sixth field,
     command, should be executed.  They can have the following ranges of
     values:

          minute         The minute of the hour, 0-59

          hour           The hour of the day, 0-23

          monthday       The day of the month, 1-31

          month          The month of the year, 1-12

          weekday        The day of the week, 0-6, 0=Sunday
```

Each pattern can be either an asterisk (*), meaning all legal values, or a list of elements separated by commas. An element is either a number in the ranges shown above, or two numbers in the range separated by a hyphen (meaning an inclusive range). Note that the specification of days can be made in two fields: monthday and weekday. If both are specified in an entry, they are cumulative. For example,

```
0   0   1,15   *   1   command
```

runs command at midnight on the first and fifteenth of each month, as well as every Monday. To specify days in only one field, set the other field to asterisk (*). For example,

```
0   0   *   *   1   command
```

runs command only on Mondays.

The sixth field, command (the balance of a line including blanks in a crontab file), is a string that is executed by the shell at the specified times. A percent character (%) in this field (unless escaped by a backslash (\)) is translated to a newline character, dividing the field into "lines". Only the first "line" (up to a % or end-of-line) of the command field is executed by the shell. Any other "lines" are made available to the command as standard input.

Blank lines and those whose first non-blank character is # will be ignored.

cron invokes the command from the user's HOME directory with the POSIX shell, (/usr/bin/sh). It runs in the c queue (see queuedefs(4)).

cron supplies a default environment for every shell, defining:

```
HOME=user's-home-directory
LOGNAME=user's-login-id
PATH=/usr/bin:/usr/sbin:.
SHELL=/usr/bin/sh
```

Users who desire to have their .profile executed must explicitly do so in the crontab entry or in a script called by the entry.

You can execute crontab if your name appears in the file /var/adm/cron/cron.allow. If that file does not exist, you can use crontab if your name does not appear in the file /var/adm/cron/cron.deny. If only cron.deny exists and is empty, all users can use crontab. If neither file exists, only the root user can use crontab. The allow/deny files consist of one user name per line.

EXTERNAL INFLUENCES

Environment Variables

LC_CTYPE determines the interpretation of text within file as single- and/or multi-byte characters.

LC_MESSAGES determines the language in which messages are displayed.

If LC_CTYPE or LC_MESSAGES is not specified in the environment or is set to the empty string, the value of LANG is used as a default for each unspecified or empty variable. If LANG is not specified or is set to the empty string, a default of "C" (see lang(5)) is used instead of LANG.

If any internationalization variable contains an invalid setting, crontab behaves as if all internationalization variables are set to "C". See environ(5). EDITOR determines the editor to be invoked when -e option is specified. The default editor is vi.

International Code Set Support

Single-byte and multi-byte character code sets are supported.

WARNINGS

Be sure to redirect the standard output and standard error from commands. If this is not done, any generated standard output or standard error is mailed to the user.

FILES

```
     /var/adm/cron                Main cron directory
     /var/adm/cron/cron.allow     List of allowed users
     /var/adm/cron/cron.deny      List of denied users
     /var/adm/cron/log            Accounting information
     /var/spool/cron/crontabs     Directory containing the crontab files

SEE ALSO
     sh(1), cron(1M), queuedefs(4).

STANDARDS CONFORMANCE
     crontab: SVID2, SVID3, XPG2, XPG3, XPG4
```

[Team LiB]

PREVIOUS NEXT

find



find - Find the files on system by recursively searching through the hierarchy.

```
find(1)                                                          find(1)

NAME
     find - find files

SYNOPSIS

     find pathname_list [expression]

DESCRIPTION
     The find command recursively descends the directory hierarchy for each
     path name in pathname_list (that is, one or more path names) seeking
     files that match a Boolean expression written in the primaries given
     below.  By default, find does not follow symbolic links.

     The Boolean expression is evaluated using short-circuit evaluation.
     This means that whenever the result of a Boolean operation (AND or OR)
     is known from evaluating the left-hand argument, the right-hand
     argument is not evaluated.

     In the descriptions of the primaries, the argument n represents a
     decimal integer; +n means more than n, -n means less than n, and n
     means exactly n.

     The following primaries are recognized:

     -depth                  A position-independent term which causes
                             descent of the directory hierarchy to be done
                             so that all entries in a directory are acted
                             on before the directory itself.  This can be
                             useful when find is used with cpio(1) to
                             transfer files that are contained in
                             directories without write permission.  It is
                             also useful when using cpio(1) and the
                             modification dates of directories must be
                             preserved.  Always true.

     -follow                 A position-independent term which causes find
                             to follow symbolic links.  Always true.

     -fsonly FStype          A position-independent term which causes find
                             to stop descending any directory whose file
                             system is not of the type specified by
                             FStype, where FStype is one of cdfs, hfs, or
                             nfs, representing the CDFS, HFS, or NFS file
                             system type, respectively.

                             In this context, mount points inherit the
                             FStype of their parent directory.  This means
                             that when -fsonly hfs has been specified and
                             find encounters an NFS mount point that is
                             mounted on an HFS file system, the mount
                             point will be visited but entries below that
                             mount point will not.  It is important to
                             note that when -fsonly nfs has been
                             specified, any HFS file systems that are
                             beneath the mount point of an NFS file system
                             are not traversed.  Always true.
```

-xdev
: A position-independent term that causes find to avoid crossing any file system mount points that exist below starting points enumerated in pathname_list. The mount point itself is visited, but entries below the mount point are not. Always true.

-mountstop
: Identical to -xdev. This primary is provided for backward compatibility only. -xdev is preferred over -mountstop.

-name file
: True if file matches the last component of the current file name. The matching is performed according to Pattern Matching Notation (see regexp(5)).

-path file
: Same as -name except the full path (as would be output by -print) is used instead of just the base name. Note that / characters are not treated as a special case. For example, */.profile matches ./home/fred/.profile.

-perm [-]mode
: In this primary, the argument mode is used to represent file mode bits. The argument is identical in format to the mode operand as described in chmod(1), with the exception that the first character must not be the - operator. When using the symbolic form of mode, the starting template is assumed to have all file mode bits cleared.

  If the leading minus is omitted, this primary is true when the file permission bits exactly match the value of mode. Bits associated with the symbolic attributes s (set-user-ID, set-group-ID) and t (sticky bit) are ignored when the minus is omitted.

  If mode is preceded by a minus, this primary is true if all of the bits that are set in mode are also set in the file permission bits. In this case, the bits associated with the symbolic attributes s and t are significant.

-fstype FStype
: True if the file system to which the file belongs is of type FStype, where FStype is one of cdfs, hfs, or nfs, corresponding to the CDFS, HFS, or NFS file system type, respectively.

-type c
: True if the type of the file is c, where c is one of:

f	Regular file
d	Directory
b	Block special file
c	Character special file
p	FIFO (named pipe)
l	Symbolic link
s	Socket
n	Network special file
M	Mount point

-links n
: True if the file has n links.

-user uname
: True if the file belongs to the user uname. If uname is numeric and does not appear as a login name in the /etc/passwd file, it is taken as a user ID. The uname operand can be preceded by a + or - to modify the comparison operation as described previously.

-group gname
: True if the file belongs to the group gname.

If gname is numeric and does not appear in the /etc/group file, it is taken as a group ID. The gname operand can be preceded by a + or - to modify the comparison operation as described previously.

-nouser True if the file belongs to a user ID that is not listed in the password database. See passwd(4).

-nogroup True if the file belongs to a group ID that is not listed in the group database. See group(4).

-size n[c] True if the file is n blocks long. If n is followed by a c, the size is in bytes.

-atime n True if the file has been accessed in n days. The access time of directories in pathname_list is changed by find itself.

-mtime n True if the file has been modified in n days.

-ctime n True if the file inode has been changed in n days.

-newer file True if the current file has been modified more recently than the argument file.

-newer[tv1[tv2]] file True if the indicated time value (tv1) of the current file is newer than the indicated time value (tv2) of file. The time values tv1 and tv2 are each selected from the set of characters:

- a The time the file was last accessed
- c The time the inode of the file was last modified
- m The time the file was last modified

If the tv2 character is omitted, it defaults to m. Note that the -newer option is equivalent to -newermm.

Syntax examples;

```
-newera file
-newermc file
```

-inum n True if the file serial number (inode number) is n. Note that file serial numbers are unique only within a given file system. Therefore, matching file serial numbers does not guarantee that the referenced files are the same unless you restrict the search to a single file system.

-linkedto path True if the file is the same physical file as the file specified by path (i.e., linked to path). This primary is similar to -inum, but correctly detects when a file is hard-linked to path, even when multiple file systems are searched.

-print Causes the current path name to be printed. Always true.

-exec cmd True if the executed cmd returns a zero value as exit status. The end of cmd must be punctuated by a semicolon (semicolon is special to the shell and must be escaped). Any command argument {} is replaced by the

current path name.

-ok cmd — Same as -exec except that the generated command line is printed with a question mark first, and is executed only if the user responds by typing y.

-cpio device — Write the current file on device in cpio(4) format (5120-byte records). The use of -cpio implies -depth. Always true.

-ncpio — Same as -cpio but adds the -c option to cpio. The use of -ncpio implies -depth. Always true.

-prune — If the current entry is a directory, cause find to skip that directory. This can be useful to avoid walking certain directories, or to avoid recursive loops when using cpio -p. Note, however, that -prune is useless if the -depth option has also been given. See the description of -only and the EXAMPLES section, below, for more information. Always true.

-only — This is a positive-logic version of -prune. A -prune is performed after every directory, unless -only is successfully evaluated for that directory. As an example, the following three commands are equivalent:

```
find . -fsonly hfs -print
find . -print -fstype hfs -only
find . -print ! -fstype hfs -prune
```

Note, however, that -only is useless if the -depth option has also been given. Always true.

(expression) — True if the parenthesized expression is true. The spaces are required. Parentheses are special to the shell and must be escaped, as in \(and \).

Primaries can be combined by using the following operators (in order of decreasing precedence):

! expression — Logical NOT operator. True if expression is not true.

expression [-a] expression — Logical AND operator. True if both of the expressions are true.

expression -o expression — Logical OR operator. True if either or both of the expressions are true.

If expression is omitted, or if none of -print, -ok, -exec, -cpio, or -ncpio is specified, -print is assumed.

Access Control Lists

The -acl primary enables the user to search for access control list entries. It is true if the file's access control list matches an access control list pattern or contains optional access control list entries (see acl(5)). It has three forms:

-acl aclpatt — Match all files whose access control list includes all (zero or more) pattern entries specified by the aclpatt pattern.

-acl =aclpatt — Match a file only if its access control list includes all (zero or more) pattern entries specified by the aclpatt pattern, and every entry in its access control list is matched by at least one pattern entry specified in the aclpatt pattern.

-acl opt Match all files containing optional access control list entries.

The aclpatt string can be given as an operator or short form pattern; see acl(5).

By default, -acl is true for files whose access control lists include all the (zero or more) access control list patterns in aclpatt. A file's access control list can also contain unmatched entries.

If aclpatt begins with =, the remainder of the string must match all entries in a file's access control list.

The aclpatt string (by default, or the part following =) can be either an access control list or an access control list pattern. However, if it is an access control list, aclpatt must include at least the three base entries ((user.%, mode), (%.group, mode), and (%.%, mode)).

As a special case, if aclpatt is the word opt, the primary is true for files with access control list entries.

EXTERNAL INFLUENCES

Environment Variables

If an internationalization variable is not specified or is null, it defaults to the value of LANG.

If LANG is not specified or is null, it defaults to C (see lang(5)).

If LC_ALL is set to a nonempty string value, it overrides the values of all the other internationalization variables.

If any internationalization variable contains an invalid setting, all internationalization variables default to C (see environ(5)).

LC_CTYPE determines the interpretation of text as single and/or multibyte characters, the classification of characters as printable, and the characters matched by character class expressions in regular expressions.

LC_MESSAGES determines the locale that should be used to affect the format and contents of diagnostic messages written to standard error and informative messages written to standard output.

NLSPATH determines the location of message catalogues for the processing of LC_MESSAGES.

International Code Set Support

Single- and multibyte character code sets are supported.

EXAMPLES

Search the two directories /example and /new/example for files containing the string Where are you and print the names of the files:

```
find /example /new/example -exec grep -l 'Where are you' {} \;
```

Remove all files named a.out or *.o that have not been accessed for a week:

```
find / \( -name a.out -o -name '*.o' \) -atime +7 -exec rm {} \;
```

Note that the spaces delimiting the escaped parentheses are required.

Print the names of all files on this machine. Avoid walking nfs directories while still printing the nfs mount points:

```
find / -fsonly hfs -print
```

Copy the entire file system to a disk mounted on /Disk, avoiding the recursive copy problem. Both commands are equivalent (note the use of -path instead of -name):

```
cd /; find . ! -path ./Disk -only -print | cpio -pdxm /Disk
```

```
          cd /; find . -path ./Disk -prune -o -print | cpio -pdxm /Disk

     Copy the root disk to a disk mounted on /Disk, skipping all mounted
     file systems below /.  Note that -xdev does not cause / to be skipped,
     even though it is a mount point.  This is because / is the starting
     point and -xdev only affects entries below starting points.

          cd /;  find . -xdev -print | cpio -pdm /Disk

     Change permissions on all regular files in a directory subtree to mode
     444, and permissions on all directories to 555:

          find <pathname> -type f -print | xargs chmod 444
          find <pathname> -type d -print | xargs chmod 555

          Note that output from find was piped to xargs(1) instead of using
          the -exec primary.  This is because when a large number of files
          or directories is to be processed by a single command, the -exec
          primary spawns a separate process for each file or directory,
          whereas xargs collects file names or directory names into
          multiple arguments to a single chmod command, resulting in fewer
          processes and greater system efficiency.

   Access Control List Examples
     Find all files not owned by user karl that have access control lists
     with at least one entry associated with karl, and one entry for no
     specific user in group bin with the read bit on and the write bit off:

          find  /  ! -user karl -acl 'karl.*, %.bin+r-w' -print

     Find all files that have a read bit set in any access control list
     entry:

          find  /  -acl '*.*+r' -print

     Find all files that have the write bit unset and execute bit set in
     every access control list entry:

          find  /  -acl '=*.*-w+x' -print

     Find all files that have optional access control list entries:

          find  /  -acl opt -print

DEPENDENCIES
   NFS
     The -acl primary is always false for NFS files.

WARNINGS
     Because of interoperability goals, cpio does not support archiving
     files larger than 2GB or files that have user/group IDs larger than
     60,000 (60K).  Files with user/group IDs greater than 60K are archived
     and restored under the user/group ID of the current process.

AUTHOR
     find was developed by AT&T and HP.

FILES
     /etc/group               Group names
     /etc/mnttab              Mount points
     /etc/passwd              User names

SEE ALSO
     chacl(1), chmod(1), cpio(1), sh(1), test(1), xargs(1), mknod(2),
     stat(2), cpio(4), fs(4), group(4), passwd(4), acl(5), environ(5),
     lang(5), regexp(5).

STANDARDS CONFORMANCE
     find: SVID2, SVID3, XPG2, XPG3, XPG4, POSIX.2
```

[Team LiB]

PREVIOUS NEXT

grep



grep - Command to match a specified pattern.

```
grep(1)                                                          grep(1)

NAME
     grep, egrep, fgrep - search a file for a pattern

SYNOPSIS

   Plain call with pattern
     grep [-E|-F] [-c|-l|-q] [-insvx] pattern [file ...]

   Call with (multiple) -e pattern
     grep [-E|-F] [-c|-l|-q] [-binsvx] -e pattern...  [-e pattern] ...
          [file ...]

   Call with -f file
     grep [-E|-F] [-c|-l|-q] [-insvx] [-f pattern_file] [file ...]

   Obsolescent:
     egrep [-cefilnsv] [expression] [file ...]

     fgrep [-cefilnsvx] [strings] [file ...]

DESCRIPTION
     The grep command searches the input text files (standard input
     default) for lines matching a pattern.  Normally, each line found is
     copied to the standard output.  grep supports the Basic Regular
     Expression syntax (see regexp(5)).  The -E option (egrep) supports
     Extended Regular Expression (ERE) syntax (see regexp(5)).  The -F
     option (fgrep) searches for fixed strings using the fast Boyer-Moore
     string searching algorithm.  The -E and -F options treat newlines
     embedded in the pattern as alternation characters.  A null expression
     or string matches every line.

     The forms egrep and fgrep are maintained for backward compatibility.
     The use of the -E and -F options is recommended for portability.

   Options
          -E                  Extended regular expressions.  Each pattern
                              specified is a sequence of one or more EREs.
                              The EREs can be separated by newline
                              characters or given in separate -e expression
                              options.  A pattern matches an input line if
                              any ERE in the sequence matches the contents
                              of the input line without its trailing
                              newline character.  The same functionality is
                              obtained by using egrep.

          -F                  Fixed strings.  Each pattern specified is a
                              sequence of one or more strings.  Strings can
                              be separated by newline characters or given
                              in separate -e expression options.  A pattern
                              matches an input line if the line contains
                              any of the strings in the sequence.  The same
                              functionality is obtained by using fgrep.

          -b                  Each line is preceded by the block number on
                              which it was found.  This is useful in
                              locating disk block numbers by context.
                              Block numbers are calculated by dividing by
```

512 the number of bytes that have been read from the file and rounding down the result.

`-c` Only a count of matching lines is printed.

`-e expression` Same as a simple expression argument, but useful when the expression begins with a hyphen (-). Multiple -e options can be used to specify multiple patterns; an input line is selected if it matches any of the specified patterns.

`-f pattern_file` The regular expression (grep and grep -E) or strings list (grep -F) is taken from the pattern_file.

`-i` Ignore uppercase/lowercase distinctions during comparisons.

`-l` Only the names of files with matching lines are listed (once), separated by newlines. If standard input is searched, a path name of - is listed.

`-n` Each line is preceded by its relative line number in the file starting at 1. The line number is reset for each file searched. This option is ignored if -c, -b, -l, or -q is specified.

`-q` (Quiet) Do not write anything to the standard output, regardless of matching lines. Exit with zero status upon finding the first matching line. Overrides any options that would produce output.

`-s` Error messages produced for nonexistent or unreadable files are suppressed.

`-v` All lines but those matching are printed.

`-x` (eXact) Matches are recognized only when the entire input line matches the fixed string or regular expression.

In all cases in which output is generated, the file name is output if there is more than one input file. Care should be taken when using the characters $, *, [, ^, |, (,), and \ in expression, because they are also meaningful to the shell. It is safest to enclose the entire expression argument in single quotes ('...').

EXTERNAL INFLUENCES

Environment Variables

LANG determines the locale to use for the locale categories when both LC_ALL and the corresponding environment variable (beginning with LC_) do not specify a locale. If LANG is not specified or is set to the empty string, a default of C (see lang(5)) is used.

LC_ALL determines the locale to use to override any values for locale categories specified by the settings of LANG or any environment variables beginning with LC_.

LC_COLLATE determines the collating sequence used in evaluating regular expressions.

LC_CTYPE determines the interpretation of text as single byte and/or multi-byte characters, the classification of characters as letters, the case information for the -i option, and the characters matched by character class expressions in regular expressions.

LC_MESSAGES determines the language in which messages are displayed.

If any internationalization variable contains an invalid setting, the commands behave as if all internationalization variables are set to C.

See environ(5).

International Code Set Support
Single-byte and multi-byte character code sets are supported.

RETURN VALUE

Upon completion, grep returns one of the following values:

0 One or more matches found.
1 No match found.
2 Syntax error or inaccessible file (even if matches were found).

EXAMPLES

In the Bourne shell (sh(1)) the following example searches two files, finding all lines containing occurrences of any of four strings:

```
grep -F 'if
then
else
fi' file1 file2
```

Note that the single quotes are necessary to tell grep -F when the strings have ended and the file names have begun.

For the C shell (see csh(1)) the following command can be used:

```
grep -F 'if\ then\ else\ fi' file1 file2
```

To search a file named address containing the following entries:

```
Ken   112 Warring St.  Apt. A
Judy  387 Bowditch  Apt. 12
Ann   429 Sixth St.
```

the command:

```
grep Judy address
```

prints:

```
Judy  387 Bowditch  Apt. 12
```

To search a file for lines that contain either a Dec or Nov, use either of the following commands:

```
grep -E '[Dd]ec|[Nn]ov' file

egrep -i 'dec|nov' file
```

Search all files in the current directory for the string xyz:

```
grep xyz *
```

Search all files in the current directory subtree for the string xyz, and ensure that no error occurs due to file name expansion exceeding system argument list limits:

```
find . -type f -print |xargs grep xyz
```

The previous example does not print the name of files where string xyz appears. To force grep to print file names, add a second argument to the grep command portion of the command line:

```
find . -type f -print |xargs grep xyz /dev/null
```

In this form, the first file name is that produced by find, and the second file name is the null file.

WARNINGS

(XPG4 only.) If the -q option is specified, the exit status will be zero if an input line is selected, even if an error was detected. Otherwise, default actions will be performed.

SEE ALSO

```
     sed(1), sh(1), regcomp(3C), environ(5), lang(5), regexp(5).

STANDARDS CONFORMANCE
     grep: SVID2, SVID3, XPG2, XPG3, XPG4, POSIX.2

     egrep: SVID2, SVID3, XPG2, XPG3, XPG4, POSIX.2

     fgrep: SVID2, SVID3, XPG2, XPG3, XPG4, POSIX.2
```

[Team LiB]

hpux



hpux - Boot HP-UX operating system.

```
hpux(1M)                                                        hpux(1M)

NAME
     hpux - HP-UX bootstrap

SYNOPSIS

     hpux [-F] [-lm] [-a[C|R|S|D] devicefile] [-fnumber] [-istring] [boot]
     [devicefile]
     hpux ll [devicefile] (same as hpux ls -aFln)
     hpux ls [-aFiln] [devicefile]
     hpux set autofile devicefile string
     hpux show autofile [devicefile]
     hpux -v
     hpux restore devicefile (Series 700 only; see DEPENDENCIES.)

DESCRIPTION
     hpux is the HP-UX specific secondary system loader (SSL) utility for
     bootstrap (see isl(1M) for the initial system loader).  It supports
     the operations summarized below, as shown in the SYNOPSIS and detailed
     later in this DESCRIPTION.

          boot                 Loads an object file from an HP-UX file
                               system or raw device and transfers control
                               to the loaded image.  (Note, the boot
                               operation is position dependent).

          ll                   Lists the contents of HP-UX directories in
                               a format similar to ls -aFln.  (See ls(1);
                               ls only works on a local disk with a HFS
                               file system).

          ls                   Lists the contents of HP-UX directories.
                               (See ls(1); ls only works on a local disk
                               with a HFS file system).

          show autofile        Displays the contents of the autoexecute
                               file.

          set autofile         Changes the contents of the autoexecute
                               file to that specified by string.

          -v                   Displays the release and version numbers of
                               the hpux utility.

          restore              Recovers the system from a properly
                               formatted bootable tape.  (Series 700
                               specific; see DEPENDENCIES.)

     hpux commands can be given interactively from the keyboard, or
     provided in an isl autoexecute file.

     hpux is limited to operations on the interface initialized by pdc(1M).
     In most cases, operations are limited to the boot device interface.

   Notation
     hpux accepts numbers (numeric constants) in many of its options.
     Numbers follow the C language notation for decimal, octal, and
     hexadecimal constants.  A leading 0 (zero) implies octal and a leading
```

0x or 0X implies hexadecimal. For example, 037, 0x1F, 0X1f, and 31 all represent the same number, decimal 31.

hpux boot, ll, ls, set autofile, show autofile, and restore operations accept devicefile specifications, which have the following format:

```
manager(w/x.y.z;n)filename
```

The devicefiles specification is comprised of a device name and a file name. The device name (manager(w/x.y.z;n)), consists of a generic name of an I/O system manager (device or interface driver) such as disc, a hardware path to the device, and minor number. The manager name can be omitted entirely if the default is used. w/x.y.z is the physical hardware path to the device, identifying bus converters, slot numbers, and hardware addresses. For Series 700 machines, there are a set of mnemonics that can be used instead of the hardware paths. The n is the minor number that controls manager-dependent functionality. The file name part, filename, is a standard HP-UX path name. Some hpux operations have defaults for particular components. A devicefile specification containing a device part only specifies a raw device. A devicefile specification containing a file name implies that the device contains an HP-UX file system, and that the filename resides in that file system.

A typical boot devicefile specification is

```
disc(2/4.0.0;0)/stand/vmunix
```

The manager is disc, the hardware path to the disk device is 2/4.0.0, the minor number shown as 0 by default, and the /stand/vmunix is the filename for the boot device.

hpux now supports a consolidated list of managers: disc, tape, and lan. The manager disc manages all CS/80 disks connected via HP-IB (formerly disc0); CS/80 disks connected via the HP27111 interface (formerly disc2); CS/80 disks connected via NIO HP-IB (formerly disc1); all disks connected via SCSI, (formerly disc3), and all autochanger disk devices (formerly disc30). The manager lan manages remote boot through the HP28652A NIO based LAN interface (formerly lan1). Remote boot is currently supported on this card only and not on any CIO-based LAN card. The manager tape manages the HP7974, HP7978, and HP7980 tape drives via HP-IB (formerly tape1) and tape drives via SCSI (formerly tape2).

The hardware path in a devicefile specification is a string of numbers, each suffixed by slash, (/), followed by a string of numbers separated by dots (.), each number identifying a hardware component notated sequentially from the bus address to the device address. A hardware component suffixed by a slash indicates a bus converter and may not be necessary on your machine. For example, in w/x.y.z w is the address of the bus converter, x is the address of the MID-BUS module, y is the CIO slot number, and z is the HP-IB address or HP27111 bus address.

The minor number, n, in a devicefile specification controls driver-dependent functionality. (See the manual, Configuring HP-UX for Peripherals, for minor-number bit assignments of specific drivers).

File names are standard HP-UX path names. No preceding slash (/) is necessary and specifying one will not cause problems.

Defaults

Default values chosen by hpux to complete a command are obtained through a sequence of steps. First, any components of the command specified explicitly are used. If the command is not complete, hpux attempts to construct defaults from information maintained by pdc (see pdc(1M)). If sufficient information to complete the command is unavailable, the autoexecute file is searched. If the search fails, any remaining unresolved components of the command are satisfied by hard-coded defaults.

There is no hard-coded default choice for a manager; if none can be chosen, hpux reports an error.

When the hardware path to the boot device is not specified, hpux

defaults to information maintained by pdc. The hardware path element has no hard-coded default.

If the minor number element is not supplied, hpux takes its default from the autoexecute file. Failing that, the hard-coded default of 0 is used.

For the boot command, a devicefile specification without a file name indicates that the boot device does not contain an HP-UX file system. hpux interprets this as a NULL (instead of missing) file name and does not search for a default. If the entire devicefile specification is missing, hpux searches for a default; either the autoexecute file contents or the hard-coded default is chosen.

There are two possible hard-coded default devicefile specifications. One hard-coded default devicefile specification is /vmunix. The other hard-coded default devicefile specification is /stand/vmunix.

If you have a LVM system where the boot volume and the root volume are on different logical volumes, the kernel would be /vmunix. This is because the boot volume will be mounted under /stand when the system is up.

For all other configurations, the kernel would be /stand/vmunix.

The search order for the hard-coded defaults is /stand/vmunix and then /vmunix.

boot Operation

The boot operation loads an object file from an HP-UX file system or raw device as specified by the optional devicefile. It then transfers control to the loaded image.

Any missing components in a specified devicefile are supplied with a default. For example, a devicefile of vmunix.new would actually yield:

```
disc(8.0.0;0)vmunix.new
```

and a devicefile of (8.0.1)/stand/vmunix, for booting from the disk at HP-IB address 1, would yield

```
disc(8.0.1;0)/stand/vmunix
```

Regardless of how incomplete the specified devicefile may be, boot announces the complete devicefile specification used to find the object file. Along with this information, boot gives the sizes of the TEXT, DATA, and BSS, segments and the entry offset of the loaded image, before transferring control to it.

The boot operation accepts several options. Note that boot options must be specified positionally as shown in the syntax statement in the SYNOPSIS. Options for the boot operations are as follows:

-a[C|R|S|D] devicefile	Accept a new location (as specified by devicefile) and pass it to the loaded image. If that image is an HP-UX kernel, the kernel will erase its predefined I/O configuration, and configure in the specified devicefile. If the C, R, S, or D option is specified, the kernel configures the devicefile as the console, root, swap, or dump device, respectively. Note that -a can be repeated multiple times.
-fnumber	Use the number and pass it as the flags word to the loaded image.
-istring	Set the initial run-level for init (see init(1M)) when booting the system. The run-level specified will override any run-level specified in an initdefault entry in /etc/inittab (see inittab(4)).

-lm
Boot the system in LVM maintenance mode, configure only the root volume, and then initiate single user mode.

-F
Use with SwitchOver/UX software. Ignore any locks on the boot disk. The -F option should be used only when it is known that the processor holding the lock is no longer running. (If this option is not specified and a disk is locked by another processor, the kernel will not boot from it, to avoid the corruption that would result if the other processor were still using the disk).

boot places some restrictions on object files it can load. It accepts only the HP-UX magic numbers EXECMAGIC (0407), SHAREMAGIC (0410), and DEMANDMAGIC (0413). See magic(4). The object file must contain an Auxiliary Header of the HPUX_AUX_ID type and it must be the first Auxiliary Header (see a.out(4)).

ll and ls Operations
The ll and ls operations list the contents of the HP-UX directory specified by the optional devicefile. The output is similar to that of ls -aFl command, except the date information is not printed.

The default devicefile is generated just as for boot, defaulting to the current directory.

set autofile Operation
The set autofile operation overwrites the contents of the autoexecute file, autofile, with the string specified (see autoexecute in the EXAMPLES section).

show autofile Operation
The show autofile operation displays the contents of the autoexecute file, autofile (see autoexecute in the EXAMPLES section).

DIAGNOSTICS

If an error is encountered, hpux prints diagnostic messages to indicate the cause of the error. These messages fall into the General, Boot, Copy, Configuration, and System Call categories. System Call error messages are described in errno(2). The remaining messages are listed below.

General

bad minor number in devicefile spec
The minor number in the devicefile specification is not recognized.

bad path in devicefile spec
The hardware path in the devicefile specification is not recognized.

command too complex for parsing
The command line contains too many arguments.

no path in devicefile spec
The devicefile specification requires (but does not contain) a hardware path component.

panic (in hpuxboot): (display==number, flags==number) string
A severe internal hpux error has occurred. Report to your nearest HP Field Representative.

Boot

bad magic
The specified object file does not have a recognizable magic number.

bad number in flags spec
The flags specification in the -f option is not recognized.

Exec failed: Cannot find /stand/vmunix or /vmunix.
Neither /stand/vmunix or /vmunix could be found.

booting from raw character device
In booting from a raw device, the manager specified only has a character interface, which might cause problems if the block size is incorrect.

isl not present, please hit system RESET button to continue
An unsuccessful boot operation has overlaid isl in memory. It is impossible to return control to isl.

short read
The specified object file is internally inconsistent; it is not long enough.

would overlay
Loading the specified object file would overlay hpux.

Configuration

cannot add path, error number
An unknown error has occurred in adding the hardware path to the I/O tree. The internal error number is given. Contact your HP Field Representative.

driver does not exist
The manager specified is not configured into hpux.

driver is not a logical device manager
The manager named is not that of a logical device manager and cannot be used for direct I/O operations.

error rewinding device"
An error was encountered attempting to rewind a device.

error skipping file
An error was encountered attempting to forward-space a tape device.

negative skip count
The skip count, if specified, must be greater than or equal to zero.

no major number
The specified manager has no entry in the block or character device switch tables.

path incompatible with another path
Multiple incompatible hardware paths have been specified.

path long
The hardware path specified contains too many components for the specified manager.

path short
The hardware path specified contains too few components for the specified manager.

table full
Too many devices have been specified to hpux.

EXAMPLES

As a preface to the examples which follow, here is a brief overview of HP-UX system boot-up sequences.

Automatic Boot

Automatic boot processes on various HP-UX systems follow similar general sequences. When power is applied to the HP-UX system processor, or the system Reset button is pressed, processor-dependent code (firmware) is executed to verify hardware and general system integrity (see pdc(1M)). After checking the hardware, pdc gives the user the option to override the autoboot sequence by pressing the Esc key. At that point, a message resembling the following usually appears on the console.

```
      (c) Copyright. Hewlett-Packard Company. 1994.
      All rights reserved.

      PDC ROM rev. 130.0
      32 MB of memory configured and tested.

      Selecting a system to boot.
      To stop selection process, press and hold the ESCAPE key...
```

If no keyboard activity is detected, pdc commences the autoboot sequence by loading isl (see isl(1M)) and transferring control to it. Since an autoboot sequence is occurring, isl finds and executes the autoexecute file which, on an HP-UX system, requests that hpux be run with appropriate arguments. Messages similar to the following are displayed by isl on the console:

```
      Booting from: scsi.6  HP 2213A
      Hard booted.
      ISL Revision A.00.09  March 27, 1990
      ISL booting  hpux boot disk(;0)/stand/vmunix
```

hpux, the secondary system loader, then announces the operation it is performing, in this case boot, the devicefile from which the load image comes, and the TEXT size, DATA size, BSS size, and start address of the load image, as shown below, before control is passed to the image.

```
      Booting disk(scsi.6;0)/stand/vmunix
      966616+397312+409688 start 0x6c50
```

The loaded image then displays numerous configuration and status messages.

Interactive Boot

To use hpux interactively, isl must be brought up in interactive mode by pressing the Esc key during the interval allowed by pdc. pdc then searches for and displays all bootable devices and presents a set of boot options. If the appropriate option is chosen, pdc loads isl and isl interactively prompts for commands. Information similar to the following is displayed:

```
      Selection process stopped.

      Searching for Potential Boot Devices.
      To terminate search, press and hold the ESCAPE key.

      Device Selection    Device Path             Device Type
      ------------------------------------------------------------
      P0                  scsi.6.0                QUANTUM PD210S
      P1                  scsi.1.0                HP      2213A
      p2                  lan.ffffff-ffffff.f.f   hpfoobar

      b)  Boot from specified device
      s)  Search for bootable devices
      a)  Enter Boot Administration mode
      x)  Exit and continue boot sequence

      Select from menu: b p0 isl

      Trying scsi.6.0
      Boot path initialized.
      Attempting to load IPL.

      Hard booted.
      ISL Revision A.00.2G  Mar  27, 1994
      ISL>
```

Although all of the operations and options of hpux can be used from isl interactively, they can also be executed from an autoexecute file. In the examples below, user input is the remainder of the line after each ISL> prompt shown. The remainder of each example is text displayed by the system. Before going over specific examples of the various options and operations of hpux, here is an outline of the steps taken in the automatic boot process. Although the hardware

configuration and boot paths shown are for a single Series 800 machine, the user interfaces are consistent across all models. When the system Reset button is depressed, pdc executes self-test, and assuming the hardware tests pass, pdc announces itself, sends a BELL character to the controlling terminal, and gives the user 10 seconds to override the autoboot sequence by entering any character. Text resembling the following is displayed on the console:

```
     Processor Dependent Code (PDC) revision 1.2
     Duplex Console IO Dependent Code (IODC) revision 3

     Console path        = 56.0.0.0.0.0.0   (dec)
                           38.0.0.0.0.0.0   (hex)

     Primary boot path   = 44.3.0.0.0.0.0   (dec)
                           2c.00000003.0.0.0.0.0   (hex)

     Alternate boot path = 52.0.0.0.0.0.0   (dec)
                           34.0.0.0.0.0.0   (hex)

     32 MB of memory configured and tested.

     Autosearch for boot path enabled

     To override, press any key within 10 seconds.
```

If no keyboard character is pressed within 10 seconds, pdc commences the autoboot sequence by loading isl and transferring control to it. Because an autoboot sequence is occurring, isl merely announces itself, finds and executes the autoexecute file which, on an HP-UX system, requests that hpux be run with appropriate arguments. The following is displayed on the console.

```
     10 seconds expired.
     Proceeding with autoboot.

     Trying Primary Boot Path
     ------------------------
     Booting...
     Boot IO Dependent Code (IODC) revision 2

     HARD Booted.

     ISL Revision A.00.2G Mar  20, 1994

     ISL booting  hpux
```

hpux then announces the operation it is performing, in this case boot, the devicefile from which the load image comes, and the TEXT size, DATA size, BSS size, and start address of the load image. The following is displayed before control is passed to the image.

```
     Boot
     : disc3(44.3.0;0)/stand/vmunix
     3288076 + 323584 + 405312 start 0x11f3e8
```

Finally, the loaded image displays numerous configuration and status messages, then proceeds to init run-level 2 for multiuser mode of operation.

isl must be brought up in interactive mode to use the operations and options of hpux. To do this, simply enter a character during the 10 second interval allowed by pdc. pdc then asks if the primary boot path is acceptable. Answering yes (Y) is usually appropriate. pdc then loads isl and isl interactively prompts for commands. The following lines show the boot prompt, the Y response, subsequent boot messages, and finally the Initial System Loader (ISL) prompt that are sent to the display terminal:

```
     Boot from primary boot path (Y or N)?> y
     Interact with IPL (Y or N)?> y

     Booting...
     Boot IO Dependent Code (IODC) revision 2
```

```
       HARD Booted.

       ISL Revision A.00.2G Mar  20, 1994

       ISL>
```

Although all of the operations and options of hpux can be used from isl interactively, they can also be executed from an autoexecute file. In the examples below, all user input follows the ISL> prompt on the same line. Subsequent text is resultant messages from the ISL.

Default Boot

Entering hpux initiates the default boot sequence. The boot path read from pdc is 8.0.0, the manager associated with the device at that path is disc, the minor number, in this case derived from the autoexecute file, is 4 specifying section 4 of the disk, and the object file name is /stand/vmunix.

```
       ISL> hpux

       Boot
       : disc3(44.3.0;0)/stand/vmunix
       3288076 + 323584 + 405312 start 0x11f3e8
```

Booting Another Kernel

In this example, hpux initiates a boot operation where the name of the object file is vmunix.new.

```
       ISL> hpux vmunix.new

       Boot
       : disc3(44.3.0;0)/stand/vmunix.new
       3288076 + 323584 + 405312 start 0x11f3e8
```

Booting From Another Section

In this example (shown for backward compatibility), a kernel is booted from another section of the root disk. For example, suppose kernel development takes place under /mnt/azure/root.port which happens to reside in its own section, section 3 of the root disk. By specifying a minor number of 3 in the above example, the object file sys.azure/S800/vmunix is loaded from /mnt/azure/root.port.

```
       ISL> hpux (;3)sys.azure/S800/vmunix

       Boot
       : disc(8.0.0;0x3)sys.azure/S800/vmunix
       966616+397312+409688 start 0x6c50
```

Booting From Another Disk

Only the hardware path and file name are specified in this example. All other values are boot defaults. The object file comes from the file system on another disk.

```
       ISL> hpux (52.5.0.0)/stand/vmunix

       Boot
       : disc(52.5.0.0)/stand/vmunix
       966616+397312+409688 start 0x6c50
```

Booting From LAN

This example shows how to boot a cluster client from the LAN. Though this example specifies a devicefile, you can also use default boot, as shown in a previous example. For a boot operation other than default boot, the file name must be specified and can be no longer than 11 characters. Booting to isl from a local disk then requesting an image to be loaded from the LAN is not supported.

```
       ISL> hpux lan(32)/stand/vmunix

       Boot
       : lan(32;0x0)/stand/vmunix
       966616+397312+409688 start 0x6c50
```

Booting To Single User Mode

In this example, the -i option is used to make the system come up in

run-level s, for single user mode of operation.

```
ISL> hpux -is

Boot
: disc(8.0.0;0x0)/stand/vmunix
966616+397312+409688 start 0x6c50

   Kernel Startup Messages Omitted

INIT: Overriding default level with level 's'

INIT: SINGLE USER MODE
WARNING:  YOU ARE SUPERUSER !!
#
```

Booting With A Modified I/O Configuration

Here, a tape driver is configured in at CIO slot 2, HP-IB address 0. Regardless of what was present in the kernel's original I/O configuration, the driver tape is now configured at that hardware path. Similarly, mux0 is configured in at CIO slot 1 which is to be the console. The only other devices configured are the console and root device, which boot derived from pdc.

```
ISL> hpux -aC mux0(8.1) -a tape(8.2.0)

Boot
: disc(8.0.0;0x0)/stand/vmunix
: Adding mux0(8.1;0x0)...
: Adding tape(8.2.0;0x0)...
966616+397312+409688 start 0x6c50
Beginning I/O System Configuration.
cio_ca0 address = 8
   hpib0 address = 0
      disc0 lu = 0 address = 0
   mux0 lu = 0 address = 1
   hpib0 address = 2
      tape1 lu = 0 address = 0
I/O System Configuration complete.

   Additional Kernel Startup Messages Omitted
```

Booting From A Raw Device

This example shows booting from a raw device (that is, a device containing no file system). Note that no file name is specified in the devicefile. The device is an HP7974 tape drive, and therefore tape is the manager used. The tape drive is at CIO slot 2, HP-IB address 3. The first file on the tape will be skipped. The minor number specifies a tape density of 1600 BPI with no rewind on close. Depending on the minor number, tape requires the tape be written with 512 or 1024 byte blocks.

```
ISL> hpux tape(8.2.3;0xa0000)

Boot
: tape(8.2.3;0xa0000)
966616+397312+409688 start 0x6c50
```

Displaying The Autoexecute File

In this example, show autofile is used to print the contents of the autoexecute file residing in the boot LIF, on the device from which hpux was booted. Optionally, a devicefile can be specified in order to read the autoexecute file from the boot LIF of another boot device.

```
  ISL> hpux show autofile
  Show autofile
  : AUTO file contains (hpux)
```

Changing The Autoexecute File

This example shows how to change the contents of the autoexecute file. Once done, the system can be reset, and the new command will be used during any unattended boot.

```
ISL> hpux set autofile "hpux /stand/vmunix.std"
Set autofile
```

```
        : disk(2/0/1.3.0.0.0.0.0;0)
        : AUTO file now contains "(hpux /stand/vmunix.std)"
```

Listing Directory Contents

The contents of the directory (/stand) on the root disk are listed. The format shows the file protections, number of links, user id, group id, and size in bytes for each file in the directory. There are three available kernels to boot: vmunix, vmunix.test, and vmunix.prev. Listing the files over the LAN is not supported.

```
        ISL> hpux ll /stand

        Ls
        : disk(2/0/1.3.0.0.0.0.0;0)/stand
        dr-xr-xr-x    3 2        2              1024 ./
        drwxr-xr-x   17 0        0              1024 ../
        -rw-r--r--    1 0        3               191 bootconf
        drwxr-xr-x    2 0        0              1024 build/
        -rw-r--r--    1 0        0               632 ioconfig
        -rw-r--r--    1 0        3                82 kernrel
        -r--r--r--    1 0        3               426 system
        -rw-r--r--    1 0        3               437 system.prev
        -rwxr-xr-x    1 0        3           7771408 vmunix*
        -rwxr-xr-x    1 0        3           7771408 vmunix.prev*
```

Getting The Version

The -v option is used to get the version numbers of hpux.

```
        ISL> hpux -v

        Release: 10.00
        Release Version:
        @(#) X10.20.B HP-UX() #1: Dec  4 1995 16:55:08
```

DEPENDENCIES

Series 700 Only

The restore operation is provided as a recovery mechanism in the event that a disk becomes totally corrupted. It copies data from a properly formatted bootable tape to disk. When this tape contains a backup image of the disk, the entire disk is restored. To create a properly formatted tape (DDS ONLY), the following commands should be executed:

```
        dd if=/usr/lib/uxbootlf of=/dev/rmt/0mn bs=2k
        dd if=/dev/rdsk/1ss of=/dev/rmt/0m bs=64k
```

The first dd puts a boot area on the tape, making it a bootable image (see dd(1)). Once the boot image is on tape, the tape is not rewound. The next dd appends an image of the disk to the tape. The entire process takes about one hour for a 660 MB HP2213 disk. To avoid later problems with fsck after the disk is restored, bring the system to single user mode and type sync a few times before doing the second dd (see fsck(1M)). Once created, the tape can be used to completely restore the disk:

1. Insert the tape into the tape drive.

2. Instruct the machine to boot to ISL from the tape. This is usually done by specifying scsi.3 as the boot path.

3. Enter the following in response to the ISL prompt:

```
        ISL> hpux restore disk(scsi.1;0)
```

This restores the disk image from the tape to the actual disk at scsi.1. Any existing data on the disk will be lost. This command destroys the contents of the device specified by devicefile. The restoration process takes about one hour for a 660 MB drive.

NOTE: There is a 2 GB limit on the amount of data that can be restored. The tape and disk must be on the boot device interface.

Also, this command may be replaced in the future by superior installation and recovery mechanisms. At that time, this command will be removed.

SEE ALSO
boot(1M), fsck(1M), init(1M), isl(1M), pdc(1M), errno(2), a.out(4),
inittab(4), magic(4).

[Team LiB]

PREVIOUS NEXT

ifconfig



ifconfig - Display or configure network interface parameters.

```
ifconfig(1M)                                                    ifconfig(1M)

NAME
     ifconfig - configure network interface parameters

SYNOPSIS

     ifconfig interface address_family [address [dest_address]] [parameters]

     ifconfig interface [address_family]

DESCRIPTION
     The first form of the ifconfig command assigns an address to a network
     interface and/or configures network interface parameters.  ifconfig
     must be used at boot time to define the network address of each
     interface present on a machine.  It can also be used at other times to
     redefine an interface's address or other operating parameters.

     The second form of the command, without address_family, displays the
     current configuration for interface.  If address_family is also
     specified, ifconfig reports only the details specific to that address
     family.

     Only a user with appropriate privileges can modify the configuration
     of a network interface.  All users can run the second form of the
     command.

   Arguments
     ifconfig recognizes the following arguments:

          address         Either a host name present in the host name
                          database (see hosts(4)), or a DARPA Internet
                          address expressed in Internet standard dot
                          notation (see inet(3N)).  The host number can be
                          omitted on 10MB/second Ethernet interfaces (which
                          use the hardware physical address), and on
                          interfaces other than the first.

          address_family  Name of protocol on which naming scheme is based.
                          An interface can receive transmissions in
                          differing protocols, each of which may require
                          separate naming schemes.  Therefore, it is
                          necessary to specify the address_family, which
                          may affect interpretation of the remaining
                          parameters on the command line.  The only address
                          family currently supported is inet (DARPA-
                          Internet family).

          dest_address    Address of destination system.  Consists of
                          either a host name present in the host name
                          database (see hosts(4)), or a DARPA Internet
                          address expressed in Internet standard dot
                          notation (see inet(3N)).

          interface       A string of the form nameunit, such as lan0.
                          (See the LAN Card Numbering subsection.)

          parameters      One or more of the following operating
                          parameters:
```

up	Mark an interface "up". Enables interface after an ifconfig down. Occurs automatically when setting the address on an interface. Setting this flag has no effect if the hardware is "down".
down	Mark an interface "down". When an interface is marked "down", the system will not attempt to transmit messages through that interface. If possible, the interface will be reset to disable reception as well. This action does not automatically disable routes using the interface.
broadcast	(Inet only) Specify the address that represents broadcasts to the network. The default broadcast address is the address with a host part of all 1's.
debug	Enable driver-dependent debugging code. This usually turns on extra console error logging.
-debug	Disable driver-dependent debugging code.
ipdst	(NS only) This is used to specify an Internet host that is willing to receive IP packets encapsulating NS packets bound for a remote network. In this case, an apparent point-to-point link is constructed, and the address specified is taken as the NS address and network of the destination.
metric n	Set the routing metric of the interface to n. The default is 0. The routing metric is used by the routing protocol (see gated(1m)). Higher metrics have the effect of making a route less favorable; metrics are counted as additional hops to the destination network or host.
netmask mask	(Inet only) Specify how much of the address to reserve for subdividing networks into sub-networks or aggregating networks into supernets. mask can be specified as a single hexadecimal number with a leading 0x, with a dot-notation Internet address, or with a pseudo-network name listed in the network table (see networks(4)). For subdividing networks into sub-networks, mask must include the network part of the local address, and the subnet part which is taken from the host field of the address. mask must contain 1's in the bit positions in the 32-bit address that are to be used for the network and subnet parts, and 0's in the host part. The 1's in the mask must be contiguous starting from the leftmost bit position in the 32-

bit field. mask must contain at least the standard network portion, and the subnet field must be contiguous with the network portion. The subnet field must contain at least 2 bits. The subnet part after performing a bit-wise AND operation between the address and the mask must not contain all 0's or all 1's. For aggregating networks into supernets, mask must only include a portion of the network part. mask must contain contiguous 1's in the bit positions starting from the leftmost bit of the 32-bit field.

trailers — Request the use of a "trailer" link-level encapsulation when sending. If a network interface supports trailers, the system will, when possible, encapsulate outgoing messages in a manner that minimizes the number of memory-to-memory copy operations performed by the receiver. On networks that support the Address Resolution Protocol, this flag indicates that the system should request that other systems use trailers when sending to this host. Similarly, trailer encapsulations will be sent to other hosts that have made such requests. Currently used by Internet protocols only. See WARNINGS section.

-trailers — Disable the use of a "trailer" link-level encapsulation (default).

LAN Card Numbering

The name of an interface associated with a LAN card is lan, and its unitnumber is determined as follows. The LAN card installed first in the system is given interface unit number 0; the next LAN card installed is given interface unit number 1; and so on. When there are two or more LAN cards installed at the same time, interface unit numbers are assigned according to card positions in the backplane: the LAN card that appears "first" in the backplane is given the interface unit number N; the next LAN card in the backplane is given the number N+1.

The lanscan command can be used to display the name and unit number of each interface that is associated with a LAN card (see lanscan(1M)).

Supernets

A supernet is a collection of smaller networks. Supernetting is a technique of using the netmask to aggregate a collection of smaller networks into a supernet. This technique is particularly useful for class C networks. A Class C network can only have 254 hosts. This can be too restrictive for some companies. For these companies, a netmask that only contains a portion of the network part can be applied to the hosts in these class C networks to form a supernet. This supernet netmask should be applied to those interfaces that connect to the supernet using the ifconfig command. For example, a host can configure its interface to connect to a class C supernet, 192.6, by configuring an IP address of 192.6.1.1 and a netmask of 255.255.0.0 to its interface.

DIAGNOSTICS

Messages indicate if the specified interface does not exist, the requested address is unknown, or the user is not privileged and tried to alter an interface's configuration.

```
WARNINGS
     Currently, all HP 9000 systems can receive trailer packets but do not
     send them.  Setting the trailers flag has no effect.

SEE ALSO
     netstat(1), lanconfig(1m), lanscan(1m) hosts(4), routing(7).
```

[Team LiB]

PREVIOUS NEXT

ignite

ignite - 

Invoke the Graphical User Interface (GUI) of Ignite-UX.

```
ignite(5)                                                    ignite(5)



NAME
     ignite - HP-UX configuration and installation manager

SYNOPSIS
     /opt/ignite/bin/ignite

DESCRIPTION
   Introduction
     ignite is part of the Ignite-UX product, a client-server application
     that provides the ability to configure and install HP-UX systems.
     Ignite-UX is available on all HP-UX 10.XX platforms (with the
     exception of HP-UX 10.00) and supports the installation of all 10.XX
     releases and 10.XX applications.

     The ignite command is the graphical user interface of this client-
     server application.  It provides the ability to build software
     configurations and to use these configurations to install HP-UX
     systems.

     Key features:

   + True client-server framework enabling an install session for
     multiple targets to be controlled from a single server.

   + Support of client standalone installation by enabling an
     installation to be controlled from a terminal user interface
     running on the target machine.

   + Support for non-interactive installations initiated from the
     client or server.

   + Support for multiple sources within a single install session.
     This allows end customers to install multiple applications at
     the same time they install the base OS.

   + Support for Software Distributor (SD) and non-SD sources, by
     providing the ability to load SD software, as well as software
     from a non-SD sources (tar, cpio, or pax).  One use of this
     feature is the ability to load a system with an OS and
     applications, capture that system into an archive and then use
     that archive to install to multiple targets.

   + Support for user-defined configurations by allowing a system
     administrator to construct a description (configuration) for a
     target system including disk and networking layout, software to
     install, kernel modifications and post-installation
     customizations.  Once this configuration has been defined it
     may then be applied to one or more target machines.

   + Support for user-defined customization both by defining what a
     target system should look like and by allowing execution of
     user-supplied scripts at pre-defined points in the installation
     process.
```

+ Support for saved configurations by enabling a user to modify
  an existing configuration, save these changes, and then quickly
  apply the new configuration to a target system.

+ Support for system manifest creation by providing a simple
  method to capture a snapshot of the currently loaded software
  along with a complete hardware inventory.

 Hardware/Software Requirements
   For doing 10.XX loads, the following hardware is required to set up
   either a Series 700 or Series 800 Ignite-UX server. If a Series 800
   Ignite-UX server is set up, it needs a graphics display or the display
   can be redirected to another X(1) windows system. The redirection is
   accomplished by setting the DISPLAY environment variable. For
   example, in the Korn Shell or Posix Shell you would type: export
   DISPLAY=hpfcdn:0.0

   Ignite-UX server requirements:

+ A Series 700/800 system running HP-UX 10.XX.

+ An X11 display server (workstation, X-terminal, PC running an
  X server, etc). Can be the same system as above.

+ Sufficient disk space to load Ignite-UX, and any software
  depots and/or archives to be used during the install.

+ Access to the Ignite-UX tool set. The tool set can be loaded
  onto any 10.XX system.

+ Tape/CD-ROM in order to load Ignite-UX and any software depots
  you plan to distribute onto the server.

+ Network access to any clients to be installed. Client and
  Server must be on the same subnet if you plan to do the initial
  boot of the client over the network.

 Server Setup: Overview
1. Install HP-UX 10.XX.

2. Install Ignite-UX tools and data.

3. Set up core software.

4. Add additional applications (optional).

5. Run ignite to complete the configuration and to start the
    process.

    Note: All operations are executed as "root" on the Ignite-UX
    server. Except where noted, all commands referenced here are
    located in /opt/ignite/bin.

 Server Setup: Details
    1. Install HP-UX 10.XX
 Refer to the manual for instructions on how to update a system to
 HP-UX 10.XX.

    2. Install Ignite-UX Software
 The Ignite-UX tool-set is contained on the HP-UX application set
 of CD-ROMs. The software bundles are named as such: Ignite-UX-
 10-01, Ignite-UX-10-10, Ignite-UX-10-20, etc. Each software
 bundle contains the Ignite-UX tools plus the data files required
 for support of the particular HP-UX release indicated by the
 bundle name. You may load one or more of the Ignite-UX-10-XX
 bundles onto your server depending on which releases of HP-UX you
 plan on installing onto clients.

 The Ignite-UX product replaces the capability previously supplied
 by the NetInstall bundle that came with HP-UX releases 10.01,
 10.10 and 10.20. Loading one of the Ignite-UX software bundles
 will cause the NetInstall bundle to be automatically removed.

 Once the application CD-ROM containing Ignite-UX has been

mounted, you may use the swinstall command to load the desired Ignite-UX bundles. For example, the command below would load the support needed for installing HP-UX 10.20 onto clients:

```
swinstall -s /cdrom Ignite-UX-10-20
```

3. Set up "core" HP-UX software

Before Ignite-UX can be used, you must configure the software for it to load onto the clients. Since both SD sources and archive (non-SD) sources (tar, cpio, or pax) may be used, both cases will be considered separately below.

For SD OS software:

For Ignite-UX to use an SD source, you must have a registered SD depot available, and an Ignite-UX configuration file generated from that depot.

If you already have an SD depot containing the core HP-UX software, then you can enable Ignite-UX to access it by using the make_config and manage_index commands. If you do not already have a depot available, you may want to use the add_release command to lead you though copying the software from the distribution media into a depot and then configuring Ignite-UX to use it. Both methods are outlined below.

Enabling (or updating) an existing depot

If you already have an SD depot available, or if you have made changes to a depot that Ignite-UX knows about, then you can use the make_config and manage_index commands to generate a configuration file that Ignite-UX will use in accessing the depot. For example:

```
make_config -s server:/depot_700 \
 -c /var/opt/ignite/data/Rel_B.10.20/core_700
manage_index -a -f /var/opt/ignite/data/Rel_B.10.20/core_700
```

If at a later time, you modify the contents of a depot (you use swcopy to add software, for example), then the same make_config step will need to be rerun in order for Ignite-UX to be aware of the modifications.

Note: The make_config command only operates on software that is contained in a bundle. If you have a depot that has products not in a bundle, then you can run the make_bundles command on the depot prior to running make_config.

Using add_release to do it all

If you need to copy the core HP-UX software off the distribution media into a depot, and configure Ignite-UX to use it, then the add_release command can be used to lead you though all the steps.

To run add_release to see what it would do and not actually modify anything, you can specify the -p option (preview mode).

For example:

```
/opt/ignite/bin/add_release  -s /dev/dsk/c0t2s0 -p
```

To use a depot other than /dev/rmt/0m to read the software, you can specify it with the -s option.

For example:

```
/opt/ignite/bin/add_release -s jupiter:/depot/s700_10.XX_final
```

For non-SD (archive) OS software:

Ignite-UX has the capability of loading a system from an archive image taken from a system that represents your standard configuration. This method gives significantly faster install times, but may not be as flexible as using an SD source.

You will first need to generate the archive image of a
system in the desired state. It is recommended that the
/opt/ignite/data/scripts/make_sys_image script be used to
accomplish this task.

Once an archive image is created, then a configuration file
that represents the location and attributes of that image
must be created before Ignite-UX can use it.

A sample of a config file that can be used with a core
archive can be found at:
/opt/ignite/data/examples/core.cfg

The comments in this example file describe where to copy the
file and what to change in the file to make it reference
your archive and to work in your environment.

      4. Add additional applications (optional)
   If you have other software that you would like to have loaded
   during the system installation then you can create configuration
   files similar to what was done for the core-OS software by using
   make_config and manage_index or by using an example configuration
   file and modifying it.

   For SD application software
Run the following commands for each depot you plan to load
SD software from during the installation. The make_config
command only handles SD software which is packaged in bundle
form. If the SD depot you want to use has software not
contained in a bundle (for example: a collection of
patches), then the make_bundles command may be used to
create bundles in the depot.

If the contents are not 700/800 specific then the
make_config -a[78]00 option should not be used.

For example, to make a depot containing patches available:
make_bundles /depots/s700_10.20_patches

make_config -s hpfcxxx.hp.com:/depots/s700_10.20_patches
  -a 700 -c /opt/ignite/data/Rel_B.10.20/patches_700_cfg

manage_index -a -f /opt/ignite/data/Rel_B.10.20/patches_700_cfg

For example, to make a depot containing compilers available
which are already in bundles (no need to use make_bundles):
make_config -s hpfcxxx.hp.com:/depots/compiler
  -c /opt/ignite/data/Rel_B.10.XX/compilers_cfg

manage_index -a -f /opt/ignite/data/Rel_B.10.XX/compilers_cfg

The depot server (in this example hpfcxxx) should be
replaced with the server you have the SD software on. The
make_bundles script must be run on the same system where the
depot exists. If the depot is not on the Ignite-UX server,
you may need to copy the make_bundles script to the depot
server and run it there.

Note: The make_config command will need to be re-run each
time new software is added or modified in the depots.

make_config constructs Ignite-UX config files which
correspond to SD depots. When an SD depot is used as part
of the Ignite-UX process, it must have a config file which
describes the contents of the depot to Ignite-UX. This
command can automatically construct such a config file given
the name of an SD depot to operate on. This command should
be run when adding or changing a depot which will be used by
Ignite-UX.

manage_index is used to manipulate the /var/opt/ignite/INDEX
file. This utility is primarily called by other Ignite-UX
tools but can also be called directly.

   For non-SD application software:

If the source is not an "SD" depot, the make_config command
is not applicable. You will need to create a unique config
file that includes the non-SD software. A sample of a
config file that does a non-core archive can be found at:

/opt/ignite/data/examples/noncore.cfg

The comments in this example file describe where to copy the
file and what to change in the file to make it reference
your archive and to work in your environment.

5. Run ignite to complete the configuration and to start the process
On the Series 700 server run: /opt/ignite/bin/ignite. This will start
the Ignite-UX server program.

Complete the Configuration:
When the GUI display titled "Ignite-UX" appears, do the following
in that window:

a) Choose Options: Server Configuration

b) Look over both the Server Options and the Session Options to
see if they are suitable.

About the Screen: 'Add Booting IP Addresses':

Booting Clients: xx.xx.xx.xx to xx.xx.xx.xx
These IP addresses are used to initially boot the target
systems. They are used until the system is assigned one of
the DHCP-assigned addresses. One address is required for
each simultaneous boot. Typically one to three are needed,
depending on the usage.

DHCP Addresses: xx.xx.xx.xx to xx.xx.xx.xx
These IP addresses are used during the OS download and
application loading. These addresses are in use for most of
the Ignite-UX download to a target machine.

One address is required for each simultaneous download. You
should set more, if the addresses are assigned permanently.

DHCP Class ID:
The unique name for the DHCP server that serves these DHCP
Addresses. Not necessarily the install server.

If you will not be using the install server as the DHCP
server, then either do not set class ID at all, or you need
to know what the client ID is set to on your real DHCP
server.

Do not apply the class ID unless you are configuring the
install server to be a DHCP server.

DHCP Addresses are temporary:
If these DHCP Addresses are only used for doing installs,
and the clients will get reassigned new addresses when
deployed. Keep this field set.

If you want to set up the Ignite-UX server as a departmental
DHCP server, in which case the IP address leases are
permanent, and isolated to the department's DHCP server, set
this field to false.

Boot the S700 or S800 client system that supports network boot:
If the system you plan to install is running HP-UX version 9.X or
10.X, then you can use the bootsys command to remotely reboot the
client to run Ignite-UX.

If the client system is new, or disabled such that bootsys cannot
be used, then you can boot it over the network (note, that
currently the only S800 systems that are capable of network boot
are the K and D class systems).

To do a network boot, Go to the console for that client and enter

the appropriate command. (You will find the exact boot ROM commands for your system in

In general:

1) Stop the auto-boot by pressing <escape>.

2) For older Series 700s (non-bootp), type the following:

search lan

It will typically require three searches for the server to be found and listed, on older Series 700s.

Boot the client either by using the index displayed in the search or by using the following:

boot lan.080009-XXXYYY

3) For newer Series 700s, type the following to search for servers on your network. If you already know the server's IP address, or you are on a S800 (K and D class) that do not support the search command, then just issue the boot command below:

search lan install

Note: Some C-class systems had a firmware defect such that the search lan install install does not display any servers. However, doing an explicit boot using the server's IP address works correctly. A firmware upgrade is available for systems with this defect.

Choose the preferred boot server (if you have more than one) and enter the following:

boot lan.IP-Address install

If only one server appears in the listing, just type the following:

boot lan install

If the client cannot find the server, check the following items:

+ Client is on the same subnet as the server.

+ Any instl_bootd errors in /var/adm/syslog/syslog.log.

+ Your /var/adm/inetd.sec file to make sure that IP address 0.0.0.0 is not being disallowed for the instl_boots service.

+ If /etc/services comes from NIS, make sure that the NIS server has instl_boot* entries.

+ rbootd is running.

Booting a system remotely using bootsys:
If a system you need to reinstall is up and running HP-UX 9.X or 10.X and available on the network, then you can use the bootsys command from the server to cause the client system to boot Ignite-UX. Using the bootsys command has the advantages of:

+ Can be done remotely from the Ignite-UX server. Does not require access to the client console.

+ It is able to boot S800's that are not capable of a network boot.

+ It allows booting clients that are on different subnets (since it is not really doing a true network boot).

+ The bootsys command may also be used to schedule installations
to happen at a later time by calling it from at or cron.

The bootsys command has options to initiate an automated
installation (-a), or an installation controlled from the ignite
user interface (-w).

 Booting a system from customer-created install
   An install image used to boot a system can be created using
   make_medialif. When transferred to physical media, it can be used to:

+ Automatically load an archive image also stored on the media
  without intervention.

+ Initiate an install from an Ignite-UX server. The install can
  be either automated or interactive.

   This is a way for shipping user-customized install media to remote
   sites. See make_medialif(1M) for more information.

  Start the Installation:
    After the client is booted, its icon should appear on the ignite
    interface. If the server has not been set up completely, or if the
    client could not obtain enough networking parameters via DHCP, then
    the client may require interaction on the client console.

    After the client icon appears on the server screen, select it by
    clicking on the icon for that client. Use the Actions menu to select a
    task for the selected client. The first task would be to choose
    "Install Client". Then choose "New Install".

    During the installation, choose a configuration file for this
    installation. Clients will be installed per the description given in
    the configuration file. If you want to reuse this configuration, save
    the file.

    Once installation is proceeding, check the client's status on the
    server.

    When the installation completes, you can print a manifest, and either
    save the client's data in a history directory or remove the client and
    its data from the server.

    Refer to the online help: "Ignite-UX Concepts", or "Getting Started"
    for more information.

  Standalone Installation
    The standalone type of installation is invoked by booting the system
    from the network, as described in the previous section, or by booting
    from the Ignite-UX media. After choosing "Install HP-UX", select
    "Local interaction at console, installing from network server" on the
    next screen.

    Or if you are installing from media, select "Media installation, with
    user interaction at local console".

    A basic interface, "wizard mode" can be chosen at this point. This
    will direct you though the required system setup steps. This mode is
    for the novice user, and proceeds though a more limited set of
    configuration steps, while giving you recommended choices based on
    your system's hardware.

    Additionally, a more sophisticated interface can be selected if the
    user needs to modify the filesystem configuration, or would like to
    set the system hostname and networking parameters prior to completing
    the installation.

    If you find that you have selected the wrong interface to accomplish
    your task, you may use the "Cancel" button which will allow you to
    switch to the other interface mode.

    The standalone installation uses an ASCII (TUI) interface, and
    requires a keyboard for navigation.

    The following keys can be used to navigate in the various screens:

+ <return> and <space> keys select an item

+ <tab> key moves to the next item on the screen.

+ <right> arrow and <down> arrow keys cycle to the next item in the "Tab Group"

A Tab Group is defined as all of the selectable options within an area of the screen defined by dotted lines. To move to the next Tab Group, the <tab> key must be used.

+ <left> arrow and <up> arrow keys cycle to the previous item in the Tab Group. To move to the next Tab Group, the <tab> key must be used.

+ Additionally, any key with an underlined character, can be chosen by typing that underlined character.

Manifest Generation

Included in the Ignite-UX tool set is a command: print_manifest(1M). This utility prints a formatted ascii system manifest to stdout. The manifest includes information on hardware and software installed and configured on the system. It gathers information about the system every time it is run.

ignite can display and/or print the manifest of a just-installed system with the action "View/Print Manifest". If the client's data is moved to history, that data includes both the client's manifest and config file. Both these files can be recalled at a later time.

EXTERNAL INPUTS AND INFLUENCES

Default Options:

The server maintains a defaults file, ignite.defs, located at /var/opt/ignite/server/ignite.defs. This file contains a subset of the values that are entered on the Server Configuration screen.

The following values and their defaults are shipped in the Ignite-UX product:

client_timeout:30
Time (in mins) until the client is declared hung.

halt_when_done:false
Halt the client after installation rather than reboot to invoke set_parms.

ignite_welcome:true
Show the server's welcome screen.

itool_welcome:true
Ask for customer information during client installation.

new_client_notification:true
Asks whether the user be notified when new clients boot.

Locking

In order to allow multiple ignite sessions to run concurrently with the currently installing process, ignite will lock a client during a New Install or a Repeat Install. This lock is tested in the actions: New Install, Repeat Installation, Stop Client and Remove Client. The lock is removed when the client is stopped or COMPLETE. For stopped clients, it is possible for someone, other than the installer, to remove them. For COMPLETE clients, it is possible for someone, other than the installer, to remove them or move them to history.

RETURN VALUES

ignite returns the following values:

0ignite completed successfully.

1ignite failed.

DIAGNOSTICS

Logging:

```
     All major events are logged to the server logfile located at
     /var/opt/ignite/logs/server.

FILES
     /opt/ignite/bin
  Contains Ignite-UX commands.

     /opt/ignite/lbin
  Contains Ignite-UX commands used by other commands.

     /var/opt/ignite/depots
  Contains the software depots used by Ignite-UX.

     /var/opt/ignite/logs
  Contains logfiles for each command.

     /var/opt/ignite/clients
  Contains the per-client directories.

     /var/opt/ignite/server
  Contains the file ignite.defs (server defaults).

AUTHOR
     Ignite-UX was developed by the Hewlett-Packard Company.

SEE ALSO
     add_release(1M), archive_impact(1M), bootsys(1M), instl_adm(1M),
     instl_adm(4), instl_bootd(1M), make_bundles(1M), make_config(1M),
     make_depots(1M), make_medialif(1M), manage_index(1M),
     print_manifest(1M), remove_release(1M), save_config(1M), sd(5),
     setup_server(1M).
```

[Team LiB]

PREVIOUS NEXT

inittab



inittab - File that supplies scripts to init.

```
inittab(4)                                                    inittab(4)

NAME
     inittab - script for the boot init process

DESCRIPTION

     The /etc/inittab file supplies the script to the boot init daemon in
     its role as a general process dispatcher (see init(1M)).  The process
     that constitutes the majority of boot init's process dispatching
     activities is the line process /usr/sbin/getty that initiates
     individual terminal lines.  Other processes typically dispatched by
     boot init are daemons and shells.

     The inittab file is composed of entries that are position-dependent
     and have the following format:

          id:rstate:action:process

     Each entry is delimited by a newline; however, a backslash (\)
     preceding a newline indicates a continuation of the entry.  Up to 1024
     characters per entry are permitted.  Comments can be inserted in the
     process field by starting a "word" with a # (see sh(1)).  Comments for
     lines that spawn gettys are displayed by the who command (see who(1)).
     It is expected that they will contain some information about the line
     such as the location.  There are no limits (other than maximum entry
     size) imposed on the number of entries within the inittab file.

     The entry fields are:

          id        A one- to four-character value used to uniquely
                    identify an entry.  Duplicate entries cause an error
                    message to be issued, but are otherwise ignored.  The
                    use of a four-character value to identify an entry is
                    strongly recommended (see WARNINGS below).

          rstate    Defines the run level in which this entry is to be
                    processed.  Run levels correspond to a configuration of
                    processes in the system where each process spawned by
                    boot init is assigned one or more run levels in which
                    it is allowed to exist.  Run levels are represented by
                    a number in the range 0 through 6.  For example, if the
                    system is in run level 1, only those entries having a 1
                    in their rstate field are processed.

                    When boot init is requested to change run levels, all
                    processes that do not have an entry in the rstate field
                    for the target run level are sent the warning signal
                    (SIGTERM) and allowed a 20-second grace period before
                    being forcibly terminated by a kill signal (SIGKILL).
                    You can specify multiple run levels for a process by
                    entering more than one run level value in any
                    combination.  If no run level is specified, the process
                    is assumed to be valid for all run levels, 0 through 6.

                    Three other values, a, b and c, can also appear in the
                    rstate field, even though they are not true run levels.
                    Entries having these characters in the rstate field are
                    processed only when a user init process requests them
```

to be run (regardless of the current system run level). They differ from run levels in that boot init can never enter "run level" a, b, or c. Also, a request for the execution of any of these processes does not change the current numeric run level.

Furthermore, a process started by an a, b, or c option is not killed when boot init changes levels. A process is killed only if its line in inittab is marked off in the action field, its line is deleted entirely from inittab, or boot init goes into the single-user state.

action A keyword in this field tells boot init how to treat the process specified in the process field. The following actions can be specified:

boot — Process the entry only at boot init's boot-time read of the inittab file. Boot init starts the process, does not wait for its termination, and when it dies, does not restart the process. In order for this instruction to be meaningful, the rstate should be the default or it must match boot init's run level at boot time. This action is useful for an initialization function following a hardware boot of the system.

bootwait — Process the entry only at boot init's boot-time read of the inittab file. Boot init starts the process, waits for its termination, and, when it dies, does not restart the process.

initdefault — An entry with this action is only scanned when boot init is initially invoked. Boot init uses this entry, if it exists, to determine which run level to enter initially. It does this by taking the highest run level specified in the rstate field and using that as its initial state. If the rstate field is empty, boot init enters run level 6.

The initdefault entry cannot specify that boot init start in the single-user state. Additionally, if boot init does not find an initdefault entry in inittab, it requests an initial run level from the user at boot time.

off — If the process associated with this entry is currently running, send the warning signal (SIGTERM) and wait 20 seconds before forcibly terminating the process via the kill signal (SIGKILL). If the process is nonexistent, ignore the entry.

once — When boot init enters a run level that matches the entry's rstate, start the process and do not wait for its termination. When it dies, do not restart the process. If boot init enters a new run level but the process is still running from a previous run level change, the process is not restarted.

ondemand — This instruction is really a synonym for the respawn action. It is functionally identical to respawn but

```
                                is given a different keyword in order
                                to divorce its association with run
                                levels.  This is used only with the
                                a, b, or c values described in the
                                rstate field.

                powerfail       Execute the process associated with
                                this entry only when boot init
                                receives a power-fail signal (SIGPWR
                                see signal(5)).

                powerwait       Execute the process associated with
                                this entry only when boot init
                                receives a power-fail signal (SIGPWR)
                                and wait until it terminates before
                                continuing any processing of inittab.

                respawn         If the process does not exist, start
                                the process; do not wait for its
                                termination (continue scanning the
                                inittab file).  When it dies, restart
                                the process.  If the process
                                currently exists, do nothing and
                                continue scanning the inittab file.

                sysinit         Entries of this type are executed
                                before boot init tries to access the
                                console.  It is expected that this
                                entry will be only used to initialize
                                devices on which boot init might
                                attempt to obtain run level
                                information.  These entries are
                                executed and waited for before
                                continuing.

                wait            When boot init enters the run level
                                that matches the entry's rstate,
                                start the process and wait for its
                                termination.  Any subsequent reads of
                                the inittab file while boot init is
                                in the same run level cause boot init
                                to ignore this entry.

        process   This is a sh command to be executed.  The entire
                  process field is prefixed with exec and passed to a
                  forked sh as "sh -c 'exec command'".  For this reason,
                  any sh syntax that can legally follow exec can appear
                  in the process field.  Comments can be inserted by
                  using the ; #comment syntax.

WARNINGS
     The use of a four-character id is strongly recommended.  Many pty
     servers use the last two characters of the pty name as an id.  If an
     id chosen by a pty server collides with one used in the inittab file,
     the /etc/utmp file can become corrupted.  A corrupt /etc/utmp file can
     cause commands such as who to report inaccurate information.

FILES
     /etc/inittab        File of processes dispatched by boot init.

SEE ALSO
     sh(1), getty(1M), exec(2), open(2), signal(5).
```

[Team LiB]

PREVIOUS NEXT

iostat



iostat - Interactively report I/O and CPU statistics.

```
iostat(1)                                                    iostat(1)

NAME
     iostat - report I/O statistics

SYNOPSIS

     iostat [-t] [interval [count]]

DESCRIPTION
     iostat iteratively reports I/O statistics for each active disk on the
     system.  Disk data is arranged in a four-column format:

          Column Heading           Interpretation
             device                Device name
             bps                   Kilobytes transferred per second
             sps                   Number of seeks per second
             msps                  Milliseconds per average seek

     If two or more disks are present, data is presented on successive
     lines for each disk.

     To compute this information, seeks, data transfer completions, and the
     number of words transferred are counted for each disk.  Also, the
     state of each disk is examined HZ times per second (as defined in
     <sys/param.h>) and a tally is made if the disk is active.  These
     numbers can be combined with the transfer rates of each device to
     determine average seek times for each device.

     With the advent of new disk technologies, such as data striping, where
     a single data transfer is spread across several disks, the number of
     milliseconds per average seek becomes impossible to compute
     accurately. At best it is only an approximation, varying greatly,
     based on several dynamic system conditions.  For this reason and to
     maintain backward compatibility, the milliseconds per average seek (
     msps ) field is set to the value 1.0.

   Options
     iostat recognizes the following options and command-line arguments:

          -t          Report terminal statistics as well as disk
                      statistics.  Terminal statistics include:

                           tin    Number of characters read from terminals.

                           tout   Number of characters written to
                                  terminals.
                           us     Percentage of time system has spent in
                                  user mode.
                           ni     Percentage of time system has spent in
                                  user mode running low-priority (nice)
                                  processes.
                           sy     Percentage of time system has spent in
                                  system mode.
                           id     Percentage of time system has spent
                                  idling.


          interval    Display successive lines which are summaries of the
```

```
                    last interval seconds.  The first line reported is
                    for the time since a reboot and each subsequent line
                    is for the last interval only.

          count     Repeat the statistics count times.

EXAMPLES
     Show current I/O statistics for all disks:

          iostat

     Display I/O statistics for all disks every 10 seconds until INTERRUPT
     or QUIT is pressed:

          iostat 10

     Display I/O statistics for all disks every 10 seconds and terminate
     after 5 successive readings:

          iostat 10 5

     Display I/O statistics for all disks every 10 seconds, also show
     terminal and processor statistics, and terminate after 5 successive
     readings:

          iostat -t 10 5

WARNINGS
     Users of iostat must not rely on the exact field widths and spacing of
     its output, as these will vary depending on the system, the release of
     HP-UX, and the data to be displayed.

AUTHOR
     iostat was developed by the University of California, Berkeley, and
     HP.

FILES
      /usr/include/sys/param.h

SEE ALSO
      vmstat(1).
```

[Team LiB]



isl



isl - Initial System Loader (isl) overview.

```
isl(1M)                        Series 800 Only                        isl(1M)

NAME
     isl - initial system loader

DESCRIPTION

     isl implements the operating system independent portion of the
     bootstrap process.  It is loaded and executed after self-test and
     initialization have completed successfully.

     The processor contains special purpose memory for maintaining critical
     configuration related parameters (e.g. Primary Boot, Alternate Boot,
     and Console Paths).  Two forms of memory are supported: Stable Storage
     and Non-Volatile Memory (NVM).

     Typically, when control is transferred to isl, an autoboot sequence
     takes place.  An autoboot sequence allows a complete bootstrap
     operation to occur with no intervention from an operator.  isl
     executes commands from the autoexecute file in a script-like fashion.
     autoboot is enabled by a flag in Stable Storage.

     autosearch is a mechanism that automatically locates the boot and
     console devices.  For further information, see pdc(1M).

     During an autoboot sequence, isl displays its revision and the name of
     any utility it executes.  However, if autoboot is disabled, after isl
     displays its revision, it then prompts for input from the console
     device.  Acceptable input is any isl command name or the name of any
     utility available on the system.  If a non-fatal error occurs or the
     executed utility returns, isl again prompts for input.

   Commands
     There are several commands available in isl. The following is a list
     with a short description.  Parameters may be entered on the command
     line following the command name.  They must be separated by spaces.
     isl prompts for any necessary parameters that are not entered on the
     command line.

          ?
          help           Help - List commands and available utilities

          listf
          ls             List available utilities

          autoboot       Enable or disable the autoboot sequence
                         Parameter - on or off

          autosearch     Enable or disable the autosearch sequence
                         Parameter - on or off

          primpath       Modify the Primary Boot Path
                         Parameter - Primary Boot Path in decimal

          altpath        Modify the Alternate Boot Path
                         Parameter - Alternate Boot Path in decimal

          conspath       Modify the Console Path
                         Parameter - Console Path in decimal
```

```
          lsautofl
          listautofl     List contents of the autoexecute file

          display        Display the Primary Boot, Alternate Boot, and
                         Console Paths

          readnvm        Display the contents of one word of NVM in
                         hexadecimal
                         Parameter - NVM address in decimal or standard
                         hexadecimal notation

          readss         Display the contents of one word of Stable Storage
                         in hexadecimal
                         Parameter - Stable Storage address in decimal or
                         standard hexadecimal notation

DIAGNOSTICS
     isl displays diagnostic information through error messages written on
     the console and display codes on the LED display.

     For the display codes, CE0x are informative only.  CE1x and CE2x
     indicate errors, some of which are fatal and cause the system to halt.
     Other errors merely cause isl to display a message.

     Non-fatal errors during an autoboot sequence cause the autoboot
     sequence to be aborted and isl to prompt for input.  After non-fatal
     errors during an interactive isl session, isl merely prompts for
     input.

     Fatal errors cause the system to halt.  The problem must be corrected
     and the system RESET to recover.


     CE00   isl is executing.
     CE01   isl is autobooting from the autoexecute file.
     CE02   Cannot find an autoexecute file.  autoboot aborted.
     CE03   No console found, isl can only autoboot.
     CE05   Directory of utilities is too big, isl reads only 2K bytes.
     CE06   autoexecute file is inconsistent.  autoboot aborted.
     CE07   Utility file header inconsistent: SOM values invalid.
      CE08   autoexecute file input string exceeds 2048 characters.  autoboot aborted.
     CE09   isl command or utility name exceeds 10 characters.
     CE0F   isl has transferred control to the utility.
     CE10   Internal inconsistency: Volume label - FATAL.
     CE11   Internal inconsistency: Directory - FATAL.
     CE12   Error reading autoexecute file.
     CE13   Error reading from console - FATAL.
     CE14   Error writing to console - FATAL.
     CE15   Not an isl command or utility.
     CE16   Utility file header inconsistent: Invalid System ID.
     CE17   Error reading utility file header.
     CE18   Utility file header inconsistent: Bad magic number.
     CE19   Utility would overlay isl in memory.
     CE1A   Utility requires more memory than is configured.
     CE1B   Error reading utility into memory.
     CE1C   Incorrect checksum: Reading utility into memory.
     CE1D   Console needed - FATAL.
     CE1E   Internal inconsistency: Boot device class - FATAL.
     CE21   Destination memory address of utility is invalid.
     CE22   Utility file header inconsistent: pdc_cache entry.
     CE23   Internal inconsistency: iodc_entry_init - FATAL.
     CE24   Internal inconsistency: iodc_entry_init - console - FATAL.
     CE25   Internal inconsistency: iodc_entry_init - boot device - FATAL.
     CE26   Utility file header inconsistent: Bad aux_id.
     CE27   Bad utility file type.

SEE ALSO
     boot(1M), hpux_800(1M), pdc(1M).
```

[Team LiB]

◀ PREVIOUS NEXT ▶

netstat



netstat - Display statistics related to networking.

```
netstat(1)                                                    netstat(1)

NAME
     netstat - show network status

SYNOPSIS

     netstat [-aAn] [-f address-family] [system [core]]
     netstat [-mMnrsv] [-f address-family] [-p protocol] [system [core]]
     netstat [-gin] [-I interface] [interval] [system [core]]

DESCRIPTION
     netstat displays statistics for network interfaces and protocols, as
     well as the contents of various network-related data structures.  The
     output format varies according to the options selected.  Some options
     are ignored when used in combination with other options.

     Generally, the netstat command takes one of the three forms shown
     above:

          -  The first form of the command displays a list of active
             sockets for each protocol.

          -  The second form displays the contents of one of the other
             network data structures according to the option selected.

          -  The third form displays configuration information for each
             network interface.  It also displays network traffic data on
             configured network interfaces, optionally updated at each
             interval, measured in seconds.

Options are interpreted as follows:

          -a                Show the state of all sockets, including
                            passive sockets used by server processes.  When
                            netstat is used without any options (except -A
                            and -n), only active sockets are shown.  This
                            option does not show the state of X.25
                            programmatic access sockets.  The option is
                            ignored if the -g, -i, -I, -m, -M, -p, -r, -s
                            or interval option is specified.

          -A                Show the address of the protocol control block
                            associated with sockets.  This option is used
                            for debugging.  It does not show the X.25
                            programmatic access control blocks.  This
                            option is ignored if the -g, -i, -I, -m, -M,
                            -p, -r, -s or interval option is specified.

          -f address-family Show statistics or address control block for
                            only the specified address-family.  The
                            following address families are recognized: inet
                            for AF_INET, and unix for AF_UNIX.  This option
                            applies to the -a, -A and -s options.

          -g                Show multicast information for network
                            interfaces.  Only the address family AF_INET is
                            recognized by this option.  This option may be
                            combined with the -i option to display both
```

kinds of information. The option is ignored if the -m, -M or -p option is specified.

-i Show the state of network interfaces. Interfaces that are statically configured into a system, but not located at boot time, are not shown. This option is ignored if the -m, -M or -p option is specified.

-I interface Show information about the specified interface only. This option applies to the -g and -i options.

-m Show statistics recorded by network memory management routines. If this option is specified, all other options are ignored.

-M Show the multicast routing tables. When -s is used with the -M option, netstat displays multicast routing statistics instead. This option is ignored if the -m or -p option is specified.

-n Show network addresses as numbers. Normally, netstat interprets addresses and attempts to display them symbolically. This option applies to the -a, -A, -i, -r and -v options.

-p protocol Show statistics for the specified protocol. The following protocols are recognized: tcp, udp, ip, icmp, igmp, arp, and probe. This option is ignored if the -m option is specified.

-r Show the routing tables. When -v is used with the -r option, netstat also displays the network masks in the route entries. When -s is used with the -r option, netstat displays routing statistics instead. This option is ignored if the -g, -m, -M, -i, -I, -p or interval option is specified.

-s Show statistics for all protocols. When this option is used with the -r option, netstat displays routing statistics instead. When this option is used with the -M option, netstat displays multicast routing statistics instead. This option is ignored if the -g, -i, -I, -m, -p or interval option is specified.

-v Show additional routing information. When -v is used with the -r option, netstat also displays the network masks in the route entries. This option only applies to the -r option.

The arguments system and core allow substitutes for the defaults, /stand/vmunix and /dev/kmem.

If no options or only the -A or -n option is specified, netstat displays the status of only active sockets. The display of active and passive sockets status shows the local and remote addresses, send and receive queue sizes (in bytes), protocol, and the internal state of the protocol. Address formats are of the form host.port, or network.port if the host portion of a socket address is zero. When known, the host and network addresses are displayed symbolically by using gethostbyname() and getnetbyname(), respectively (see gethostbyname(3N) and getnetbyname(3N)). If a symbolic name for an address is unknown, or if the -n option is specified, the address is displayed numerically according to the address family. For more information regarding the Internet ``dot format'', refer to inet(3N). Unspecified or ``wildcard'' addresses and ports appear as an asterisk (*).

The interface display provides a table of cumulative statistics

regarding packets transferred, errors, and collisions. The network addresses of the interface and the maximum transmission unit (MTU) are also displayed. When the interval argument is specified, netstat displays a running count of statistics related to network interfaces. This display consists of a column for the primary interface (the first interface found during auto-configuration) and a column summarizing information for all interfaces. To replace the primary interface with another interface, use the -I option. The first line of each screen of information contains a summary since the system was last rebooted. Subsequent lines of output show values accumulated over the preceding interval.

The routing table display indicates the available routes and their status. Each route consists of a destination host or network, a netmask and a gateway to use in forwarding packets. The Flags field shows whether the route is up (U), whether the route is to a gateway (G), whether the route is a host or network route (with or without H), whether the route was created dynamically (D) by a redirect or by Path MTU Discovery, and whether a gateway route has been modified (M), or it has been marked doubtful (?) due to the lack of a timely ARP response.

The Netmask field shows the mask to be applied to the destination IP address of an IP packet to be forwarded. The result will be compared with the destination address in the route entry. If they are the same, then the route is one of the candidates for routing this IP packet. If there are several candidate routes, then the route with the longest Netmask field (contiguous 1's starting from the leftmost bit position) will be chosen. (see routing (7).)

The Gateway field shows the address of the immediate gateway for reaching the destination. It can be the address of the outgoing interface if the destination is on a directly connected network.

The Refs field shows the current number of active uses of the route. Connection-oriented protocols normally hold on to a single route for the duration of a connection, while connectionless protocols normally obtain a route just while sending a particular message. The Use field shows a count of the number of packets sent using the route. The Interface field identifies which network interface is used for the route.

The Pmtu and PmtuTime fields apply only to host routes. The Pmtu field for network and default routes is the same as the MTU of the network interface used for the route. If the route is created with a static PMTU value (see route(1M)), the corresponding PmtuTime field contains the word perm, and the PMTU value permanently overrides the interface MTU. If the route is created dynamically (D in the Flags field), the value in the corresponding PmtuTime field is the number of minutes remaining before the PMTU expires. When the PMTU expires, the system rediscovers the current PMTU for the route, in case it has changed. The PmtuTime field is left blank when the PMTU is identical to the MTU of the interface. An asterisk (*) in the Pmtu field indicates that user has disabled the PMTU Discovery for the route.

DEPENDENCIES

X.25:

-A and -a options do not list X.25 programmatic access information.

AUTHOR

netstat was developed by the University of California, Berkeley.

SEE ALSO

hosts(4), networks(4), gethostbyname(3N), getnetbyname(3N), protocols(4), route(1M), services(4).

[Team LiB]

PREVIOUS NEXT

pdc



pdc - Processor Dependent Code (pdc) overview.

```
pdc(1M)                                                          pdc(1M)

NAME
     pdc - processor-dependent code (firmware)

DESCRIPTION

     pdc is the firmware that implements all processor-dependent
     functionality, including initialization and self-test of the
     processor.  Upon completion, it loads and transfers control to the
     initial system loader (isl(1M)).  Firmware behavior varies somewhat,
     depending on the hardware series as described below.

   Series 800 Behavior
     To load isl from an external medium, pdc must know the particular
     device on which isl resides.  Typically the device is identified by
     the Primary Boot Path that is maintained by pdc in Stable Storage.  A
     path specification is a series of decimal numbers each suffixed by
     '/', indicating bus converters, followed by a series of decimal
     numbers separated by '.', indicating the various card and slot numbers
     and addresses.  The first number, not specifying a bus converter, is
     the MID-BUS module number (that is, slot number times four) and
     followed by the CIO slot number.  If the CIO slot contains an HP-IB
     card, the next number is the HP-IB address, followed by the unit
     number of the device if the device supports units.  If the CIO slot
     contains a terminal card, the next number is the port number, which
     must be zero for the console.

     When the processor is reset after initialization and self-test
     complete, pdc reads the Console Path from Stable Storage, and attempts
     to initialize the console device.  If the initialization fails, pdc
     attempts to find and initialize a console device.  Algorithms used to
     find a console device are model-dependent.  pdc then announces the
     Primary Boot, Alternate Boot, and Console Paths.

     If autoboot (see isl(1M)) is enabled, pdc provides a 10-second delay,
     during which time the operator can override the autoboot sequence by
     typing any character on the console.  If the operator does not
     interrupt this process, pdc initializes and reads isl from the Primary
     Boot Path.  On models that support autosearch, if this path is not
     valid and autosearch (see isl(1M)) is enabled, pdc then searches
     through the MID-BUS modules and CIO slots to find a bootable medium.
     Currently, autosearch is only implemented on the model 825.

     If the autoboot sequence is unsuccessful, overridden by the operator,
     or not enabled in the first place, pdc interactively prompts the
     operator for the Boot Path to use.  Any required path components that
     are not supplied default to zero.

     The Primary Boot, Alternate Boot, and Console Paths as well as
     autoboot and autosearch enable can be modified via isl.

   Series 700 Behavior
     To load isl from an external medium, pdc must know the particular
     device on which isl resides.  Typically the device is identified by
     the Primary Boot Path that is maintained by pdc in Stable Storage.  A
     path specification is an I/O subsystem mnemonic that varies according
     to hardware model.
```

When the processor is reset after initialization and self-test complete, pdc reads the Console Path from Stable Storage, and attempts to initialize the console device. If the initialization fails, pdc attempts to find and initialize a console device. Algorithms used to find a console device vary according to hardware model.

If autoboot and autosearch (see isl(1M)) are enabled, pdc waits for approximately 10 seconds during which time the operator can override the autoboot sequence pressing and holding the ESC (escape) key on the console.

The system then begins a search for potentially bootable devices. If allowed to complete, a list of potentially bootable devices is displayed, labeled with abbreviated path identifiers (P0, P1, etc). A simple menu is then displayed where the user can:

- Boot a specific device, using the abbreviated path identifier, or the full mnenomic.
- Start a device search where the contents are searched for IPL images (note the first search only identified devices and did not check the contents).
- Enter the boot administration level.
- Exit the menu and return to autobooting
- Get help on choices

The search of potentially bootable devices can be aborted by pressing and holding the escape key. The search for device contents can also be aborted by pressing and holding the escape key.

If the operator does not interrupt the search process, pdc initializes and reads isl from the Primary Boot Path.

If the autoboot sequence is unsuccessful, overridden by the operator, or not enabled in the first place, pdc executes the device search and enters the menu described above.

The Primary Boot, Alternate Boot, and Console Paths as well as autoboot and autosearch enable can be modified via isl or at the pdc boot administration level.

SEE ALSO

boot(1M), hpuxboot(1M), isl(1M).

[Team LiB]

PREVIOUS NEXT

ps



ps - Report the status of processes.

```
ps(1)                                                          ps(1)

NAME
     ps - report process status

SYNOPSIS

     ps [-adeflP] [-g grplist] [-p proclist] [-R prmgrplist] [-t termlist]
     [-u uidlist]

XPG4 SYNOPSIS
     ps [-aAcdefHjlP] [-C cmdlist] [-g grplist] [-G gidlist] [-n namelist]
     [-o format] [-p proclist] [-R prmgrplist] [-s sidlist] [-t termlist]
     [-u uidlist] [-U uidlist]

DESCRIPTION
     ps prints information about selected processes.  Use options to
     specify which processes to select and what information to print about
     them.

   Process Selection Options
     Use the following options to choose which processes should be
     selected.

     NOTE: If an option is used in both the default (standard HP-UX) and
     XPG4 environments, the description provided here documents the default
     behavior.  Refer to the UNIX95 variable under EXTERNAL INFLUENCES for
     additional information on XPG4 behavior.

          (none)         Select those processes associated with the current
                         terminal.

          -A             (XPG4 Only.)  Select all processes.  (Synonym for
                         -e.)

          -a             Select all processes except process group leaders
                         and processes not associated with a terminal.

          -C cmdlist     (XPG4 Only.)  Select processes executing a command
                         with a basename given in cmdlist.

          -d             Select all processes except process group leaders.


          -e             Select all processes.

          -g grplist     Select processes whose process group leaders are
                         given in grplist.

          -G gidlist     (XPG4 Only.)  Select processes whose real group ID
                         numbers or group names are given in gidlist.

          -n namelist    (XPG4 Only.)  This option is ignored; its presence
                         is allowed for standards compliance.

          -p proclist    Select processes whose process ID numbers are
                         given in proclist.

          -R prmgrplist  Select processes belonging to PRM process resource
```

groups whose names or ID numbers are given in prmgrplist. See DEPENDENCIES.

-s sidlist (XPG4 Only.) Select processes whose session leaders are given in sidlist. (Synonym for -g).

-t termlist Select processes associated with the terminals given in termlist. Terminal identifiers can be specified in one of two forms: the device's file name (such as tty04) or if the device's file name starts with tty, just the rest of it (such as 04). If the device's file is in a directory other than /dev or /dev/pty, the terminal identifier must include the name of the directory under /dev that contains the device file (such as pts/5).

-u uidlist Select processes whose effective user ID numbers or login names are given in uidlist.

-U uidlist (XPG4 Only.) Select processes whose real user ID numbers or login names are given in uidlist.

If any of the -a, -A, -d, or -e options is specified, the -C, -g, -G, -p, -R, -t, -u, and -U options are ignored.

If more than one of -a, -A, -d, and -e are specified, the least restrictive option takes effect.

If more than one of the -C, -g, -G, -p, -R, -t, -u, and -U options are specified, processes will be selected if they match any of the options specified.

The lists used as arguments to the -C, -g, -G, -p, -R, -t, -u, and -U options can be specified in one of two forms:

- A list of identifiers separated from one another by a comma.
- A list of identifiers enclosed in quotation marks (") and separated from one another by a comma and/or one or more spaces.

Output Format Options

Use the following options to control which columns of data are included in the output listing. The options are cumulative.

(none) The default columns are: pid, tty, time, and comm, in that order.

-f Show columns user, pid, ppid, cpu, stime, tty, time, and args, in that order.

-l Show columns flags, state, uid, pid, ppid, cpu, intpri, nice, addr, sz, wchan, tty, time, and comm, in that order.

-fl Show columns flags, state, user, pid, ppid, cpu, intpri, nice, addr, sz, wchan, stime, tty, time, and args, in that order.

-c (XPG4 Only.) Remove columns cpu and nice; replace column intpri with columns cls and pri.

-j (XPG4 Only.) Add columns pgid and sid after column ppid (or pid, if ppid is not being displayed).

-P Add column prmid (for -l) or prmgrp (for -f or -fl) immediately before column pid. See DEPENDENCIES.

-o format (XPG4 Only.) format is a comma- or space-separated list of the columns to display, in the order they should be displayed. (Valid column names are listed below.) A column name can optionally be followed by an equals sign (=) and a string to use

as the heading for that column. (Any commas or spaces after the equals sign will be taken as a part of the column heading; if more columns are desired, they must be specified with additional -o options.) The width of the column will be the greater of the width of the data to be displayed and the width of the column heading. If an empty column heading is specified for every heading, no heading line will be printed. This option overrides options -c, -f, -j, -l, and -P; if they are specified, they are ignored.

-H (XPG4 Only.) Shows the process hierarchy. Each process is displayed under its parent, and the contents of the args or comm column for that process is indented from that of its parent. Note that this option is expensive in both memory and speed.

The column names and their meanings are given below. Except where noted, the default heading for each column is the uppercase form of the column name.

addr The memory address of the process, if resident; otherwise, the disk address.

args The command line given when the process was created. This column should be the last one specified, if it is desired. Only a subset of the command line is saved by the kernel; as much of the command line will be displayed as is available. The output in this column may contain spaces. The default heading for this column is COMMAND if -o is specified and CMD otherwise.

cls Process scheduling class, see rtsched(1).

comm The command name. The output in this column may contain spaces. The default heading for this column is COMMAND if -o is specified and CMD otherwise.

cpu Processor utilization for scheduling. The default heading for this column is C.

etime Elapsed time of the process. The default heading for this column is ELAPSED.

flags Flags (octal and additive) associated with the process:

0	Swapped
1	In core
2	System process
4	Locked in core (e.g., for physical I/O)
10	Being traced by another process
20	Another tracing flag

The default heading for this column is F.

intpri The priority of the process as it is stored internally by the kernel. This column is provided for backward compatibility and its use is not encouraged.

gid The group ID number of the effective process owner.

group The group name of the effective process owner.

nice Nice value; used in priority computation (see nice(1)). The default heading for this column is NI.

pcpu
The percentage of CPU time used by this process during the last scheduling interval. The default heading for this column is %CPU.

pgid
The process group ID number of the process group to which this process belongs.

pid
The process ID number of the process.

ppid
The process ID number of the parent process.

pri
The priority of the process. The meaning of the value depends on the process scheduling class; see cls, above, and rtsched(1).

prmid
The PRM process resource group ID number.

prmgrp
The PRM process resource group name.

rgid
The group ID number of the real process owner.

rgroup
The group name of the real process owner.

ruid
The user ID number of the real process owner.

ruser
The login name of the real process owner.

sid
The session ID number of the session to which this process belongs.

state
The state of the process:

0	Nonexistent
S	Sleeping
W	Waiting
R	Running
I	Intermediate
Z	Terminated
T	Stopped
X	Growing

The default heading for this column is S.

stime
Starting time of the process. If the elapsed time is greater than 24 hours, the starting date is displayed instead.

sz
The size in physical pages of the core image of the process, including text, data, and stack space. Physical page size is defined by _SC_PAGE_SIZE in the header file <unistd.h> (see sysconf(2) and unistd(5)).

time
The cumulative execution time for the process.

tty
The controlling terminal for the process. The default heading for this column is TT if -o is specified and TTY otherwise.

uid
The user ID number of the effective process owner.

user
The login name of the effective process owner.

vsz
The size in kilobytes (1024 byte units) of the core image of the process. See column sz, above.

wchan
The event for which the process is waiting or sleeping; if there is none, a hyphen (-) is displayed.

Notes

ps prints the command name and arguments given at the time of the process was created. If the process changes its arguments while running (by writing to its argv array), these changes are not displayed by ps.

A process that has exited and has a parent, but has not yet been waited for by the parent, is marked <defunct> (see zombie process in exit(2)).

The time printed in the stime column, and used in computing the value for the etime column, is the time when the process was forked, not the time when it was modified by exec*().

To make the ps output safer to display and easier to read, all control characters in the comm and args columns are displayed as "visible" equivalents in the customary control character format, ^x.

EXTERNAL INFLUENCES

Environment Variables

UNIX95 specifies to use the XPG4 behavior for this command. The changes for XPG4 include support for the entire option set specified above and include the following behavioral changes:

- The TIME column format changes from mmmm:ss to [dd-]hh:mm:ss.

- When the comm, args, user, and prmgrp fields are included by default or the -f or -l flags are used, the column headings of those fields change to CMD, CMD, USER, and PRMGRP, respectively.

- -a, -d, and -g will select processes based on session rather than on process group.

- The uid or user column displayed by -f or -l will display effective user rather than real user.

- The -u option will select users based on effective UID rather than real UID.

- The -C and -H options, while they are not part of the XPG4 standard, are enabled.

LC_TIME determines the format and contents of date and time strings. If it is not specified or is null, it defaults to the value of LANG.

If LANG is not specified or is null, it defaults to C (see lang(5)).

If any internationalization variable contains an invalid setting, all internationalization variables default to C (see environ(5)).

International Code Set Support

Single-byte character code sets are supported.

EXAMPLES

Generate a full listing of all processes currently running on your machine:

```
ps -ef
```

To see if a certain process exists on the machine, such as the cron clock daemon, check the far right column for the command name, cron, or try

```
ps -f -C cron
```

WARNINGS

Things can change while ps is running; the picture it gives is only a snapshot in time. Some data printed for defunct processes is irrelevant.

If two special files for terminals are located at the same select code, that terminal may be reported with either name. The user can select processes with that terminal using either name.

Users of ps must not rely on the exact field widths and spacing of its output, as these will vary depending on the system, the release of HP-UX, and the data to be displayed.

DEPENDENCIES

HP Process Resource Manager

The -P and -R options require the optional HP Process Resource Manager (PRM) software to be installed and configured. See prmconfig(1) for a description of how to configure HP PRM, and prmconf(4) for the definition of "process resource group."

If HP PRM is not installed and configured and -P or -R is specified, a warning message is displayed and (for -P) hyphens (-) are displayed in the prmid and prmgrp columns.

[Team LiB]

PREVIOUS NEXT

sed



sed - Stream text editor.

```
sed(1)                                                          sed(1)

NAME
     sed - stream text editor

SYNOPSIS

     sed [-n] script [file ...]

     sed [-n] [-e script] ... [-f script_file] ... [file ...]

DESCRIPTION
     sed copies the named text files (standard input default) to the
     standard output, edited according to a script containing up to 100
     commands.  Only complete input lines are processed.  Any input text at
     the end of a file that is not terminated by a new-line character is
     ignored.

   Options
     sed recognizes the following options:

          -f script_file
                     Take script from file script_file.

          -e script  Edit according to script.  If there is just one -e
                     option and no -f options, the flag -e can be omitted.


          -n         Suppress the default output.

     sed interprets all -escript and -fscript_file arguments in the order
     given.  Use caution, if mixing -e and -f options, to avoid
     unpredictable or incorrect results.

   Command Scripts
     A script consists of editor commands, one per line, of the following
     form:

          [address [, address]] function [arguments]

     In normal operation, sed cyclically copies a line of input into a
     pattern space (unless there is something left after a D command),
     applies in sequence all commands whose addresses select that pattern
     space, and, at the end of the script, copies the pattern space to the
     standard output (except under -n) and deletes the pattern space.

     Some of the commands use a hold space to save all or part of the
     pattern space for subsequent retrieval.

   Command Addresses
     An address is either a decimal number that counts input lines
     cumulatively across files, a $ which addresses the last line of input,
     or a context address; that is, a /regular expression/ in the style of
     ed(1) modified thus:

          -  In a context address, the construction \?regular expression?,
             where ? is any character, is identical to /regular
             expression/.  Note that in the context address \xabc\xdefx,
             the second x stands for itself, so that the regular expression
```

is abcxdef.

- The escape sequence \n matches a new-line character embedded in the pattern space.

- A period (.) matches any character except the terminal new-line of the pattern space.

- A command line with no addresses selects every pattern space.

- A command line with one address selects each pattern space that matches the address.

- A command line with two addresses selects the inclusive range from the first pattern space that matches the first address through the next pattern space that matches the second (if the second address is a number less than or equal to the line number first selected, only one line is selected). Thereafter the process is repeated, looking again for the first address.

sed supports Basic Regular Expression syntax (see regexp(5)).

Editing commands can also be applied to only non-selected pattern spaces by use of the negation function ! (described below).

Command Functions

In the following list of functions, the maximum number of permissible addresses for each function is indicated in parentheses. Other function elements are interpreted as follows:

text	One or more lines, all but the last of which end with \ to hide the new-line. Backslashes in text are treated like backslashes in the replacement string of an s command, and can be used to protect initial blanks and tabs against the stripping that is done on every script line.
rfile	Must terminate the command line, and must be preceded by exactly one blank.
wfile	Must terminate the command line, and must be preceded by exactly one blank. Each wfile is created before processing begins. There can be at most 10 distinct wfile arguments.

sed recognizes the following functions:

(1)a\ text	Append. Place text on the output before reading next input line.
(2)b label	Branch to the : command bearing label. If no label is specified, branch to the end of the script.
(2)c\ text	Change. Delete the pattern space. With 0 or 1 address or at the end of a 2-address range, place text on the output. Start the next cycle.
(2)d	Delete pattern space and start the next cycle.
(2)D	Delete initial segment of pattern space through first new-line and start the next cycle.
(2)g	Replace contents of the pattern space with contents of the hold space.
(2)G	Append contents of hold space to the pattern space.
(2)h	Replace contents of the hold space with contents of the pattern space.
(2)H	Append the contents of the pattern space to the hold space.

(1)i\
text Insert. Place text on the standard output.

(2)l List the pattern space on the standard output in an unambiguous form. Non-printing characters are spelled in three-digit octal number format (with a preceding backslash), and long lines are folded.

(2)n Copy the pattern space to the standard output if the default output has not been suppressed (by the -n option on the command line or the #n command in the script file). Replace the pattern space with the next line of input.

(2)N Append the next line of input to the pattern space with an embedded new-line. (The current line number changes.)

(2)p Print. Copy the pattern space to the standard output.

(2)P Copy the initial segment of the pattern space through the first new-line to the standard output.

(1)q Quit. Branch to the end of the script. Do not start a new cycle.

(1)r rfile Read contents of rfile and place on output before reading the next input line.

(2)s/regular expression/replacement/flags
Substitute replacement string for instances of regular expression in the pattern space. Any character can be used instead of /. For a fuller description see ed(1). flags is zero or more of:

- n n=1-2048 (LINE_MAX). Substitute for just the nth occurrence of regular expression in the pattern space.
- g Global. Substitute for all non-overlapping instances of regular expression rather than just the first one.
- p Print the pattern space if a replacement was made and the default output has been suppressed (by the -n option on the command line or the #n command in the script file).
- w wfile Write. Append the pattern space to wfile if a replacement was made.

(2)t label Test. Branch to the : command bearing the label if any substitutions have been made since the most recent reading of an input line or execution of a t. If label is empty, branch to the end of the script.

(2)w wfile Write. Append the pattern space to wfile.

(2)x Exchange the contents of the pattern and hold spaces.

(2)y/string1/string2/
Transform. Replace all occurrences of characters in string1 with the corresponding character in string2. The lengths of string1 and string2 must be equal.

(2)! function
Don't. Apply the function (or group, if function is {) only to lines not selected by the address or addresses.

(0): label This command does nothing; it bears a label for b and t commands to branch to.

(1)= Place the current line number on the standard output as a line.

(2){ Execute the following commands through a matching } only when the pattern space is selected. The syntax is:

```
{ cmd1
cmd2
cmd3
 .
 .
 .
}
```

(0) An empty command is ignored.

(0)# If a # appears as the first character on the first line of a script file, that entire line is treated as a comment with one exception: If the character after the # is an n, the default output is suppressed. The rest of the line after #n is also ignored. A script file must contain at least one non-comment line.

EXTERNAL INFLUENCES

Environment Variables

LANG provides a default value for the internationalization variables that are unset or null. If LANG is unset or null, the default value of "C" (see lang(5)) is used. If any of the internationalization variables contains an invalid setting, sed will behave as if all internationalization variables are set to "C". See environ(5).

LC_ALL If set to a non-empty string value, overrides the values of all the other internationalization variables.

LC_CTYPE determines the interpretation of text as single and/or multi-byte characters, the classification of characters as printable, and the characters matched by character class expressions in regular expressions.

LC_MESSAGES determines the locale that should be used to affect the format and contents of diagnostic messages written to standard error and informative messages written to standard output.

NLSPATH determines the location of message catalogues for the processing of LC_MESSAGES.

International Code Set Support

Single- and multi-byte character code sets are supported.

EXAMPLES

Make a simple substitution in a file from the command line or from a shell script, changing abc to xyz:

```
sed 's/abc/xyz/' file1 >file1.out
```

Same as above but use shell or environment variables var1 and var2 in search and replacement strings:

```
sed "s/$var1/$var2/" file1 >file1.out
```

or

```
sed 's/'$var1'/'$var2'/' file1 >file1.out
```

Multiple substitutions in a single command:

```
sed -e 's/abc/xyz/' -e 's/lmn/rst/' file1
```

or

```
sed -e 's/abc/xyz/' \
-e 's/lmn/rst/' \
file1 >file1.out
```

WARNINGS

sed limits command scripts to a total of not more than 100 commands.

```
     The hold space is limited to 8192 characters.

     sed processes only text files.  See the glossary for a definition of
     text files and their limitations.

AUTHOR
     sed was developed by OSF and HP.

SEE ALSO
     awk(1), ed(1), grep(1), environ(5), lang(5), regexp(5).

     sed: A Non-Interactive Streaming Editor tutorial in the Text
     Processing Users Guide.

STANDARDS CONFORMANCE
     sed: SVID2, SVID3, XPG2, XPG3, XPG4, POSIX.2
```

[Team LiB]

PREVIOUS NEXT

shutdown



shutdown - Terminate all running processes in an orderly fashion.

```
shutdown(1M)                                                    shutdown(1M)

NAME
     shutdown - terminate all processing

SYNOPSIS

     /sbin/shutdown [-h|-r] [-y] [-o] [grace]

DESCRIPTION
     The shutdown command is part of the HP-UX system operation procedures.
     Its primary function is to terminate all currently running processes
     in an orderly and cautious manner.  shutdown can be used to put the
     system in single-user mode for administrative purposes such as backup
     or file system consistency checks (see fsck(1M)), and to halt or
     reboot the system.  By default, shutdown is an interactive program.

   Options and Arguments
     shutdown recognizes the following options and arguments.

          -h        Shut down the system and halt.

          -r        Shut down the system and reboot automatically.

          -y        Do not require any interactive responses from the user.
                    (Respond yes or no as appropriate to all questions,
                    such that the user does not interact with the shutdown
                    process.)

          -o        When executed on the cluster server in a diskless
                    cluster environment, shutdown the server only and do
                    not reboot clients. If this argument is not entered the
                    default behavior is to reboot all clients when the
                    server is shutdown.

          grace     Either a decimal integer that specifies the duration in
                    seconds of a grace period for users to log off before
                    the system shuts down, or the word now.  The default is
                    60.  If grace is either 0 or now, shutdown runs more
                    quickly, giving users very little time to log out.

          If neither -r (reboot) nor -h (halt) is specified, standalone and
          server systems are placed in single-user state.  Either -r
          (reboot) or -h (halt) must be specified for a client; shutdown to
          single-user state is not allowed for a client. See dcnodes(1M),
          init(1M).

   Shutdown Procedure
     shutdown goes through the following steps:

          -  The PATH environment variable is reset to
             /usr/bin:/usr/sbin:/sbin.

          -  The IFS environment variable is reset to space, tab, newline.

          -  The user is checked for authorization to execute the shutdown
             command.  Only authorized users can execute the shutdown
             command.  See FILES for more information on the
             /etc/shutdown.allow authorization file.
```

- The current working directory is changed to the root directory (/).

- All file systems' super blocks are updated; see sync(1M). This must be done before rebooting the system to ensure file system integrity.

- The real user ID is set to that of the superuser.

- A broadcast message is sent to all users currently logged in on the system telling them to log out. The administrator can specify a message at this time; otherwise, a standard warning message is displayed.

- The next step depends on whether a system is standalone, a server, or a client.

  - If the system is standalone, /sbin/rc is executed to shut down subsystems, unmount file systems, and perform other tasks to bring the system to run level 0.

  - If the system is a server, the optional -o argument is used to determine if all clients in the cluster should also be rebooted. The default behavior (command line parameter -o is not entered) is to reboot all clients using /sbin/reboot; entering -o results in the server only being rebooted and the clients being left alone. Then /sbin/rc is executed to shut down subsystems, unmount file systems, and perform other tasks to bring the system to run level 0.

  - If the system is a client, /sbin/rc is executed to bring the system down to run-level 2, and then /sbin/reboot is executed. Shutdown to the single-user state is not an allowed option for clients.

- The system is rebooted or halted by executing /sbin/reboot if the -h or -r option was chosen. If the system was not a cluster client and the system was being brought down to single-user state, a signal is sent to the init process to change states (see init(1M)).

DIAGNOSTICS

device busy

This is the most commonly encountered error diagnostic, and happens when a particular file system could not be unmounted; see mount(1M).

user not allowed to shut down this system

User is not authorized to shut down the system. User and system must both be included in the authorization file /etc/shutdown.allow.

EXAMPLES

Immediately reboot the system and run HP-UX again:

```
shutdown -r 0
```

Halt the system in 5 minutes (300 seconds) with no interactive questions and answers:

```
shutdown -h -y 300
```

Go to run-level s in 10 minutes:

```
shutdown 600
```

FILES

/etc/shutdown.allow

Authorization file.

The file contains lines that consist of a system host name and the login name of a user who is authorized to reboot or halt the system. A superuser's login name must be included in this file in order to execute shutdown. However, if the file is missing or of zero length, the root user can run the shutdown program to bring the system down.

This file does not affect authorization to bring the system down to single-user state for maintenance purposes; that operation is permitted only when invoked by a superuser.

A comment character, #, at the beginning of a line causes the rest of the line to be ignored (comments cannot span multiple lines without additional comment characters). Blank lines are also ignored.

The wildcard character + can be used in place of a host name or a user name to specify all hosts or all users, respectively (see hosts.equiv(4)).

For example:

```
# user1 can shut down systemA and systemB
systemA user1
systemB user1
# root can shut down any system
+ root
# Any user can shut down systemC
systemC  +
```

WARNINGS

The user name compared with the entry in the shutdown.allow file is obtained using getlogin() or, if that fails, using getpwuid() (see getlogin(3) and getpwuid(3)).

The hostname in /etc/shutdown.allow is compared with the hostname obtained using gethostbyname() (see gethostbyname(3)).

shutdown must be executed from a directory on the root volume, such as the / directory.

The maximum broadcast message that can be sent is approximately 970 characters.

When executing shutdown on an NFS diskless cluster server and the -o option is not entered, clients of the server will be rebooted. No clients should be individually rebooted or shutdown while the cluster is being shutdown.

SEE ALSO

dcnodes(1M), fsck(1M), init(1M), killall(1M), mount(1M), reboot(1M), sync(1M), dcnodes(3), gethostbyname(3), getpwuid(3), hosts.equiv(4).

[Team LiB]

PREVIOUS NEXT

"sw" command summaries



"sw" commands - Command summaries related to software distribution.

```
swacl(1M)                 Hewlett-Packard Company                 swacl(1M)

NAME
     swacl - View or modify the Access Control Lists (ACLs) which protect
     software products

SYNOPSIS

     swacl -l level [-M acl_entry| -D acl_entry| -F acl_file] [-x
     option=value]
     [-X option_file] [-f software_file] [-t target_file]
     [software_selections] [@ target_selections]




swagentd(1M)              Hewlett-Packard Company              swagentd(1M)
swagent(1M)                                                     swagent(1M)

NAME

     swagentd - Serve local or remote SD software management tasks,
     including invoking a swagent command.

     swagent - Perform SD software management tasks as the agent of an SD
     command.

     SWAGENTD.EXE - Perform HP OpenView Software Distributor PC software
     management tasks, or serve local PC software for distribution. See
     "Remarks:" below.

SYNOPSIS

     swagentd [-k] [-n] [-r] [-x option=value] [-X option_file]

     SWAGENTD.EXE (HP OpenView Software Distributor only)



swcluster(1M)                                                 swcluster(1M)

NAME
     swcluster - install or remove software from diskless server

SYNOPSIS

     swcluster [XToolkit Options] [-v][v] [-i] [-p] [-f] [-r] [-b] [-l
     list_class] [-n] [-s source]
     [-C session_file] [-S session_file] [-x option=value] [-X option_file]
     [software_selections] [@ target_selections]



swconfig(1M)              Hewlett-Packard Company              swconfig(1M)

NAME
```

swconfig - Configure, unconfigure, or reconfigure installed software

SYNOPSIS

swconfig [-p] [-v] [-u] [-x option=value] [-X option_file] [-f software_file] [-t target_file]
[-C session_file] [-S session_file] [-Q date] [-J jobid]
[software_selections] [@ target_selections]

swgettools(1M) Hewlett-Packard Company swgettools(1M)

NAME

swgettools - Utility for retrieving the SD product from new SD media

SYNOPSIS

swgettools -s <source_media_location> [-t <temp_dir_location>]

swinstall(1M) Hewlett-Packard Company swinstall(1M)
swcopy(1M) swcopy(1M)

NAME

swinstall - Install and configure software products

swcopy - Copy software products for subsequent installation or distribution

SYNOPSIS

swinstall [XToolkit Options] [-i] [-p] [-v] [-r] [-s source] [-x option=value] [-X option_file]
[-f software_file] [-t target_file] [-C session_file] [-S session_file]
[-Q date] [-J jobid]
[software_selections] [@ target_selections]

swcopy [XToolkit Options] [-i] [-p] [-v] [-s source] [-x option=value]
[-X option_file]
[-f software_file] [-t target_file] [-C session_file] [-S session_file]
[-Q date] [-J jobid]
[software_selections] [@ target_selections]

swjob(1M) Hewlett-Packard Company swjob(1M)
sd(1M) sd(1M)

NAME

swjob - Display job information and remove jobs.

sd - Interactive interface for creating and monitoring jobs.

For a description of the HP OpenView Software Distributor objects, attributes and data formats, see the sd(4) manual page by typing:
man 4 sd

For an overview of all HP OpenView Software Distributor commands, see the sd(5) manual page by typing:
man 5 sd

SYNOPSIS

swjob [-i] [-u] [-v] [-R] [-a attribute] [-x option=value] [-X option_file] [-f jobid_file]
[-t target_file] [-C session_file] [-S session_file] [jobid(s)] [
@ target_selections]

```
     sd [-x option=value] [-X option_file]
```

```
swlist(1M)                Hewlett-Packard Company                swlist(1M)

NAME
     swlist - Display information about software products

SYNOPSIS

     swlist [-d|-r] [-l level] [-v] [-a attribute] [-R] [-s source] [-x
     option=value] [-X option_file]
     [-f software_file] [-t target_file] [-C session_file] [-S session_file]
     [software_selections] [@ target_selections]
```

```
swmodify(1M)              Hewlett-Packard Company              swmodify(1M)

NAME
     swmodify - Modify software products in a target root or depot

SYNOPSIS

     swmodify [-d|-r] [-p] [-P pathname_file] [-v[v]] [-V] [-u]
     [-s product_specification_file| -a attribute=[value]] [-x option=value]
     [-X option_file]
     [-f software_file] [-C session_file] [-S session_file]
     [software_selections] [@ target_selection]
```

```
swpackage(1M)             Hewlett-Packard Company             swpackage(1M)

NAME
     swpackage - Package software products into a target depot or tape

     For a description of the Product Specification File used as input to
     the swpackage command, see the swpackage(4) manual page by typing:
          man 4 swpackage

SYNOPSIS

     swpackage [-p] [-v[v]] [-V] [-s product_specification_file|directory]
     [-d directory|device]
     [-x option=value] [-X option_file] [-f software_file] [-C session_file]
     [-S session_file]
     [software_selections] [@ target_selection]
```

```
swreg(1M)                 Hewlett-Packard Company                 swreg(1M)

NAME
     swreg - Register or unregister depots and roots

SYNOPSIS

     swreg -l level [-u] [-v] [-x option=value] [-X option_file] [-f
     object_file] [-t target_file]
     [-C session_file] [-S session_file] [objects_to_(un)register] [
     @ target_selections]
```

```
swremove(1M)              Hewlett-Packard Company              swremove(1M)

NAME
     swremove - Unconfigure and remove software products

SYNOPSIS

     swremove [XToolkit Options] [-i] [-p] [-v] [-d|-r] [-x option=value]
     [-X option_file] [-f software_file] [-t target_file] [-C session_file]
     [-S session_file] [-Q date] [-J jobid] [software_selections] [
     @ target_selections]
```

```
swverify(1M)              Hewlett-Packard Company              swverify(1M)

NAME
     swverify - Verify software products

SYNOPSIS

     swverify [-v] [-d|-r] [-x option=value] [-X option_file] [-f
     software_file] [-t target_file]
     [-C session_file] [-S session_file] [-Q date] [-J jobid]
     [software_selections] [@ target_selections]
```

[Team LiB]

PREVIOUS NEXT

top



top - Provide information about top processes on the system.

```
top(1)                                                          top(1)

NAME
     top - display and update information about the top processes on the
     system

SYNOPSIS

     top [-s time] [-d count] [-q] [-u] [-n number]

DESCRIPTION
     top displays the top processes on the system and periodically updates
     the information.  Raw CPU percentage is used to rank the processes.

   Options
     top recognizes the following command-line options:

          -s time     Set the delay between screen updates to time seconds.
                      The default delay between updates is 5 seconds.

          -d count    Show only count displays, then exit.  A display is
                      considered to be one update of the screen.  This
                      option is used to select the number of displays to be
                      shown before the program exits.

          -q          This option runs the top program at the same priority
                      as if it is executed via a nice -20 command so that
                      it will execute faster (see nice(1)).  This can be
                      very useful in discovering any system problem when
                      the system is very sluggish.  This option is
                      accessibly only to users who have appropriate
                      privileges.

          -u          User ID (uid) numbers are displayed instead of
                      usernames.  This improves execution speed by
                      eliminating the additional time required to map uid
                      numbers to user names.

          -n number   Show only number processes per screen.  Note that
                      this option is ignored if number is greater than the
                      maximum number of processes that can be displayed per
                      screen.

   Screen-Control Commands
     When displaying multiple-screen data, top recognizes the following
     keyboard screen-control commands:

          j           Display next screen if the current screen is not the
                      last screen.

          k           Display previous screen if the current screen is not
                      the first screen.

          t           Display the first (top) screen.

   Program Termination
     To exit the program and resume normal user activities, type q at any
     time.
```

Display Description

Three general classes of information are displayed by top:

System Data:

The first few lines at the top of the display show general information about the state of the system, including:

- System name and current time.
- Load averages in the last one, five, and fifteen minutes.
- Number of existing processes and the number of processes in each state (sleeping, waiting, running, starting, zombie, and stopped).
- Percentage of time spent in each of the processor states (user, nice, system, idle, interrupt and swapper) per processor on the system.
- Average value for each of the processor states (only on multi-processor systems).

Memory Data

Includes virtual and real memory in use (with the amount of memory considered "active" in parentheses) and the amount of free memory.

Process Data

Information about individual processes on the system. When process data cannot fit on a single screen, top divides the data into two or more screens. To view multiple-screen data, use the j, k, and t commands described previously. Note that the system- and memory-data displays are present in each screen of multiple-screen process data.

Process data is displayed in a format similar to that used by ps(1):

CPU	Processor number on which the process is executing (only on multi-processor systems).
TTY	Terminal interface used by the process.
PID	Process ID number.
USERNAME	Name of the owner of the process. When the -u option is specified, the user ID (uid) is displayed instead of USERNAME.
PRI	Current priority of the process.
NI	Nice value ranging from -20 to +20.
SIZE	Total size of the process in kilobytes. This includes text, data, and stack.
RES	Resident size of the process in kilobytes. The resident size information is, at best, an approximate value.
STATE	Current state of the process. The various states are sleep, wait, run, idl, zomb, or stop.
TIME	Number of system and CPU seconds the process has consumed.
%WCPU	Weighted CPU (central processing unit) percentage.
%CPU	Raw CPU percentage. This field is used to sort the top processes.

```
          COMMAND     Name of the command the process is
                      currently running.

EXAMPLES
     top can be executed with or without command-line options.  To display
     five screens of data at two-second intervals then automatically exit,
     use:

          top -s2 -d5

AUTHOR
     top was developed by HP and William LeFebvre of Rice University.
```

[Team LiB]

PREVIOUS NEXT

vmstat



vmstat - Report process, virtual memory, trap, and CPU activity.

```
vmstat(1)                                                    vmstat(1)

NAME
     vmstat - report virtual memory statistics

SYNOPSIS

     vmstat [-dnS] [interval [count]]

     vmstat -f | -s | -z

DESCRIPTION
     The vmstat command reports certain statistics kept about process,
     virtual memory, trap, and CPU activity.  It also can clear the
     accumulators in the kernel sum structure.

   Options
     vmstat recognizes the following options:

          -d        Report disk transfer information as a separate section,
                    in the form of transfers per second.

          -n        Provide an output format that is more easily viewed on
                    an 80-column display device.  This format separates the
                    default output into two groups: virtual memory
                    information and CPU data.  Each group is displayed as a
                    separate line of output.  On multiprocessor systems,
                    this display format also provides CPU utilization on a
                    per CPU basis.

          -S        Report the number of processes swapped in and out (si
                    and so) instead of page reclaims and address
                    translation faults (re and at).

          interval  Display successive lines which are summaries over the
                    last interval seconds.  If interval is zero, the output
                    is displayed once only.  If the -d option is specified,
                    the column headers are repeated.  If -d is omitted, the
                    column headers are not repeated.

                    The command vmstat 5 prints what the system is doing
                    every five seconds.  This is a good choice of printing
                    interval since this is how often some of the statistics
                    are sampled in the system; others vary every second.

          count     Repeat the summary statistics count times.  If count is
                    omitted or zero, the output is repeated until an
                    interrupt or quit signal is received.  From the
                    terminal, these are commonly ^C and ^\, respectively
                    (see stty(1)).

          -f        Report on the number of forks and the number of pages
                    of virtual memory involved since boot-up.

          -s        Print the total number of several kinds of paging-
                    related events from the kernel sum structure that have
                    occurred since boot-up or since vmstat was last
                    executed with the -z option.
```

-z Clear all accumulators in the kernel sum structure. This requires write file access permission on /dev/kmem. This is normally restricted to users with appropriate privileges.

If none of these options is given, vmstat displays a one-line summary of the virtual memory activity since boot-up or since the -z option was last executed.

Column Descriptions

The column headings and the meaning of each column are:

procs Information about numbers of processes in various states.

- **r** In run queue
- **b** Blocked for resources (I/O, paging, etc.)
- **w** Runnable or short sleeper (< 20 secs) but swapped

memory Information about the usage of virtual and real memory. Virtual pages are considered active if they belong to processes that are running or have run in the last 20 seconds.

- **avm** Active virtual pages
- **free** Size of the free list

page Information about page faults and paging activity. These are averaged each five seconds, and given in units per second.

- **re** Page reclaims (without -S)
- **at** Address translation faults (without -S)
- **si** Processes swapped in (with -S)
- **so** Processes swapped out (with -S)
- **pi** Pages paged in
- **po** Pages paged out
- **fr** Pages freed per second
- **de** Anticipated short term memory shortfall
- **sr** Pages scanned by clock algorithm, per second

faults Trap/interrupt rate averages per second over last 5 seconds.

- **in** Device interrupts per second (nonclock)
- **sy** System calls per second
- **cs** CPU context switch rate (switches/sec)

cpu Breakdown of percentage usage of CPU time

- **us** User time for normal and low priority processes
- **sy** System time
- **id** CPU idle

EXAMPLES

The following examples show the output for various command options. For formatting purposes, some leading blanks have been deleted.

1. Display the default output.

```
vmstat

     procs           memory                  page
           faults       cpu
r     b     w      avm    free   re   at    pi   po    fr   de    sr
     in     sy    cs  us sy id
0     0     0     1158     511    0    0     0    0     0    0     0
    111     18     7   0  0 100
```

2. Add the disk tranfer information to the default output.

```
vmstat -d

     procs           memory                  page
           faults       cpu
r     b     w      avm    free   re   at    pi   po    fr   de    sr
     in     sy    cs  us sy id
0     0     0     1158     511    0    0     0    0     0    0     0
    111     18     7   0  0 100

Disk Transfers
  device    xfer/sec
  c0t6d0         0
  c0t1d0         0
  c0t3d0         0
  c0t5d0         0
```

3. Display the default output in 80-column format.

```
vmstat -n

VM
    memory                      page                           faults
  avm    free   re   at    pi   po    fr   de    sr     in     sy    cs
 1158     430    0    0     0    0     0    0     0    111     18     7
CPU
    cpu          procs
 us sy  id    r    b     w
  0  0 100    0    0     0
```

4. Replace the page reclaims and address translation faults with process swapping in the default output.

```
vmstat -S

     procs           memory                  page
           faults       cpu
r     b     w      avm    free   si   so    pi   po    fr   de    sr
     in     sy    cs  us sy id
0     0     0     1158     430    0    0     0    0     0    0     0
    111     18     7   0  0 100
```

5. Display the default output twice at five-second intervals. Note that the headers are not repeated.

```
vmstat 5 2

     procs           memory                  page
           faults       cpu
r     b     w      avm    free   re   at    pi   po    fr   de    sr
     in     sy    cs  us sy id
0     0     0     1158     456    0    0     0    0     0    0     0
    111     18     7   0  0 100
0     0     0     1221     436    5    0     5    0     0    0     0
    108     65    18   0  1 99
```

6. Display the default output twice in 80-column format at five-second intervals. Note that the headers are not repeated.

```
vmstat -n 5 2

VM
   memory                     page                            faults
 avm    free   re   at    pi   po    fr   de    sr     in     sy    cs
1221     436    0    0     0    0     0    0     0    111     18     7
CPU
    cpu          procs
 us sy id    r     b     w
  0  0 100    0     0     0
1221     435    2    0     2    0     0    0     0    109     35    17
  0  1 99     0     0     0
```

7. Display the default output and disk transfers twice in 80-column format at five-second intervals. Note that the headers are repeated.

```
vmstat -dn 5 2

VM

   memory                     page                            faults
 avm    free   re   at    pi   po    fr   de    sr     in     sy    cs
1221     435    0    0     0    0     0    0     0    111     18     7
CPU
    cpu          procs
 us sy id    r     b     w
  0  0 100    0     0     0

Disk Transfers
  device    xfer/sec
  c0t6d0         0
  c0t1d0         0
  c0t3d0         0
  c0t5d0         0

VM
   memory                     page                            faults
 avm    free   re   at    pi   po    fr   de    sr     in     sy    cs
1219     425    0    0     0    0     0    0     0    111     54    15
CPU
    cpu          procs
 us sy id    r     b     w
  1  8 92     0     0     0

Disk Transfers
  device    xfer/sec
  c0t6d0         0
  c0t1d0         0
  c0t3d0         0
  c0t5d0         0
```

8. Display the number of forks and pages of virtual memory since boot-up.

```
vmstat -f

24558 forks, 1471595 pages, average=  59.92
```

9. Display the counts of paging-related events.

```
vmstat -s

0 swap ins
0 swap outs
0 pages swapped in
0 pages swapped out
1344563 total address trans. faults taken
542093 page ins
2185 page outs
602573 pages paged in
```

```
   4346 pages paged out
   482343 reclaims from free list
   504621 total page reclaims
   124 intransit blocking page faults
   1460755 zero fill pages created
   404137 zero fill page faults
   366022 executable fill pages created
   71578 executable fill page faults
   0 swap text pages found in free list
   162043 inode text pages found in free list
   196 revolutions of the clock hand
   45732 pages scanned for page out
   4859 pages freed by the clock daemon
   36680636 cpu context switches
   1497746186 device interrupts
   1835626 traps
   87434493 system calls

WARNINGS
     Users of vmstat must not rely on the exact field widths and spacing of
     its output, as these will vary depending on the system, the release of
     HP-UX, and the data to be displayed.

AUTHOR
     vmstat was developed by the University of California, Berkeley and HP.

FILES
     /dev/kmem

SEE ALSO
     iostat(1).
```

[Team LiB]

PREVIOUS NEXT



Appendix B. Virtual Partitions Online Manual Pages

- Man Pages
- vparboot
- vparcreate
- vparmodify
- vparremove
- vparreset
- vparresources
- vparstatus
- vpartition
- vparutil



Man Pages

The commands for working with Virtual Partitions appear in every chapter in this book. Many examples are provided in the chapters. In addition, the Quick Reference Tear Out card has on it summaries and examples of all of the commands. There is, however, nothing like the command detail that appears in the man pages for a command. The man pages for Virtual Partitions commands are well written and thorough. This appendix consists of the manual pages for Virtual Partitions commands only. The following is a list of the commands:

vparboot Boot (start) a virtual partition.

vparcreate Create a new virtual partition.

vparmodify Modify an existing virtual partition.

vparremove Remove (delete) an existing virtual partition.

vparreset Send a TOC or hard reset to a virtual partition.

vparresources: Provides description of Virtual Partitions and their resources.

vparstatus Display virtual partition and available resources information.

vpartition: Displays information about the vPars Command Line Interface

vparutil Work with SCSI-related information of disks in a vPar.

vparboot



vparboot - Boots (starts) a virtual partition.

```
vparboot(1M)                                                vparboot(1M)


NAME
     vparboot - boot a virtual partition

SYNOPSIS
     vparboot -p vp_name [-b kernel_path] [-o boot_opts] [-B boot_addr]

     vparboot -p vp_name -I ignite_kernel

DESCRIPTION
     The vparboot command causes the virtual partition monitor to boot the
     specified virtual partition.  The monitor must be running.  The
     virtual partition must exist in the monitor configuration and be in
     the Down state. For example, not running. For this reason, it is not
     possible to boot the current partition using this command.  The
     partition in which the command is executing is called the current
     partition.

     Only a superuser can execute the vparboot command.

   Options
     vparboot recognizes the following command line options and arguments:

        -p vp_name      Specifies the unique name of the virtual partition
                        which is to be booted.  The virtual partition must
                        exist in the monitor's database, must be in the
                        Down state, and must have a configured boot
                        device.  Required.

        -b kernel_path  Specifies the absolute path to a bootable kernel
                        for the virtual partition.  For example, if a
                        non-partitioned system start string at the ISL
                        prompt is:

 ISL> hpux -iS /stand/vmunix

 the kernel-path is the /stand/vmunix portion of
 this string.

 If this option is omitted, the value is taken from
 any -b kernel_path specified during virtual
 partition creation or its most recent
 modification, or /stand/vmunix if -b kernel_path
 has never been specified.

        -o boot_opts    Specifies the command-line options applied when
                        the virtual partition is booted.  For example, if
                        a non-partitioned system start string at the ISL
                        prompt is:

 ISL> hpux -iS /stand/vmunix

 the boot_opts string in this case will be -iS.
 Note that if any whitespace is included in the
 string, the string must be quoted.
```

Refer to the virtual partitions administration guide and the chapter titled "Monitor and Shell Commands" for a full list of supported boot option strings.

If the -o option is omitted, the value is taken from any -o boot_opts specified during virtual partition creation or its most recent modification, or the empty string if -o boot_opts has never been specified.

-B boot_addr Specifies the device from which the virtual partition kernel image is read. It must be one of the following three forms:

bo[ot] | pri[mary]
(case-insensitive). This is the default if the -B option is omitted. vparboot attempts to boot vp_name from the device previously configured with the BOOT attribute. It is an error if no device has been so configured.

alt[ernate|boot]
(case-insensitive). vparboot attempts to boot vp_name from the device previously configured with the ALTBOOT attribute. It is an error if no device has been so configured.

Boot From Hardware Path:
Any other string is assumed to be the hardware path (for example, 10/6.0) to the desired boot device. vparboot does not check any part of such a specification. If it is not a proper path, the path does not exist, the virtual partition does not own the device at that path, or there is no bootable device at that path, the command either fails with an error, or the boot attempt fails with no error indication.

-I ignite_kernel Specifies a path to a cold-install kernel such as /opt/ignite/boot/WINSTALL. This form of the command is used to cold-install an instance of HP-UX on a virtual partition. Most often, this is done on a newly-created virtual partition, one that has a completely unwritten disk. However, it can be used to cold-install HP-UX over an existing instance.

ignite_kernel can reside either on the current virtual partition (the one on which vparboot is run) or on any other networked system or virtual partition accessible from that partition. The syntax for ignite_kernel is:

[Server name or IP addr,]cold-install_kernel

If cold-install_kernel begins with a slash, "/", you are specifying an absolute path and vparboot looks for it at that location. If cold-install_kernel does not begin with "/", the default path /opt/ignite/boot/ is prepended to cold-install_kernel.

If the Server name or IP address, is not specified, vparboot loads the cold-install kernel from the filespace of the current virtual partition. Otherwise the command loads it from the filespace of the specified server. The comma is a required part of the server name syntax.

RETURN VALUE

The vparboot command exits with one of the following values:

```
     0   Successful completion.
     1   One or more error conditions occurred.

EXAMPLES
     Boot the virtual partition called Oslo

  vparboot -p Oslo

     Boot the virtual partition Bergen specifying an alternate kernel

  vparboot -p Bergen -b /stand/vmunix_debug

     Cold-install the virtual partition Trondheim from a file on a remote
     server

  vparboot -p Trondheim -I mysys.myserver.mydomain.com,/mypath/myinst

     Cold-install the virtual partition Sandefjord from a file at the
     default path /opt/ignite/boot on the current virtual partition

  vparboot -p Sandefjord -I WINSTALL

ERRORS
     vparboot displays error messages on stderr for any of the following
     conditions:

     vp_name does not exist in the monitor.

     vp_name is in some state other than Down, and so cannot be
     booted.  Use the vparreset(1M) command instead.

     An error occurs while trying to access an Ignite server or cold-
     install file.

     You have specified the BOOT or ALTBOOT device, but the
     corresponding device has not been configured in vp_name.

     NOTE: If the boot device has not been configured properly the
     vparboot command will complete without error, but the virtual
     partition will not boot.  Two possible reasons for this could be:

  +  The BOOT or ALTBOOT attribute was assigned to a device
     other than a disc or tape.

  +  The file at the default or configured kernel path does
     not exist or is not a bootable image.

   AUTHOR
        vparboot was developed by the Hewlett-Packard Company.

   SEE ALSO
        vparcreate(1M), vparmodify(1M), vparremove(1M), vparreset(1M),
        vparstatus(1M), vparresources(5), vpartition(5).
```

[Team LiB]

◀ PREVIOUS NEXT ▶

vparcreate



vparcreate - Creates a new virtual partition.

```
vparcreate(1M)                                          vparcreate(1M)


NAME
     vparcreate - create a virtual partition

SYNOPSIS
     vparcreate -p vp_name [-B boot_attr] [-D db_file] [-S static_attr]
  [-b kernel_path] [-o boot_opts] [-a rsrc]...

DESCRIPTION
     The vparcreate command creates a new virtual partition (vPar) using
     the specified hardware resources.       vp_name is a symbolic name for the
     virtual partition and must be used in all references to it.  If the -D
     option is specified, the virtual partition is created in db_file.
     Otherwise it is created in the virtual partition monitor database.  In
     either case, a virtual partition named vp_name must not already exist
     in the database.   Any error terminates the command without creating a
     virtual partition.

     Only a superuser can execute the vparcreate command.

   Options
     No option except -a (add resource) may be specified more than once.
     Resources allocated with the -a option must be available, that is, not
     already allocated to a virtual partition, nor exceeding the overall
     limits of the resources.    Resources assigned with multiple -a options
     must not conflict. Violation of any condition is a command error.

     The vparcreate command recognizes the following command line options
     and arguments:

        -p vp_name      Specifies the symbolic name of the virtual
                        partition.  The name can consist of alpha-numeric
                        characters A-Z, a-z, 0-9, the underbar character
                        (_), and the period (.).  The maximum length of
                        the name is 239 characters.  Required.

        -B boot_attr    Specifies the autoboot attribute of the virtual
                        partition.  boot_attr can have the following
                        (case-insensitive) values:

 auto      sets the autoboot attribute.

 manual      clears the autoboot attribute.

                        If the autoboot attribute is set to auto, the
                        virtual partition is rebooted after a successful
                        vparreset command or when the virtual partition
                        monitor is first loaded, if appropriate monitor
                        options have been specified.

                        If the attribute is set to manual, the virtual
                        partition halts after a vparreset, and does not
                        boot when the monitor is loaded.  It must then be
                        booted manually with the vparboot command.

                        If the -B option is omitted, the attribute
```

defaults to auto.

-D db_file Create the virtual partition in the virtual partition database contained in db_file. If db_file does not exist, it is created by this command. This option should be used when creating a new virtual partition database.

The virtual partition monitor need not be running. If it is, it is not notified of the new virtual partition.

Although db_file can reside in any path when accessed as an alternate database, it must be placed in the /stand directory before it can be loaded by the virtual partition monitor as its live database.

If the -D option is omitted and the virtual partition monitor is running, the virtual partition is created in the virtual partition monitor's live database. If the monitor is not running, the virtual partition is created in the default database file /stand/vpdb, which is itself created if it does not exist.

-S static_attr Specifies the static vPar attribute. static_attr can have the following (case-insensitive) values:

static sets the static attribute.

dynamic resets the static attribute.

virtual partitions set to static do not support any dynamic resource migration, nor can resource attributes be added/deleted/modified with vparmodify. This attribute becomes effective after any initial resource allocation, so it is possible to assign resources when creating a new virtual partition and still specify that the resulting virtual partition be static.

If the -S option is omitted, the attribute defaults to dynamic.

-b kernel_path Specifies the absolute path to a bootable kernel for the partition. For example, if a non-partitioned system start string at the ISL prompt is:

```
ISL> hpux -iS /stand/vmunix
```

the kernel_path is the /stand/vmunix portion of this string.If this option is omitted, the virtual partition is created with a default kernel path of /stand/vmunix.

-o boot_opts Specifies the command-line string, except for the kernel path, applied when the virtual partition is booted. For example, if a non-partitioned system start string at the ISL prompt is:

```
ISL> hpux -iS /stand/vmunix
```

the boot_opts string is the -iS portion of this string. Note that if the string includes any whitespace, the string must be quoted.

Refer to the virtual partitions administration guide and the chapter titled "Monitor and Shell Commands" for a full list of supported boot option strings.

The default is the empty string.

-a rsrc Adds resources to a virtual partition. rsrc is a hardware resource specification, as described in detail in the vparresources(5) manpage. A summary of resource syntax forms is shown in Table 1 below. Multiple -a rsrc specifications in the same command are allowed, but some syntax forms are only allowed once. This in indicated in Table 1. In all cases, resources in multiple specifications must not repeat or conflict with each other.

Table 1. Resource syntax summary

Resource	Forms	# times/command
CPU	cpu:path	Multiple
	cpu::num	Once
	cpu:::[min][:[max]]	Once
I/O	io:path[:attr1[,attr2]]	Multiple
Memory	mem::size	Once
	mem:::base:range	Multiple

Certain tasks can affect the outcome of others. To avoid errors, see the detailed description of the dependencies in the vparresources(5) manpage.

RETURN VALUE

The vparcreate command exits with one of the following values:

0 Successful completion.
1 One or more error conditions occurred.

EXAMPLES

Create the virtual partition called Oslo in the virtual partition monitor database, but do not assign any resources:

```
vparcreate -p Oslo
```

Create the virtual partition Bergen in the virtual partition monitor database, setting the autoboot attribute and specifying five processors, two of which are bound, 2 Gb of physical memory, and a boot disk:

```
vparcreate -p Bergen -a cpu::5 -a cpu:::2:6 \
   -a mem::2048 -a io:1/0/2/0.6.0:boot -B auto
```

Since the -b option was not specified, the default kernel path /stand/vmunix will be used at boot time.

ERRORS

vparcreate displays error messages on stderr for any of the following conditions:

An invalid option is specified.

An invalid value is specified for an option or a value is omitted.

vp_name already exists in the monitor database or specified db_file. Use the vparmodify command instead.

One or more options other than -a has been specified more than once or the same resource was allocated more than once.

An unavailable resource (allocated to another virtual partition or exceeding the available resource limit) was specified.

A value was omitted for an argument that requires one, or a value was supplied for an argument which does not take one.

AUTHOR

vparcreate was developed by the Hewlett-Packard Company.

SEE ALSO

vparboot(1M), vparmodify(1M), vparremove(1M), vparreset(1M), vparstatus(1M), vparresources(5), vpartition(5).

[Team LiB]

PREVIOUS NEXT

vparmodify



vparmodify - Modifies an existing virtual partition.

```
vparmodify(1M)                                              vparmodify(1M)


NAME
     vparmodify - modify the attributes of a virtual partition

SYNOPSIS
     vparmodify -p vp_name [-B boot_attr] [-D db_file] [-S static_attr]
  [-b kernel_path] [-o boot_opts] [-P new_vp_name] [-a rsrc]...
  [-m rsrc]... [-d rsrc]...

DESCRIPTION
     The vparmodify command modifies attributes and resources of the
     specified virtual partition.  The command can also rename the virtual
     partition.  vp_name is a symbolic name for an existing virtual
     partition and must be used in all references to it.  vparmodify cannot
     create a virtual partition.  Use the vparcreate command for that
     purpose.

     Only a superuser can execute the vparmodify command.

     If the -D option is specified, vp_name in db_file is modified;
     otherwise vp_name in the virtual partition monitor database is
     modified.

     If the static attribute has been configured in vp_name, none of its
     resources may be added, modified, or deleted.  Attributes (all options
     except -a, -m, and -d) may be modified.  Refer to the -S option
     description.

   Options
     Only the -S, -a, -m, and/or -d options may be specified more than once
     in a command.  Resources allocated with the -a option must be
     available, that is, not already allocated to a virtual partition.
     They must also not exceed the overall limits of the resources.
     Resources to be modified (-m) or deleted (-d) must be owned by
     vp_name.  Violation of any condition is a command error.

     vparmodify recognizes the following command line options and
     arguments.  The options are processed from left to right.

        -p vp_name      Specifies the symbolic name of the existing
                        virtual partition which is to be modified.
                        Required.

        -B boot_attr    Specifies the autoboot attribute of the virtual
                        partition.  boot_attr can have the following
                        case-insensitive values:

 auto      sets the autoboot attribute.

 manual      clears the autoboot attribute.

                        If the autoboot attribute is set to auto, the
                        virtual partition is rebooted after a successful
                        vparreset command or when the virtual partition
                        monitor is first loaded, if appropriate monitor
                        options have been specified.
```

If the attribute is set to manual, the virtual partition halts after a vparreset and does not boot when the monitor is loaded. It must then be booted manually with the vparboot command.

If the -B option is omitted, the attribute is not changed.

-D db_file Apply changes to the vp_name contained in db_file. The virtual partition monitor need not be running. If it is, it is not notified of the modification.

Although db_file can reside in any path when accessed as an alternate database, it must be placed in the /stand directory before it can be loaded by the virtual partition monitor as its live database.

If the -D option is omitted, the virtual partition monitor must be running. Changes are applied to vp_name in the monitor's database. A vp_name in the monitor database must be in the Down state, with one exception: Unbound cpu resources can be added to or deleted from a running virtual partition. This restriction may be relaxed for other types of resource management in a future release.

-S static_attr Specifies the static virtual partition attribute. static_attr can have the following values:

static sets the static attribute.

dynamic resets the static attribute.

If the -S option is omitted, the attribute is not changed.

No hardware resource changes can be made to a static virtual partition, that is, the -a, -m, and -d options are not allowed. This restriction applies to all virtual partitions, whether in an alternate database file, or in the monitor's database.

Since command line options are processed left-to-right, you can modify resources in a static partition by specifying -S dynamic on the command line before any resource options, then specifying -S static after all resource options to prevent further resource modifications.

-b kernel_path Specifies the absolute path to a bootable kernel for the partition. If the option is omitted, the kernel_path is not changed.

-o boot_opts Specifies the command-line string, except for the kernel path, applied when the virtual partition is booted. For example, if a non-partitioned system start string at the ISL prompt is:

```
ISL> hpux -iS /stand/vmunix
```

the boot_opts string is the -iS portion of this string. Note that if the string includes any whitespace, the string must be quoted.

Refer to the virtual partitions administration guide and the chapter titled "Monitor and Shell Commands" for a full list of supported boot option strings.

If the -o option is omitted, the boot_opts is not changed.

-P new_vp_name Specifies the new name for the virtual partition being modified, that is, after all modifications are successfully applied to vp_name, it is renamed to new_vp_name, assuming no virtual partition with that name already exists in the monitor database or specified alternate database file. The name can consist of alpha-numeric characters A-Z, a-z, 0-9, the underbar character (_), and period (.). The maximum length of the name is 239 characters.

a rsrc -a adds resources to a virtual partition.

-d rsrc -d deletes resources from a virtual partition.

-m rsrc -m modifies existing resources in a virtual partition. rsrc is a hardware resource specification, as described in detail in the vparresources(5) manpage. A summary of resource syntax forms is shown in Table 1 below. Multiple resource specifications in the same command are allowed, but some syntax forms are restricted to specific options, and some are only allowed once. Both are indicated in Table 1. In any case, resources in multiple specifications must not repeat or conflict with each other.

Table 1. Resource syntax summary

Resource	Forms	Options	# times/command
CPU	cpu:path	-a, -d	Multiple
	cpu::num	-a, -m, -d	Once
	cpu:::[min]:[max]	-m	Once
I/O	io:path[:attr1[,attr2]]	-a, -m, -d	Multiple
Memory	mem::size	-a, -m, -d	Once
	mem:::base:range	-a, -d	Multiple

Certain tasks can affect the outcome of others. To avoid errors, see the detailed description of the dependencies in the vparresources(5) manpage.

RETURN VALUE

The vparmodify command exits with one of the following values:

0 Successful completion.
1 One or more error conditions occurred.

EXAMPLES

Change the name of the virtual partition called Oslo to Bergen

```
vparmodify -p Oslo -P Bergen
```

Turn off the autoboot attribute for the virtual partition Oslo

```
vparmodify -p Oslo -B manual
```

Add two processors to the virtual partition Bergen in alternate database file /stand/Norway

```
vparmodify -p Bergen -D /stand/Norway -a cpu::2
```

ERRORS

vparmodify displays error messages on stderr for any of the following conditions:

An invalid option is specified.

An invalid value is specified for an option or a value is omitted.

The specified db_file does not exist, cannot be accessed, or has been corrupted.

vp_name does not exist in the specified db_file or in the monitor database.

new_vp_name already exists in the monitor database or specified db_file.

vp_name is static and you have attempted to change a resource.

One or more options other than -S, -a, -m, -d is specified more than once.

For the -a or -m option, an unavailable resource (allocated to another virtual partition or exceeding the available resource limit) is specified.

For the -m or -d option, the specified resource is not presently assigned to vp_name.

The state of the modified vPar following an error depends on the following factors:

+ If the error is detected during syntax and semantic checks of the command line, the vPar is not changed.

+ If the error is detected while actually modifying the vPar, AND you are modifying an alternate database, the vPar is not changed. This is because changes are not committed until the end of the command.

+ If the error is detected while modifying the vPar, AND you are modifying the database currently loaded into the monitor, then any changes made to the point of the error (working left-to-right on the command line) will remain. This is because changes to the live database are made incrementally, as they are requested.

AUTHOR

vparmodify was developed by the Hewlett-Packard Company.

SEE ALSO

vparboot(1M), vparcreate(1M), vparremove(1M), vparreset(1M), vparstatus(1M), vparresources(5), vpartition(5).

[Team LiB]

PREVIOUS NEXT

vparremove



vparremove - Removes (deletes) an existing virtual partition.

```
vparremove(1M)                                                vparremove(1M)


NAME
     vparremove - remove a virtual partition

SYNOPSIS
     vparremove -p vp_name [-D db_file] [-f]

DESCRIPTION
     The vparremove command deletes a virtual partition previously created
     using the vparcreate(1M) command.  All resources associated with the
     virtual partition are made available for allocation to other
     partitions.

     Only a superuser can execute the vparremove command.

     Unintentional use of this command has serious consequences; therefore
     the user is prompted to confirm the operation unless the -f (force)
     option is specified.

   Options
     vparremove recognizes the following command line options and
     arguments:

        -p vp_name      Specifies the unique name of the virtual partition
                        which is to be removed.  Required.


        -D db_file      Removes a virtual partition from the partition
                        database contained in db_file.  If this option is
                        omitted, the virtual partition is removed from the
                        monitor's database.  In this case, the partition
                        must be in the Down state to be removed.  Thus it
                        is an error to try to remove the current virtual
                        partition, the one in which the command is
                        executing.

        -f              Specifies the force option.  Omits the
                        confirmation dialog before removing the virtual
                        partition.  This option is intended for use by
                        scripts and other non-interactive applications.

RETURN VALUE
     The vparremove command exits with one of the following values:

     0   Successful completion.
     1   One or more error conditions occurred.

EXAMPLES
      Destroy the virtual partition Oslo in the partition database currently
      running in the monitor:

   vparremove -p Oslo
   Remove virtual partition Oslo? [n]: y

      Destroy the virtual partition Oslo in the partition database currently
      running in the monitor using the force option:
```

```
   vparremove -f -p Oslo

      Delete partition Bergen in partition database file Norway

   vparremove -f -D /stand/Norway -p Bergen

 ERRORS
      vparremove displays error messages on stderr for any of the following
      conditions:

+  db_file does not exist, cannot be accessed, is not a virtual
    partition database file, or is corrupt.

+  vp_name does not exist in the monitor's database or in
   db_file.

+  vp_name in the monitor's database is in some state other than Down.

 AUTHOR
      vparremove was developed by the Hewlett-Packard Company.

 SEE ALSO
      vparboot(1M), vparcreate(1M), vparmodify(1M), vparreset(1M),
      vparstatus(1M), vparresources(5), vpartition(5).
```

[Team LiB]

PREVIOUS NEXT

vparreset



vparreset - Sends a TOC or hard reset to a virtual partition.

```
vparreset(1M)                                                vparreset(1M)


NAME
     vparreset - reset a virtual partition

SYNOPSIS
     vparreset -p vp_name [-h|-t] [-q] [-f]

DESCRIPTION
     The vparreset command simulates, at the virtual partition level, the
     RS and TOC operations at a Control-B prompt on the system console.

     Only a superuser can execute the vparreset command.

     vparreset causes the virtual partition monitor to simulate a hard (RS)
     reset or soft (TOC) reset of the specified virtual partition.  Either
     operation gathers new Processor Information Module (PIM) data, which
     is displayed unless the -q (quiet) option is also specified.

     Unintentional use of the vparreset command has serious consequences;
     therefore the user is prompted to confirm the operation unless the -f
     (force) option is specified.

   Options
     vparreset recognizes the following command line options and arguments:

        -p vp_name      Specifies the unique name of the virtual partition
                        to be reset.The virtual partition must be in a
                        state other than Down or Crashing.  Required.


        -t              Simulates a TOC reset.  Displays current PIM data
                        before resetting unless the -q option is also
                        specified.

                        The TOC reset is also the default operation of the
                        command, applied if neither the -h nor -t option
                        is specified.

                        The -t and -h options cannot both be specified in
                        the same command.

        -h              Simulates a hard (RS) reset instead of a TOC.
                        Displays current PIM data before resetting unless
                        the -q option is also specified.

        -q              Bypasses the display of current PIM data when
                        resetting the virtual partition.

        -f              The force option.  Omits the confirmation dialog
                        before resetting the virtual partition.  This
                        option is intended for use by scripts and other
                        non-interactive applications.

RETURN VALUE

     The vparreset command exits with one of the following values:
```

```
     0   Successful completion.
     1   One or more error conditions occurred.

EXAMPLES

     Perform a TOC reset on the virtual partition called Oslo, dumping the
     PIM data:

  vparreset -p Oslo -t
  Reset virtual partition Oslo? [n] y


     Perform a hard reset on virtual partition Bergen skipping the
     confirmation dialog:


  vparreset -p Bergen -h -f


ERRORS
     vparreset displays error messages on stderr for any of the following
     conditions:

     + A required option is omitted.

     + An unknown option is specified.

     + A value is omitted for an argument that requires one, or a value is
specified for an argument which does not take one.

     +  vp_name does not exist in the monitor database, or is in the wrong
state.

     +  Both the -h and -t options are specified.

AUTHOR
     vparreset was developed by the Hewlett-Packard Company.

SEE ALSO
     vparboot(1M), vparcreate(1M), vparmodify(1M), vparremove(1M),
     vparstatus(1M), vparresources(5), vpartition(5).
```

[Team LiB]

PREVIOUS NEXT

vparresources



vparresources - Description of virtual partition resoures.

```
vparresources(5)                                          vparresources(5)


NAME
     vparresources - description of virtual partition resources and their
     requirements

DESCRIPTION
     Hardware resources are the most important property of a virtual
     partition (vPar).  These resources are divided into three major
     categories:

     + CPUs, or processors

     + Memory

     + I/O devices, such as disks, terminals, and printers.

     CPUs are further subdivided into bound and unbound processors.  A
     bound processor is interrupt-enabled and, at initial release, cannot
     be de-configured while the vPar is running.  An unbound, or floating,
     processor cannot process interrupts, but can be de-configured from a
     running vPar, and assigned to another running vPar.

     Each vPar can configure a subset of total system hardware resources
     such that a given physical resource is assigned to at most one vPar.
     This job is managed by two of the six virtual partition commands:

     +  vparcreate(1M), used when creating a new vPar. Resources can only
     be added.

     +  vparmodify(1M), used when modifying an existing vPar configuration.
     Resources can be added, modified, or deleted.

     Each command has specific resource syntax and semantic requirements.
     For example, some resource changes can only be made if the target vPar
     is not running.  Some syntax forms can be specified multiple times in
     one command.  Others can only be specified once. All of these
     conditions are described in the tables below.

     The general form of a resource specification is up to five positional
     fields delimited by colons (":"). No whitespace is allowed within any
     field.

     Table I  summarizes the three categories and all the allowable
     forms for each.

     Table II  specifies which forms are allowed for each of the three
     tasks (add, modify, or delete).

     Table III  is a detailed description of each syntax form and the
     conditions required for its use.

   Table I.  Resource syntax summary
     ____________________________________________________
     |Resource |   Form                  | # times/command |
     |_________|_________________________|_________________|
     |CPU      | cpu:path                | Multiple        |
```

	cpu::num	Once
	cpu:::[min][:[max]]	Once
I/O	io:path[:attr1[,attr2]]	Multiple
Memory	mem::size	Once
	mem:::base:range	Multiple

The first field is always one of the (case-insensitive) strings cpu, io, or mem.

The second field, when used, is a hardware path, for example, 10/12/6.

num, min, and max are all positive integers.

size and range are positive 64-bit integers in units of megabytes. base is an unsigned 64-bit integer in units of bytes. The commands round each of them upward as required to 64 megabyte boundaries. size, range, and base may each be specified in decimal or in hexadecimal. A hex specification should be preceded by 0x, as in 0x8000000.

The attributes for the I/O specification are zero, one, or both of the following (case-insensitive) strings: ALTBOOT and BOOT. If both are specified, separate them with a comma.

Each of the attributes can be assigned to no more than one I/O device. If it is already assigned to a device, a new assignment silently de-assigns it from its present device. However, one device can associate both attributes. This means it is possible for one device to own both the ALTBOOT and BOOT attributes, but it is not possible for two or more devices to own BOOT.

Users must guard against assigning an attribute to an inappropriate device, for example, assigning BOOT to a tty. The commands do not check for this, nor do they prevent it.

Table II. Allowed forms for each task

Task	Form	Allowed with vPar running
-a (add)	cpu:path	No
	cpu::num	Yes
	cpu:::[min][:[max]] (vparcreate only)	N/A
	io:path[:attr1[,attr2]]	No
	mem::size	No
	mem:::base:range	No
-m (modify)	cpu::num	Yes
	cpu:::[min][:[max]]	No
	io:path[:attr1[,attr2]]	No
	mem::size	No
-d (delete)	cpu:path	No
	cpu::num	Yes
	io:path[:attr1[,attr2]]	No
	mem::size	No
	mem:::base:range	No

The forms above are subject to the following semantic rules enforced by the commands.Note that according to Table II, except for the cpu::num form, a vPar must be in the Down state (or in an alternate database) to apply any of the changes described below. This requirement may be relaxed or removed in a future release.

+ CPUs

+ The total number of CPUs assigned to a vPar, specified by

cpu::num, must always be within the range specified by the cpu:::min:max specification. min of these are bound; (num-min) are unbound.

+ Any or all bound CPUs can be specified by explicit path, the cpu:path form. Any that are not so specified are allocated from the available pool and bound by the vPar monitor.

  A corollary to this is that you cannot specify more than the min CPUs by explicit path. If you wish to specify more, you must first increase the min, which in turn may require you to increase the max, num, or both.

  When you use -a cpu:path or -d cpu:path you do not change the total number of CPUs in your vPar. Instead, you replace a monitor-assigned bound CPU with one that you specify (-a) or replace a CPU that you specified with a monitor-assigned CPU (-d).

  Hewlett-Packard recommends that users configure specific CPUs only when required for performance reasons. In other situations, specify only the total number of CPUs (num) and how many of these should be bound (min), and allow the monitor to manage the actual CPUs allocated.

+ Since the monitor is not consulted when you are configuring a vPar in an alternate database (it need not even be running), you can specify any value for min (but <= max) in this case. When this database is loaded into the monitor, the monitor adjusts min downward as required if the specified number of CPUs is not available.

+ Defaults: When a vPar is created, the following defaults are in effect:

  + min: Platform dependent, usually 1

  + num: 1

  + max: If the vPar is created in an alternate database, 32767. If it is created in the live monitor database, max is equal to the total number of CPUs on the entire hardware system.

    You can modify these defaults with command line options to the vparcreate or vparmodify command.

+ Memory

  + The total amount of memory specified in explicit memory ranges, using the mem:::base:range specification, must be less than or equal to the total memory assigned to the vPar using the mem::size specification.

    When you specify a specific range of memory (whether adding or deleting), you do not change the total amount of memory assigned to the vPar. When adding, you merely specify that the particular range you specify be one part of the total amount assigned to your vPar. When deleting, the specific range is returned to the pool of unspecified total memory assigned to the vPar.

    Hewlett-Packard recommends that users configure specific memory ranges only when required for performance reasons. In other situations, specify only total memory and allow the monitor to manage the actual ranges allocated.

EXAMPLES

If more than one task is specified in a command, they are processed in the order (left-to-right) in which they are encountered on the command line. Some tasks will affect the outcome of others. Here is an example of correct usage, as well as counterexamples within the description.

```
vparcreate -p winona2 -a cpu::2 -a cpu:::2:4 -a cpu:41 -a cpu:45
```

-a mem::1280 -a mem:::0x40000000:128 -a io:0/8 -a io:1/10

-a 0/8/0/0.5.0:boot

At creation time, before any options are processed, min is equal to 1, as does num. Assume that the default max is sufficiently high, and that the specified resources are available for allocation.

+ -a cpu::2 succeeds because num(2) is within the range of the min and the max.

+ -a cpu:::2:4 then succeeds because num(2) is still within the required range. Note that if the two options were reversed the command would return an error due to left-to-right option processing, and the desired min would exceed the default num.

+ Finally, the specification of the two CPUs at explicit paths 41 and 45 succeeds because such assignments create bound CPUs. Since we have set min equal to 2, they can be accommodated. Note that without the first two -a options, the -a cpu:45 option would fail, because there would only be room for one (the default) bound CPU.

+ The allocation of 128 MB of specific memory at address 0x40000000 succeeds only because the total allocated memory was first set to 1280 MB. The 128 MB is taken from that 1280 MB; no new memory is added as a result of the -a mem::0x40000000:128 option.

The following vparmodify(1M) command adds two floating (unbound) CPUs to a vPar that has the Static attribute set. The Static attribute is then restored.

vparmodify -p winona3 -S dynamic -a cpu::2 -S static

Table III. Detailed resource specifications

Task: -a cpu:path

Explanation	Assigns a CPU resource at a specific path (bound CPU) to the vPar's configuration. The CPU replaces a monitor-assigned bound CPU. Bound CPUs are able to process interrupts.
Value	A text string of the form returned by the ioscan(1M) command, such as "10/12/6"
Usage Restrictions	The vPar must not be running. This restriction may be removed in a future release. The total number of CPUs at a specific path must be <= min In addition, if the vPar is in the monitor database: o A CPU must exist at path, o It must be available (not assigned to a vPar, including the target vPar).
Usage Guidelines	The addition does not increase either the number of total CPUs or bound CPUs allocated to the vPar. Instead, it replaces a previously bound CPU whose path had been selected by the monitor. To increase total CPU allocation, specify an appropriate new num. To increase the number of bound CPUs, specify a new min. Hewlett-Packard recommends that users configure specific CPUs only when required for performance reasons.In other situations, specify only the total number of CPUs (num) and how many of these should be bound (min), and allow the monitor to manage the actual CPUs allocated.

Task: -a cpu::num

Explanation	Adds num unbound CPUs to the vPar's configuration. The CPUs are drawn from a systemwide pool of available CPUs.
Value	A positive integer
Usage Restrictions	If the vPar is in an alternate database file or is in the live monitor database but not running: o Total number of CPUs <= max In addition, if the vPar is running: o num CPUs must exist on the system, o They must have been available (not bound to any vPar) at the time the vPar was booted. If another vPar is deleted with the vparremove command, its bound CPUs cannot be added as floaters to a running vPar.

Task: -a cpu:::[min][:[max]]

Explanation	Specifies the minimum and maximum number of CPUs allowed for the vPar. This operation does not allocate any CPUs, but specifies the limits of other allocation tasks. Both min and max are optional. The default min is platform-dependent but is currently = 1. The default max is 32767 if creating a vPar in an alternate database. If creating a vPar in the monitor database, the default max is the total number of CPUs on the entire hardware system.
Value	If specified, min and max must be positive integers such that min <= max.
Usage Restrictions	This option is allowed only in the vparcreate command. min cannot exceed the total number of CPUs. The default total when the vPar is created is 1. Use the -a cpu::num option to change the total before setting a min other than 1. max cannot be less than the total. min cannot be less than the total number of CPUs configured by explicit path.

Task: -a io:path[:attr1[,attr2]]

Explanation	Adds the I/O resource at path to the vPar. If attributes are specified, they are associated with the resource. If the vPar already owns the resource, any specified attributes are added to its configuration. This option only adds specified attributes. The state of unspecified attributes is not changed.
Value	path: A text string of the form returned by the ioscan(1M) command, such as "10/12/6" attr: One or both of the case-insensitive attribute strings ALTBOOT and BOOT. If both are specified, they are separated by a comma (",").
Usage Restrictions	The vPar must not be running.

	The I/O resource must either be unassigned or (when adding attributes to an already assigned resource) be assigned to the target vPar.
Usage Guidelines	At most one device can be assigned the ALTBOOT and BOOT attributes. Assigning one of these attributes to a device silently deletes it from its former device, if any. Caution: You should assign attributes to appropriate devices, but this is not checked. For example, ALTBOOT and BOOT should be assigned to a disk or tape. Failure to do this may result in an unbootable partition.

Task: -a mem::size

Explanation	Specifies the increase, in megabytes rounded upward to a 64 megabyte boundary, in the total amount of memory to be allocated to the vPar. This memory is taken from unspecified ranges of memory available to the system when the partition boots.
Value	A positive 64-bit integer <= 17592186044352
Usage Restrictions	The vPar must not be running. If the vPar is in the monitor database, this memory must physically exist and be available after the memory requirements of all other vPars have been satisfied. If the vPar is in an alternate database file, the assignment always succeeds. The amount of memory actually allocated if the database is loaded into the monitor may be less if some or all of it is needed in other vPars.

Task: -a mem:::base:range

Explanation	Specifies an explicit address space of memory starting at base bytes and extending for range megabytes.Both quantities are rounded upward as required to be aligned on 64 megabyte boundaries.
Values	Base: An unsigned 64-bit integer <= 0xfffffffffc000000 Range: A positive 64-bit integer <= 17592186044352
Usage Restrictions	The vPar must not be running. No part of the range may be already owned by this or another vPar. If the vPar is in the monitor database, the entire range must exist in the system. The total memory allocated in specific ranges must not exceed the vPar's memory size specification.
Usage Guidelines	Addition of specific memory ranges does not increase the total amount of memory allocated to the vPar. Any such memory is a part of that total amount. Caution:It is possible to specify memory ranges and sizes such that none of the vPars will launch. Hewlett-Packard recommends that users configure specific memory ranges only when required for performance reasons. In other situations, specify only total memory and allow the monitor to manage the actual ranges allocated.

Task: -m cpu::num

Explanation	Sets the total number of CPUs (bound and unbound) to num.
Value	A positive integer
Usage Restrictions	If the vPar is in an alternate database file or is in the live monitor database but not running: o num must be between min and max. In addition, if the vPar is in the live monitor database and num increases the total: o num CPUs must exist on the system, o They must have been available (not bound to any vPar) at the time the vPar was booted. If another vPar is deleted with the vparremove command, its bound CPUs cannot be added as floaters to a running vPar.

Task: -m cpu:::[min][:[max]]

Explanation	Specifies the minimum and maximum number of CPUs allowed for the vPar. This operation does not allocate any CPUs, but specifies the limits of other allocation tasks. You can change only min or max by not specifying the other field.
Value	If specified, min and max must be positive integers such that min <= max.
Usage Restrictions	The vPar must not be running. min cannot exceed the total number of CPUs. max cannot be less than the total. min cannot be less than the total number of CPUs configured by explicit path.

Task: -m io:path[:attr1[,attr2]]

Explanation	Changes the attributes of the resource to those specified in the option. Omitted attributes are removed from the attribute set. To retain an attribute, it must be specified.
Value	path: A text string of the form returned by the ioscan(1M) command, such as "10/12/6" attr: One or both of the case-insensitive attribute strings ALTBOOT and BOOT. If both are specified, they are separated by a comma (",").
Usage Restrictions	The vPar must not be running. The I/O resource must be assigned to the target vPar. Only attributes may be modified.
Usage Guidelines	At most one device can be assigned the ALTBOOT and BOOT attributes. Assigning one of these attributes to a device silently deletes it from its former device, if any. Caution:You should assign attributes to appropriate

	devices, but this is not checked. For example,
	ALTBOOT and BOOT should be assigned to a disk or tape. Failure to do this may result in an unbootable partition.

Task: -m mem::size

Explanation	Specifies the total amount of memory, in megabytes rounded upward to a 64 megabyte boundary, to be allocated to the vPar. Any memory in excess of that specifically allocated by base and range is taken from unspecified ranges of memory available to the system when the partition boots.
Value	A positive 64-bit integer <= 17592186044352
Usage Restrictions	The vPar must not be running. A decrease in total memory allocation must not result in a total less than that of all memory allocated in specific memory ranges. If the vPar is in the monitor database and the specification results in an increased memory allocation, the memory must physically exist and be available after the memory requirements of all other vPars have been satisfied. If the vPar is in an alternate database file and total memory is increased, the assignment always succeeds.The amount of memory actually allocated if the database is loaded into the monitor may be less if some or all of it is needed in other vPars.

Task: -d cpu:path

Explanation	De-assigns the bound CPU at the specified hardware path from the vPar and replaces it with a bound CPU of the monitor's choice. Depending on availability, this may be the same CPU that was de-assigned.
Value	A text string of the form returned by the ioscan(1M) command, such as "10/12/6"
Usage Restrictions	The vPar must not be running. The resource at the specified path must be a CPU resource. The vPar must own the resource.
Usage Guidelines	The deletion does not reduce either the number of total CPUs or bound CPUs allocated to the vPar. Instead, it replaces the explicitly specified CPU with a bound CPU selected by the monitor.To reduce total CPU allocation, specify an appropriate new num. To reduce the number of bound CPUs, specify a new min. Hewlett-Packard recommends that users configure specific CPUs only when required for performance reasons.In other situations, specify only the total number of CPUs (num) and how many of these should be bound (min), and allow the monitor to manage the actual CPUs allocated.

Task: -d cpu::num

Explanation	Deletes num CPUs from the vPar's total configuration.

Value	A positive integer
Usage Restrictions	The vPar must own at least num CPUs. The new total number of CPUs >= min

Task: -d io:path[:attr1[,attr2]]

Explanation	Removes the specified attributes of the resource leaving any previously assigned attributes unchanged and the resource itself assigned to the vPar. If no attribute is specified, removes the resource and all its attributes.
Value	path: A text string of the form returned by the ioscan(1M) command, such as "10/12/6" attr: One or both of the case-insensitive attribute strings ALTBOOT and BOOT. If both are specified, they are separated by a comma (",").
Usage Restrictions	The vPar must not be running. The I/O resource must be assigned to the target vPar.
Usage Guidelines	Deleting an attribute from an I/O device does not cause it to be assigned to another. You must do that in a separate option or command. At most one device can be assigned the ALTBOOT and BOOT attributes. Caution:You should assign attributes to appropriate devices, but this is not checked. For example, ALTBOOT and BOOT should be assigned to a disk or tape. Failure to do this may result in an unbootable partition.

Task: -d mem::size

Explanation	Specifies the decrease, in megabytes rounded upward to a 64 megabyte boundary, in the amount of total memory allocated to the vPar .
Value	A positive 64-bit integer <= 17592186044352
Usage Restrictions	The vPar must not be running. The vPar must own at least the specified amount (after rounding) of non-specific memory. The decrease must not result in a total less than that of all memory allocated in specific memory ranges.

Task: -d mem:::base:range

Explanation	De-assigns an explicit address space of memory starting at base bytes and extending for range megabytes. Both quantities are rounded upward as required to be aligned on 64 megabyte boundaries.
Values	Base: An unsigned 64-bit integer <= 0xfffffffffc000000 Range: A positive 64-bit integer <= 17592186044352

Usage Restrictions	The vPar must not be running. The vPar must own the entire range. Either the start or end point of the specified range must match the start or end point of an existing range.
Usage Guidelines	De-assigning specific memory ranges does not decrease the total amount of memory allocated to the vPar. Caution:It is possible to specify memory ranges and sizes such that none of the vPars will launch. Hewlett-Packard recommends that users configure specific memory ranges only when required for performance reasons. In other situations, specify only total memory and allow the monitor to manage the actual ranges allocated.

SEE ALSO

vparboot(1M), vparcreate(1M), vparmodify(1M), vparremove(1M), vparreset(1M), vparstatus(1M), vpartition(5).

[Team LiB]

vparstatus



vparstatus - Displays virtual partition and available resources information.

```
vparstatus(1M)                                                vparstatus(1M)


NAME
     vparstatus - display information about one or more virtual partitions

SYNOPSIS
     vparstatus [-v | -M] [-p vp_name]...  [-D db_file]

     vparstatus -A [-M]

     vparstatus –w

     vparstatus –e

     vparstatus -R [-p vp_name]

DESCRIPTION
     The various forms of the vparstatus command display:

     +  The attributes and hardware resources associated with one or more
virtual partitions (vPars) in either summary (the default) or
detailed format.  The -M option presents the same data in machine-
readable format.

     +  Resources currently available, that is, not assigned to any virtual
partition.  The -M option presents the same data in machine-
readable format.

     +  The name of the current virtual partition, that is, the virtual
partition from which the command is run.

     +  The virtual partition monitor's event log.

     +  Processor Information Module (PIM) data from the most recent
resetting of a virtual partition.

     Only a superuser can execute the vparstatus command.

     If no arguments are supplied, vparstatus displays a summary format of
     all attributes and resources of all virtual partitions in the monitor
     database.   One or more virtual partitions may be specified explicitly
     in order to restrict the output to information about the selected
     virtual partitions.  The -D option lists similar information from an
     alternate database file, but with a major difference.  See the
     description of the -D option for further details.

     There are three major listing formats.  The format chosen depends on
     command options and forms:

     + The summary format lists name, attributes, and resource totals.  It
is displayed if neither the -v nor -M option is used.

     + The detailed format lists name, attributes, and detailed resource
assignments, one per line with annotative headings.  The -v option
produces this format.

     + The machine readable format displays the same information as the
```

detailed format, except that field descriptive headers are omitted, and the information is all on one line. Individual fields are separated from each other by four delimiters.These are all described below.

Three additional formats are provided for the -w, -e and -R options.

Information displayed by vparstatus includes the following:

+ The name of the virtual partition (limited to 30 characters in summary format)

+ The state of the virtual partition, from the list below:

Up: The virtual partition has notified the monitor that it is up. This is the normal state of a running virtual partition, however it does not necessarily mean that the virtual partition has completed its initialization and is fully operational.

Down: The virtual partition is fully halted. This could be the result of a normal /etc/shutdown -h command, or a vparreset of a partition with its autoboot attribute set to manual. It is also the initial state of a virtual partition immediately after the virtual partition monitor is started.

Load: The monitor is loading the kernel image of the virtual partition. This state precedes the Boot state.

Boot: The virtual partition has been launched, but has not completely booted.

Crash: The virtual partition is shutting down ungracefully (either a panic or a reset)

Shut: The virtual partition is shutting down gracefully

Hung: The virtual partition has stopped sending heartbeat messages to the monitor.

N/A: The virtual partition is in an alternate database file, and so has no state.

+ The static/dynamic resource attribute

+ The auto/manual boot attribute

+ The path to the kernel (limited to 25 characters in summary format)

+ The boot options (limited to 5 characters in summary format)

+ Summary or detailed CPU, I/O, and memory resource allocations

Options

vparstatus recognizes the following command line options and arguments:

-p vp_name Restricts the command display to information about vp_name. By default, information about all virtual partitions in the monitor database or specified alternate partition database is displayed. Multiple vp_names may be specified, except when used to display PIM data.

-D db_file Displays information from the alternate partition database file db_file rather than from the monitor database.

The attributes and resources of a vPar in an alternate database file were specified in either the vparcreate or vparmodify command, but have not necessarily been checked for existence on a running system. They are referred to as "requested" information. Existence of the attributes and resources of a vPar in the live monitor database have been verified; they are referred to as "effective" information.

-v Presents a detailed display, one attribute or resource per line. Each attribute and resource type is identified.

CPU resources are identified by path when path information is available. Some CPU resources may not be known when displaying the status of alternate database vPars or vPars in the Down state. In this case, "<no path>" is displayed.

-A Displays information about available resources (that is, those not assigned to any virtual partition) in the virtual partition monitor's database. The monitor must be running.Resources are displayed one per line.

-M Displays information in a machine readable format. Individual fields are separated by one of four delimiters:

+ The colon (:) separates each field and resource type.For example, the state, attribute information, and kernel path would be displayed as:

Up:Static,Autoboot:/stand/vmunix

+ The semicolon (;) separates subfields of a resource type.For example, CPU resources are shown as:

5/10;33,37;51,53,55;<>,<>,<>

where the first subfield shows the minimum and maximum CPUs configured for the virtual partition; the second subfield lists the bound, or specified path, CPUs specifically configured by the user; the third subfield lists the remaining bound CPUs, which are assigned by the monitor; and the final subfield lists the unbound, or floating, CPUs.

In the example above, the paths of the unbound CPUs are not known. This could be the case because the example vPar is in an alternate database, or the vPar is in a Down state for which unbound CPUs are not assigned. In this case, "<>" is displayed in place of the path.

+ The comma (,) separates individual items in a list of similar items, such as the fixed path CPUs in the previous example.

+ The slash (/) separates all other related items, such as the CPU minimum and maximum in the same example.

Elements in the display are in the same order as the corresponding non-machine-readable display (-v or -A).

-w Displays the name of the current virtual partition (the one in which the command is executed). The monitor must be running.

-e Displays the monitor's event log, a circular file roughly 4K bytes long. Once the file is full, new entries overlay old ones to the nearest character. As a result, the first entry displayed may be missing some leading characters.The monitor must be running.

-R Displays Processor Information Module (PIM) data from the most recent reset of the specified virtual partition. If a virtual partition is not specified, PIM data from the current virtual partition, that is, the one in which the command is executed, is displayed.

If the virtual partition has not been reset, or if PIM data has been cleared since then, no PIM data is available so a message to that effect is displayed.

The monitor must be running.

RETURN VALUE

The vparstatus command exits with one of the following values:

0 Successful completion.
1 Syntax error or invalid option.

EXAMPLES

These examples assume the existence of an N-class hardware system, Europe, on which the virtual partition database Norway is currently loaded in the virtual partition monitor.Norway has two configured virtual partitions: Oslo, and Bergen. The vparstatus command is run from Oslo.

Europe has eight CPUs, two System Bus Adapters (SBAs), each with six Local Bus Adapters (LBAs), and 2 Gbytes of main memory. These resources are allocated among Oslo and Bergen as follows:

Oslo:

+ Two bound CPUs at hardware paths 33 and 37. The CPU at 33 was specified by the user with the -a cpu:path resource specification. The CPU at 37 was assigned by the monitor from the pool of CPUs available when Oslo was booted.
+ One floating (unbound) CPU at hardware path 41.
+ One SCSI boot disk at SBA/LBA/path 0/0/2/0.6.0.
+ One terminal at 0/0/4/0.
+ One LBA (0/4) to which are attached several LAN adapters.
+ Specific memory addresses 0x4000000 to 0xc000000 (Range = 64 MB) and 0x20000000 to 0x40000000 (Range = 128 MB). These specific ranges are part of a total memory allocation of 704 MB.

Bergen:

+ Two bound CPUs at hardware path 97 and 101, both assigned by the monitor.
+ Two unbound CPUs at hardware paths 93 and 99.
+ One SCSI boot disk at 1/4/0/0.5.0.
+ One LBA (1/10) to which are attached several LAN adapters
+ Specific memory 0x44000000 to 0x50000000 (Range = 192 MB) and 0x60000000 to 0x80000000 (Range = 512 MB).These specific ranges are part of a total memory allocation of 768 MB.

Display a summary format of all attributes and resources in all virtual partitions:

```
# vparstatus
[Virtual Partition]
  Boot
Virtual Partition NameState Attributes Kernel Path        Opts
============================== ===== ========== ======================== =====
OsloUp    Stat,Auto /stand/vmunix
BergenUp    Dyn,Manl /stand/vmunix

[Virtual Partition Resource Summary]
   CPU   Num      Memory (MB)
  CPU   Bound/   IO # Ranges/
Virtual Partition Name Min/Max  Unbound  devs Total MB    Total MB
============================== ================  ==== ====================
Oslo       2/  4    21     3    2/640    704
Bergen       2/  4    22     2    2/704    768
```

Display detailed attributes and resources of partition Oslo:

```
# vparstatus -p Oslo -v
[Virtual Partition Details]
Name:       Oslo
State:       Up
Attributes:   Static,Autoboot
Kernel Path:  /stand/vmunix
```

```
Boot Opts:

[CPU Details]
Min/Max:  2/4
Bound by User [Path]:33
Bound by Monitor [Path]:  37
Unbound [Path]:  41

[IO Details]
   0.0.2.0.6.0 BOOT
   0.0.4.0
   0.4

[Memory Details]
Specified [Base  /Range]:  0x4000000/64
  (bytes) (MB)    0x20000000/128
Total Memory (MB):  704
```

Display Bergen's CPU resources in machine-readable format:

```
# vparstatus -p Bergen -M
Bergen:Up:Dynamic,Manual:/stand/vmunix::2/4;;97,101;93,99:1.4.0.0.5.0 BOOT,1.10
:0x44000000/192,0x60000000/512;768
```

Display the name of the current virtual partition:

```
# vparstatus -w
The current virtual partition is Oslo.
```

AUTHOR

vparstatus was developed by the Hewlett-Packard Company.

SEE ALSO

vparboot(1M), vparcreate(1M), vparmodify(1M), vparremove(1M), vparreset(1M), vparresources(5), vpartition(5).

[Team LiB]

◄ PREVIOUS NEXT ►



vpartition

vpartition - Displays information about the vPars Command Line Interface.

```
vpartition(5)              vpartition(5)


NAME
     vpartition - display information about the Virtual Partition Command
     Line Interface

SYNOPSIS
     Vpartition

DESCRIPTION
     This manpage gives a listing and brief description of the commands
     which are used to manage virtual partitions.

        Command     |    Description
     _______________|_____________________________________________________
     vparboot       | Boot (start) a virtual partition.
                    |
     vparcreate     | Create a new virtual partition.
                    |
     vpardump       | Manage monitor dump files.
                    |
     vparextract    | Extract memory images from a running virtual partition
                    | system.
                    |
     vparmodify     | Modify an existing virtual partition.
                    |
     vparreloc      | Relocate the load address of a vmunix file, determine
                    | if a vmunix file is relocatable, or promote the scope of
                    | symbols in a relocatable vmunix file.
                    |
     vparremove     | Remove (delete) an existing virtual partition.
                    |
     vparreset      | Send a TOC or hard reset to a virtual partition.
                    |
     vparresources  | Description of virtual partition resourcs and
                    | their requirements.
                    |
     vparstatus     | Display virtual partition and available resources
                    | information.
                    |
     vparutil       | Get and set SCSI parameters for disk devices from
                    | a virtual partition.

AUTHOR
     vpartition was developed by the Hewlett-Packard Company.

SEE ALSO
     vparboot(1M), vparcreate(1M), vpardump(1M), vparextract(1M),
     vparmodify(1M), vparreloc(1M), vparremove(1M), vparreset(1M),
     vparstatus(1M), vparutil(1M), vparresources(5).
```

vparutil



vparutil - Works with SCSI-related information of disks in a vPar.

```
vparutil(1M)                                                    vparutil(1M)


NAME
     vparutil - get and set SCSI parameters for disk devices from a virtual
     partition

SYNOPSIS
     vparutil -g dev_path

     vparutil -s dev_path [id] [rate]

DESCRIPTION
     The vparutil command gets and sets SCSI paramaters for a specified
     SCSI device from a running virtual partition system.

     Only a superuser can execute the vparutil command.

   Options
     vparutil recognizes the following command-line options:

        -g    Gets the SCSI parameters for the specified SCSI disk device,
              dev_path.

        -s    Sets the SCSI parameters for the specified SCSI disk device,
              dev_path.

        vparutil recognizes the following command-line operands:

        dev_path            The device path of the SCSI disk device.

        Id                  The SCSI ID for the specified disk device.

        rate                The SCSI rate for the specified disk device.

EXAMPLES
     Set the SCSI ID for disk at 0/0/2/0.6.0 to 3:

  # vparutil -s 0/0/2/0.6.0 3

RETURN VALUE
  0 Successful.
  1 Could not communicate with virtual partition monitor.
  >1 Other error.

AUTHOR
    vparutil was developed by the Hewlett-Packard Company.
```

LICENSE AGREEMENT AND LIMITED WARRANTY

READ THE FOLLOWING TERMS AND CONDITIONS CAREFULLY BEFORE OPENING THIS SOFTWARE MEDIA PACKAGE. THIS LEGAL DOCUMENT IS AN AGREEMENT BETWEEN YOU AND PRENTICE-HALL, INC. (THE "COMPANY"). BY OPENING THIS SEALED SOFTWARE MEDIA PACKAGE, YOU ARE AGREEING TO BE BOUND BY THESE TERMS AND CONDITIONS. IF YOU DO NOT AGREE WITH THESE TERMS AND CONDITIONS, DO NOT OPEN THE SOFTWARE MEDIA PACKAGE. PROMPTLY RETURN THE UNOPENED SOFTWARE MEDIA PACKAGE AND ALL ACCOMPANYING ITEMS TO THE PLACE YOU OBTAINED THEM FOR A FULL REFUND OF ANY SUMS YOU HAVE PAID.

1. **GRANT OF LICENSE**: In consideration of your payment of the license fee, which is part of the price you paid for this product, and your agreement to abide by the terms and conditions of this Agreement, the Company grants to you a nonexclusive right to use and display the copy of the enclosed software program (hereinafter the "SOFTWARE") on a single computer (i.e., with a single CPU) at a single location so long as you comply with the terms of this Agreement. The Company reserves all rights not expressly granted to you under this Agreement.

2. **OWNERSHIP OF SOFTWARE**: You own only the magnetic or physical media (the enclosed software media) on which the SOFTWARE is recorded or fixed, but the Company retains all the rights, title, and ownership to the SOFTWARE recorded on the original software media copy(ies) and all subsequent copies of the SOFTWARE, regardless of the form or media on which the original or other copies may exist. This license is not a sale of the original SOFTWARE or any copy to you.

3. **COPY RESTRICTIONS**: This SOFTWARE and the accompanying printed materials and user manual (the "Documentation") are the subject of copyright. You may not copy the Documentation or the SOFTWARE, except that you may make a single copy of the SOFTWARE for backup or archival purposes only. You may be held legally responsible for any copying or copyright infringement which is caused or encouraged by your failure to abide by the terms of this restriction.

4. **USE RESTRICTIONS**: You may not network the SOFTWARE or otherwise use it on more than one computer or computer terminal at the same time. You may physically transfer the SOFTWARE from one computer to another provided that the SOFTWARE is used on only one computer at a time. You may not distribute copies of the SOFTWARE or Documentation to others. You may not reverse engineer, disassemble, decompile, modify, adapt, translate, or create derivative works based on the SOFTWARE or the Documentation without the prior written consent of the Company.

5. **TRANSFER RESTRICTIONS**: The enclosed SOFTWARE is licensed only to you and may not be transferred to any one else without the prior written consent of the Company. Any unauthorized transfer of the SOFTWARE shall result in the immediate termination of this Agreement.

6. **TERMINATION**: This license is effective until terminated. This license will terminate automatically without notice from the Company and become null and void if you fail to comply with any provisions or limitations of this license. Upon termination, you shall destroy the Documentation and all copies of the SOFTWARE. All provisions of this Agreement as to warranties, limitation of liability, remedies or damages, and our ownership rights shall survive termination.

7. **MISCELLANEOUS**: This Agreement shall be construed in accordance with the laws of the United States of America and the State of New York and shall benefit the Company, its affiliates, and assignees.

8. **LIMITED WARRANTY AND DISCLAIMER OF WARRANTY**: The Company warrants that the SOFTWARE, when properly used in accordance with the Documentation, will operate in substantial conformity with the description of the SOFTWARE set forth in the Documentation. The Company does not warrant that the SOFTWARE will meet your requirements or that the operation of the SOFTWARE will be uninterrupted or error-free. The Company warrants that the media on which the SOFTWARE is delivered shall be free from defects in materials and workmanship under normal use for a period of thirty (30) days from the date of your purchase. Your only remedy and the Company's only obligation under these limited warranties is, at the Company's option, return of the warranted item for a refund of any amounts paid by you or replacement of the item. Any replacement of SOFTWARE or media under the warranties shall not extend the original warranty period. The limited warranty set forth above shall not apply to any SOFTWARE which the Company determines in good faith has been subject to misuse, neglect, improper installation, repair,

alteration, or damage by you. EXCEPT FOR THE EXPRESSED WARRANTIES SET FORTH ABOVE, THE COMPANY DISCLAIMS ALL WARRANTIES, EXPRESS OR IMPLIED, INCLUDING WITHOUT LIMITATION, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. EXCEPT FOR THE EXPRESS WARRANTY SET FORTH ABOVE, THE COMPANY DOES NOT WARRANT, GUARANTEE, OR MAKE ANY REPRESENTATION REGARDING THE USE OR THE RESULTS OF THE USE OF THE SOFTWARE IN TERMS OF ITS CORRECTNESS, ACCURACY, RELIABILITY, CURRENTNESS, OR OTHERWISE.

IN NO EVENT, SHALL THE COMPANY OR ITS EMPLOYEES, AGENTS, SUPPLIERS, OR CONTRACTORS BE LIABLE FOR ANY INCIDENTAL, INDIRECT, SPECIAL, OR CONSEQUENTIAL DAMAGES ARISING OUT OF OR IN CONNECTION WITH THE LICENSE GRANTED UNDER THIS AGREEMENT, OR FOR LOSS OF USE, LOSS OF DATA, LOSS OF INCOME OR PROFIT, OR OTHER LOSSES, SUSTAINED AS A RESULT OF INJURY TO ANY PERSON, OR LOSS OF OR DAMAGE TO PROPERTY, OR CLAIMS OF THIRD PARTIES, EVEN IF THE COMPANY OR AN AUTHORIZED REPRESENTATIVE OF THE COMPANY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES. IN NO EVENT SHALL LIABILITY OF THE COMPANY FOR DAMAGES WITH RESPECT TO THE SOFTWARE EXCEED THE AMOUNTS ACTUALLY PAID BY YOU, IF ANY, FOR THE SOFTWARE.

SOME JURISDICTIONS DO NOT ALLOW THE LIMITATION OF IMPLIED WARRANTIES OR LIABILITY FOR INCIDENTAL, INDIRECT, SPECIAL, OR CONSEQUENTIAL DAMAGES, SO THE ABOVE LIMITATIONS MAY NOT ALWAYS APPLY. THE WARRANTIES IN THIS AGREEMENT GIVE YOU SPECIFIC LEGAL RIGHTS AND YOU MAY ALSO HAVE OTHER RIGHTS WHICH VARY IN ACCORDANCE WITH LOCAL LAW.

[Team LiB]

ACKNOWLEDGMENT

YOU ACKNOWLEDGE THAT YOU HAVE READ THIS AGREEMENT, UNDERSTAND IT, AND AGREE TO BE BOUND BY ITS TERMS AND CONDITIONS. YOU ALSO AGREE THAT THIS AGREEMENT IS THE COMPLETE AND EXCLUSIVE STATEMENT OF THE AGREEMENT BETWEEN YOU AND THE COMPANY AND SUPERSEDES ALL PROPOSALS OR PRIOR AGREEMENTS, ORAL, OR WRITTEN, AND ANY OTHER COMMUNICATIONS BETWEEN YOU AND THE COMPANY OR ANY REPRESENTATIVE OF THE COMPANY RELATING TO THE SUBJECT MATTER OF THIS AGREEMENT.

Should you have any questions concerning this Agreement or if you wish to contact the Company for any reason, please contact in writing at the address below.

Robin Short

Prentice Hall PTR

One Lake Street

Upper Saddle River, New Jersey 07458

About the CD-ROM

The CD-ROM included with this book contains the following directories:

Partitions

A white paper describing HP's virtual partitions.

ITRC

An overview document describing HP's IT Resource Center.

TechTips

Cheatsheets for various topics that are used by the HP Response Center.

Firmware

Hardware and firmware commands for N-Class.

Performance

Extensive trial performance software for HP-UX.

Additional file

About_CD.pdf - Displays the directory hierarchy on the CD-ROM.

The first four directories contain useful files for viewing. The Performance directory contains trial performance software. The procedure for loading, using, and obtaining a permanent license for this software appears in the Performance directory.

Many of the files on the CD-ROM require Acrobat Reader. These are fully formatted files such as manuals and marketing material. You can obtain Acrobat Reader by going to www.Adobe.com and click on the Get Acrobat Reader icon. Follow the instructions for your platform.



Technical Support

Prentice Hall does not offer technical support for this software. However, if there is a problem with the media, you may obtain a replacement copy by e-mailing us with your problem at:
disc_exchange@prenhall.com



[Team LiB]

[Team LiB]

[**SYMBOL**] [A] [B] [C] [D] [E] [F] [G] [H] [I] [J] [K] [L] [M] [N] [O] [P] [R] [S] [T] [U] [V] [W] [X] [Y]

$HOME/.dt/types/dtwm.fp 2nd
$HOME/.dtprofile
.bashrc 2nd
.cshrc
.dt
.dtprofile
.login
.profile
.Xdefaults
/.profile
/.rhosts 2nd 3rd
/.sh_history
/boot (Linux)
/dev 2nd
/etc
/etc/cmcluster
/etc/cmcluster/cmnodelist
/etc/cmcluster/conf.ascii
/etc/dt/appconfig/types/C/dtwm.fp
/etc/dt/config/C
/etc/dt/config/C/Xconfig
/etc/dt/config/C/Xresrouces
/etc/dt/config/C/Xsession
/etc/exports 2nd 3rd 4th 5th 6th 7th
/etc/fstab 2nd 3rd
/etc/group 2nd 3rd 4th 5th 6th
  NIS
/etc/hosts 2nd 3rd 4th 5th 6th
/etc/hosts.equiv 2nd
/etc/init.d 2nd
/etc/inittab 2nd 3rd 4th 5th 6th 7th 8th 9th
  action
  id
  process
  run state
/etc/lvmtab
/etc/named.boot
/etc/named.conf
/etc/nsswitch.compat
/etc/nsswitch.conf 2nd
/etc/opt/gnome/gdm/gdm.conf
/etc/opt/samba/smb.conf
/etc/passwd 2nd 3rd 4th 5th 6th 7th 8th 9th
  NIS
/etc/passwd.nis
/etc/passwd\
  description of fields

/etc/prmconf 2nd 3rd 4th 5th 6th 7th
/etc/prmconf.dev
/etc/prmconf.test 2nd
/etc/profile
/etc/rc.confg.d/netconf
/etc/rc.config.d 2nd 3rd 4th
/etc/rc.config.d/bv
/etc/rc.config.d/ndd
/etc/rc.config.d/netconf 2nd
/etc/rc.config.d/nfsconf
/etc/rc.config.d/prm
/etc/rc1.d
/etc/resolv.conf 2nd
/etc/samba/smbpasswd 2nd
/etc/smb.conf
/etc/xinetd.d/swat
/export
/home 2nd
/lost+found
/mnt
/net
/opt
/opt/HPvmsa/bin/vmsa
/opt/ignite/bin
/opt/samba/bin
/proc (Linux)
/sbin
/sbin/init 2nd
/sbin/init.d
/sbin/init.d/bv
/sbin/init.d/net
/sbin/init.d/vparhb
/sbin/mksf
/sbin/rc 2nd 3rd 4th 5th
/sbin/rc0.d 2nd
/sbin/rc1.d 2nd 3rd 4th 5th 6th
/sbin/rc2.d 2nd 3rd
/sbin/rc3.d 2nd 3rd
/sbin/service 2nd 3rd
/sbin/set_parms
/sbin/shutdown
/sbin/vecheck 2nd
/stand 2nd 3rd
/stand and /kernel
/stand/build 2nd
/stand/system 2nd 3rd
/stand/vmpon.dmp
/stand/vmunix 2nd 3rd
/stand/vpdb 2nd 3rd 4th
/stand/vpmon 2nd 3rd 4th 5th
/tmp

/usr
/usr/conf/gen/config.sys
/usr/dt/appconfig/icons 2nd
/usr/dt/appconfig/types/C/dtwm.fp
/usr/dt/bin/dtlogin
/usr/dt/config/C/sys.dtwmrc
/usr/dt/config/C/Xconfig
/usr/dt/config/C/Xresources 2nd
/usr/dt/config/C/Xsession
/usr/sbin/route
/usr/sbin/setup
/var
/var (and Ignite-UX)
/var/adm/crash/vpar 2nd 3rd 4th
/var/opt/ignite/clients
/var/opt/ignite/depots 2nd
/var/spool/cron/crontabs
11.11 2nd
32-bit
64-bit

[Team LiB]

[Team LiB]

Absolute and Relative Path Names
Access rights of files
action (/etc/inittab)
Adding a Peripheral
Advanced Installation 2nd 3rd
alias 2nd
Aliases
Application Startup
Application Users
arp
Assignment Operators
AUTO
awk 2nd 3rd 4th 5th 6th 7th 8th
man page

[Team LiB]

[Team LiB]

Background Jobs
Backup
- cpio
- dd
- dump
- fbackup
- frecover
- of vPars
- tar
- vxdump
- vxrestore

Base and range (memory on NUMA)
Bash
- .bashrc
- /etc/profile
- command line editing
- history list
- Recalling from the History List table
- shell quoting

BCH 2nd 3rd 4th
BCHFastBoot
bdf 2nd 3rd 4th
bg
BIND
bo p0
boot
- man page 2nd

boot boot state of vPars
Boot Console Handler 2nd
Boot Control Handler
Boot disk
boot lan
boot state
Boot States
bootable system recovery tape
bootwait
Bottlenecks
- performance

bound 2nd 3rd 4th 5th 6th
Bound
Broadvision 2nd

[Team LiB]

[Team LiB]

[SYMBOL] [A] [B] [**C**] [D] [E] [F] [G] [H] [I] [J] [K] [L] [M] [N] [O] [P] [R] [S] [T] [U] [V] [W] [X] [Y]

C Shell
- .cshrc
- alias
- Command-Line History table
- history
- Recalling from History List
- shell quoting

C Shell Programming
- Assignment Operators
- Command Substitution
- csh -xv
- Debugging
- Reading User Input
- switch
- Testing and Branching

C Shell\
- .login

case 2nd
cat 2nd 3rd
cbuf command at MON>
cd 2nd 3rd 4th
CDE 2nd 3rd 4th 5th
- Activity Light
- Adding Objects to or Removing Objects from the Front Panel
- Advanced Topics
- Application Manager
- Backdrop and Colors
- background
- Calendar
- CDE Basics
- CDE on Solaris
- Changing the Front Panel in Other Ways
- Clock
- Creating Control Buttons, Actions, and File Types
- Customizing
- dtterm
- Exit button
- File Manager
- Front Panel Animation
- GREETING
- Help
- Lock Button
- Mail
- Menu Picks for Virtual Partitions
- Modifying Things in Slide-up Subpanels
- on AIX
- on Solaris

Performance
Personal Printers
Sequence of Events When CDE Starts
Style Manager
Trash Can
Using Different Fonts 2nd
vPar Name
Workspace Area
Workspace Menu
Workspace Switch
xterm 2nd
CDE Configuration File Locations
CDE Configuration Files
change directory
chmod 2nd 3rd 4th 5th 6th 7th 8th 9th 10th 11th
CIFS/9000
cmapplyconf 2nd 3rd 4th
cmcheckconf 2nd 3rd 4th
cmhaltpkg
cmmakepkg 2nd
cmmodpkg 2nd
cmp 2nd 3rd
cmquerycl 2nd
cmruncl 2nd
cmrunpkg
cmviewcl 2nd 3rd 4th
comm
Command completion
Command completion (ksh)
Command completion (ksh)\
table summarizing
Commands of vPars (table) 2nd
Common Desktop Environment ([See CDE)]2nd [See CDE)]
Comparison operators
conf.cacheonly
Configurable Parameters
Console
console 2nd
Consoles
types
Core I/O
cp 2nd 3rd 4th 5th
-f
-i 2nd
-p
-r
-R
cpio 2nd 3rd 4th 5th 6th 7th 8th
of vPars
CPU
CPU bottleneck

CPU resources
CPU utilization
crash boot state of vPars
Create vPars
Creating a Kernel in HP-UX 11i
crontab
man page
Ctrl-A 2nd 3rd 4th
Ctrl-A (and console)
cut 2nd 3rd 4th 5th 6th

[Team LiB]

[Team LiB]

Data Files
Database
date
db.cache
dd 2nd 3rd
Debugging
Device Files 2nd 3rd 4th
diff 2nd
dircmp 2nd
Disk bottleneck
Disk utilization
diskinfo
dmesg 2nd 3rd
DNS
do
done
down boot state of vPars
dt.resources
dtmail
DtRootMenu
dtterm
dtwm.fp 2nd 3rd
dtwmrc 2nd
du
dump 2nd
Dump files

[Team LiB]

[Team LiB]

echo 2nd
EFI 2nd 3rd 4th
  Boot Admin Main Screen
  Main Screen
egrep
esac
Ethernet
Executable Files
Expressions (regular expressions)
Extensible Firmware Interface 2nd 3rd
Extensions (file types)

[Team LiB]

[Team LiB]

Fair Share Scheduler (FSS)
FastBoot
fbackup 2nd 3rd 4th 5th 6th 7th
  -f option
  -g option
  -I option
  -O option
  -u option
  command summary
  graph file
  graph file exclude
  graph file include
  of vPars 2nd 3rd 4th
  of vPars to remote device 2nd 3rd 4th 5th
fg
fgrep
fi
file Command
File Name Expansion and Pattern Matching
File Name Expansion and Wild Cards
File permissions
File Transfer Protocol
File Types
find 2nd 3rd 4th 5th 6th 7th 8th 9th 10th 11th 12th 13th 14th
Find
  empty files
find
  man page
  name
  old files
  owner
  permissions
  size
  summary
  type
find,
firewall(in Samba)
firmware
for 2nd
foreach
frecover 2nd
frestore
  of vPars
fsadm 2nd
fsck 2nd
FSS 2nd 3rd
FSS Fair Share Scheduler

ftp

[Team LiB]

[Team LiB]

gdmconfig
get auto command at MON>
glance
GlancePlus/UX 2nd
GNOME 2nd
  Control Center
  DESKTOP=GNOME
  downloading
  login screen
Gnome (on Linux)
GNU Network Object Model Environment
gpm 2nd
grep 2nd 3rd 4th 5th 6th 7th 8th 9th 10th 11th 12th 13th 14th 15th
  man page
GSP
  configuring
  table of commands
gtop
Guardian Service Processor
Guided Installation

[Team LiB]

[Team LiB]

Hardware Inventory
head 2nd 3rd 4th 5th
heartbeat
daemon of vPars
heartbeat (vPars heatbeat daemon)
heartbeat deamon of vPars
help command at MON>
HISTFILE
history 2nd 3rd 4th 5th 6th 7th 8th 9th 10th 11th
History (C Shell)
history (in ksh)
History List 2nd
hostname 2nd
hosts_to_named 2nd
HP-UX 11i
HP-UX 11i kernel creation
HP-UX Release 11.11
hpux
HPUX
hpux
examples
man page 2nd
hpux command 2nd
http
//www.hp.com/workstations/support/software/hpux/gnome/index.html
hung boot state of vPars

[Team LiB]

[Team LiB]

I/O interrupts
I/O Redirection
ICMP source quench
iCOD 2nd
id (/etc/inittab)
if 2nd
ifconfig 2nd 3rd 4th 5th
  man page 2nd
ignite
  man page 2nd
Ignite-UX 2nd 3rd
  /var
  archive information
  bdf
  boot
  depots
  disk space consumed by depots
  Ignite-UX Administrator Guide
  make_depots
  make_net_recovery
  make_tape_recovery 2nd 3rd
  server setup
Ignite-UX GUI
Ignite-UX GUI to Create a vPar Archive
Ingite-UX
  GUI
init
initdefault
Initial System Loader
inittab
  man page 2nd
Install HP-UX
Internet Protocol (IP) Addressing
ioscan 2nd 3rd 4th 5th 6th 7th 8th 9th 10th 11th 12th 13th 14th 15th 16th
  networking
iostat 2nd 3rd 4th
  man page 2nd
IP Addressing
IP classes
ip_send_source_quench 2nd
IPF
IPL
ISL 2nd 3rd 4th 5th
isl
  man page
ISL
  ODE

ISO/OSI
IT Resource Center
Itanium processor family 2nd

[Team LiB]

[Team LiB]

JFS
  fsadm
  fsck
Job Control
jobs
join

[Team LiB]

[Team LiB]

kcweb 2nd 3rd 4th
Kernel 2nd
- /stand/build
- /stand/vmunix
- Building a Kernel
- building a new kernel in HP-UX 11i
- Building with kcweb web-based tool
- ioscan
- kcweb
- kmtune 2nd 3rd
- kmupdate
- sysdef
- system_prep

kill 2nd
- signals

kill (K)
kmsystem 2nd
kmtune 2nd 3rd
kmupdate 2nd 3rd
KornShell
ksh
- .kshrc
- .profile
- .sh_history 2nd
- /etc/profile
- aliases
- command and path completion
- command line editing
- Environment Variables
- Fetching Commands
- File Name Expansion
- file name expansion
- history
- history file
- ksh-Related Variable Commands
- PS1
- PS2
- re-executing
- startup files
- unalias

ksh Programming
- Arguments to Shell Programs
- awk
- case Statement
- if
- looping
- Reading User Input

Shell Functions
Shell Variables
test
Testing and Branching
while

[Team LiB]

[Team LiB]

[SYMBOL] [A] [B] [C] [D] [E] [F] [G] [H] [I] [J] [K] [L] [M] [N] [O] [P] [R] [S] [T] [U] [V] [W] [X] [Y]

L-Class with two vPars
LAN 2nd
LAN Cards
LAN cards
LAN Console
LAN console
lanadmin 2nd
LBA 2nd 3rd 4th 5th 6th
lifls
link layer
Links
Linux
- top 2nd
- xosview 2nd

Linux File System Layout
Listing Software
load
load boot state of vPars
load state
Local Bus Adapter
Local Bus Adapters
Local Console
log command at MON>
Logical Extents (LVM)
Logical Volume
Looping
ls 2nd 3rd 4th 5th 6th 7th 8th 9th 10th
- -a
- -i
- -l
- -p
- -R
- summary

lsdev 2nd 3rd 4th 5th 6th
lvchange
lvcreate 2nd 3rd 4th 5th
lvdisplay 2nd 3rd
lvextend 2nd 3rd
lvlnboot 2nd 3rd 4th 5th
LVM
- lifls
- logical extents
- logical volume
- lvchange
- lvcreate
- lvdsiplay
- lvextend

lvlnboot
lvmerge
lvmmigrate
lvreduce
lvremove
lvsync
Mirror root and swap disk
mknod
newfs
physical extent
Physical Volume Links
procedures
PV Links
pvchange
pvcreate
pvdisplay
pvmove
setboot
strings /etc/lvmtab
vgcfgbackup
vgcfgrestore
vgchange
vgcreate
vgdisplay
vgexport
vgextend
vgimport
vgreduce
vgremove
vgscan 2nd
volume
volume group
LVM)
lvmerge
lvmmigrate
lvreduce 2nd
lvremove
lvrmboot 2nd
lvsplit
lvsync

[Team LiB]

[Team LiB]

make _net_recovery
  of vPars
make_net_recovery 2nd 3rd 4th 5th 6th 7th 8th 9th
make_recovery
  man page
make_sys_image
make_tape_recovery 2nd 3rd 4th
maxuprc
media (operating system)
mediainit
Memory
Memory bottleneck
Memory controller
Memory range (on NUMA)
Memory resources
Memory utilization
Meta-Characters
Middleware
MirrorDisk/UX
Mission Critical Operating Environment 2nd 3rd 4th
mk_kernel 2nd
mkboot
mkdir 2nd 3rd 4th 5th 6th 7th 8th 9th
mknod 2nd 3rd 4th 5th
mksf
Modify a vPar
MON
MON> 2nd 3rd
more 2nd 3rd 4th
Motif
  /usr/lib/X11/app-defaults
  system.mwmrc
Motif Configuration Files
mount 2nd 3rd
mv 2nd 3rd 4th
  -f
  -i
  -p
mxsetup

[Team LiB]

[Team LiB]

Name (using find)
named.conf
ndd
  -h
  ip_send_source_quench
net view 2nd 3rd
NetBIOS
netconf 2nd 3rd
netstat 2nd 3rd 4th 5th 6th
  -a
netstat
  -i
netstat
  -I 2nd
  -i
  -n
  -rn
  man page 2nd
Network Layer
Networking
  Ethernet Encapsulation
  ISO/OSI layers
  Link Layer
  lsdev
  network layer
  Physical Layer
  subnet mask
  trasport level
Networking bottleneck
newfs 2nd 3rd
newgrp
NFS
nfsconf
nice
NIS 2nd 3rd
nmbd 2nd
nPartition
  PRM on Superdome example
nPartitions
  ioscan
  rad
nslookup 2nd
nsquery
  group users
  hosts 192.16.16.204
  hosts www.hp.com
  passwd hackley

[Team LiB]

[Team LiB]

ODE
off (/etc/inittab)
once (/etc/inittab)
ondemand
OnlineJFS 2nd
  fsadm
  fsck
OpenView Performance Agent 2nd
OpenView Performance Analyzer/UX 2nd
Operating Environment 2nd 3rd 4th 5th 6th 7th 8th 9th 10th
Operating Environments 2nd 3rd
  Basic HP-UX OE
  Enterprise HP-UX OE
  Minimal Technical OE
  Mission Critical HP-UX OE
  Technical Computing OE
Oracle 2nd
Oracle and PRM (Process Resource Manager)
Owner (using find)

[Team LiB]

[Team LiB]

passwd -r nis
password
Password
  features
paste 2nd 3rd 4th
Patches
Path completion
Path completion (ksh)
Path completion (ksh)\
  table summarizing
Pattern Matching
Pattern matching
  in vi
PCI I/O
PDC 2nd 3rd 4th 5th 6th
pdc
  man page
PDC
  table of commands
Performance
  identifying bottlenecks
  iostat 2nd 3rd
  netstat 2nd
  ps
  vmstat 2nd
perfstat
Peripherals
Permissions 2nd 3rd 4th
  execute 2nd
  read
permissions
  read
Permissions
  write 2nd
Permissions (using find)
pg 2nd
Physical Extent
physical layer
Physical Volume (LVM)
ping 2nd 3rd
POSIX shell
powerfail
powerwait
present working directory
PRM
  /etc/passwd
  GlancePlus 2nd

gpm
groups
High Share
Low Share
Oracle Setup
prmconf
Resource Caps
Software installed
Table of vPars commands
time
prmagt
prmanalyze 2nd
prmavail 2nd 3rd 4th 5th
prmconfig 2nd 3rd 4th
prmlist 2nd 3rd 4th 5th
prmloadconf 2nd
prmmonitor 2nd 3rd 4th
prmmove
prmrecover
prmrun
process (/etc/inittab)
Process Resource Manager (PRM)
processes
Processor Dependent Code 2nd
Processor hardware paths
Processor Set (PSET)
Processor Sets
Prompt
PS1
ps 2nd 3rd
-ef
man page
ps -ef 2nd 3rd
PS1 2nd
PS2
PSET
PSET-Processor Set
psets
PV Links (LVM)
pvchange
pvcreate 2nd 3rd 4th 5th 6th
pvdisplay
pvmove
pwd 2nd 3rd 4th

[Team LiB]

[Team LiB]

r 2nd
r (re-executing in ksh)
rad
RAM
Range (memory) on NUMA
rc1.d 2nd
rc2.d 2nd
rcp 2nd
Re-Executing Commands
read 2nd
reboot
Red Hat Linux
Redirection
Redirection (I/O Redirection)
Redirection Forms (table)
Relative and absolute path names
Remote Console
remsh 2nd 3rd
  with cpio and vPars
respawn
respawn (/etc/inittab)
restore
rlogin
rm 2nd 3rd 4th 5th 6th
  -f
  -i
  -r (and -R)
rmdir 2nd 3rd
  -f
  -i
  -p
root
root directory
Root Disk
Root Swap
route 2nd 3rd 4th
rp8400 2nd
  nPartitions
rpcinfo
Run levels
run state (/etc/inittab)

[Team LiB]

[SYMBOL] [A] [B] [C] [D] [E] [F] [G] [H] [I] [J] [K] [L] [M] [N] [O] [P] [R] [**S**] [T] [U] [V] [W] [X] [Y]

SAM
- Accounts for Users and Groups
- Auditing and Security
- Backup and Recovery
- Disks and File Systems
- Kernel Configuration
- Networking and Communications
- NFS Diskless Concepts
- Partition Manager
- Performance Monitors
- Peripheral Devices
- Printers and Plotters
- Process Management
- Routine Tasks
- Run SAM on Remote Systems
- Software Management
- Time

Samba
- Mapping a Windows Drive to a Linux Directory
- mbd
- Mounting a Windows Drive on a Linux System
- Samba Web Configuration Tool
- setup
- SMB
- smb
- smb.conf
- smbclient
- smbmount
- SWAT
- TCP/IP

samba_setup
sar 2nd 3rd 4th 5th
SBA
SBA (System Bus Adapter)
scan command at MON> 2nd
SCM 2nd
SEArch
Secondary System Load
Secondary System Loader
Secure Web Console 2nd 3rd
sed 2nd 3rd 4th 5th 6th
- man page

Server Message Block
ServiceControl Manager
ServiceGuard
- Cluster 2nd
- Package

set_parms 2nd 3rd 4th 5th 6th 7th 8th 9th 10th
initial
setboot 2nd 3rd 4th 5th 6th 7th 8th 9th 10th 11th
setboot Table
Shared Volume Group
Shell Programming
Shell Programs
Shell Quoting 2nd
showmount 2nd
shut boot state of vPars
shutdown 2nd 3rd
-h
-h -y
-r
/sbin/rc
/sbin/reboot
init
man page 2nd
Updates all superblocks
Shutdown scripts
Shutdown Scripts
Shutdown scripts
Execution scripts
K
SIGHUP
SIGINT
SIGKILL 2nd
Signals
SIGQUIT
SIGSTOP
SIGTERM
Size (memory)
Size (using find)
SMB 2nd
smb.conf 2nd
SMB/CIFS
smbclient 2nd 3rd
smbd 2nd
smbmount 2nd
smbpasswd 2nd 3rd
smbrun
smbstatus
smbtar
smbumount
Software Distributor 2nd
bundle
patches
product
software.hp.com
sort 2nd 3rd 4th 5th
Source Code Files

split 2nd
start
  as part of startup
start (S)
startsmb
Startup of application
Startup of vPars
Startup script
Startup Scripts
Startup scripts
  Execution Scripts
  run levels
  S
  start
  start_msg
  stop
  stop_msg
States of vPars
States of vPars table
Steps to create virtual partitions
Steps to create vPars
stop
  as part of startup
strings /etc/lvmtab 2nd 3rd
Subnet Mask
summary.1
Supercome
  parstatus
Superdome
  PRM
  rad
sw command summaries
swap
swapinfo
SWAT 2nd
swcopy
swinstall 2nd 3rd 4th
switch
swjob
swlist 2nd 3rd 4th 5th 6th 7th
  attributes
  options
swremove
Synchronization of vpdb
sys.dtwmrc
sys.resources 2nd
sys.x11start
sys.Xdefaults
sysdef
sysinit
System Bus Adapter

System bus adapter
system.mwmrc
system_prep

[Team LiB]

[Team LiB]

Table of vPars commands
tail 2nd 3rd
tar 2nd 3rd 4th 5th
telnet
test
test Command Conditions
testparm 2nd 3rd
Text Files
  ASCII
time 2nd 3rd
timex
top 2nd 3rd
  man page 2nd
touch
tr 2nd
Transport Layer
Type (using find)

[Team LiB]

[Team LiB]

umask 2nd 3rd 4th 5th 6th
umount 2nd
unalias
uname 2nd 3rd
unbound 2nd 3rd 4th 5th 6th 7th
UNIX File System Layout
UNIX file types
UNIX files and directories
up boot state of vPars
up state
user
  /etc/passwd
  setup
Users
users
  application

[Team LiB]

[Team LiB]

[SYMBOL] [A] [B] [C] [D] [E] [F] [G] [H] [I] [J] [K] [L] [M] [N] [O] [P] [R] [S] [T] [U] [**V**] [W] [X] [Y]

Variables (environment in ksh)
  table summarizing
VERITAS Volume Manager
  Mirroring and Striping
  Setup
Veritas Volume Manager
  Storage Administrator
vgcfgbackup
vgcfgrestore 2nd
vgchange 2nd 3rd 4th 5th 6th
vgcreate 2nd 3rd 4th 5th 6th
vgdisplay 2nd 3rd 4th
vgexport 2nd 3rd 4th
vgextend 2nd 3rd 4th
vgimport 2nd 3rd 4th
vgreduce
vgremove
vgscan
vgsync 2nd
vi 2nd 3rd
  adding text
  changing text
  commands
  copying text
  cursor contol
  cursor placement
  deleting text
  exit
  indenting
  joining lines
  macros
  modes
  notations
  options
  pattern matching
  repeat
  replace
  save
  search
  section positioning
  shell escape
  shell filters
  starting
  status
  undo
Virtual Partition Boot
Virtual partition monitor

Virtual Partition Startup
vmsa
vmstat 2nd
  man page
Volume (LVM)
Volume Group
  shared
Volume Group (create new)
Volume Group (LVM)
vpar driver
vPar name
vparboot 2nd 3rd 4th 5th 6th 7th 8th 9th 10th 11th 12th 13th
  man page
vparcreate 2nd 3rd 4th 5th 6th 7th 8th 9th 10th 11th
  man page
vparcreate (with setboot)
vparhb (vPars heartbeat daemon)
vparinfo command at MON> 2nd
vparload 2nd 3rd 4th 5th 6th 7th 8th 9th 10th 11th
vparload -all
vparload -p 2nd
vparload command at MON>
vparload forms 2nd
vparmodify 2nd 3rd 4th 5th 6th 7th 8th 9th 10th 11th 12th 13th 14th
  man page
vparremove 2nd
  man page
vparreset 2nd
  man page
vparresources
  man page
vPars 2nd
  background
  base vs. add onn product
  bound CPU 2nd
  Commands
  heartbeat
  instances
  requirements
  software
  unbound CPU
  vpmon
vPars commands
vPars heartbeat daemon
vPars states
vparstatus 2nd 3rd 4th 5th 6th 7th 8th 9th 10th 11th 12th 13th 14th 15th 16th 17th 18th
19th 20th 21st 22nd 23rd 24th 25th
  man page
vpartition
  man page
vparutil

man page
vpdb 2nd 3rd 4th 5th 6th 7th
vpdb synchronization
vphb
vphb (vPars heartbeat daemon)
vpmon 2nd 3rd 4th 5th 6th 7th 8th 9th 10th 11th 12th
vpmon.1
vpmon.dmp.1
vxassist
vxdctl
vxdg 2nd
vxdisk 2nd 3rd 4th 5th 6th
vxdiskadd
vxdump 2nd 3rd
vxedit
VxFS
vxinstall 2nd 3rd
vxmake
vxmend
vxplex
vxprint 2nd 3rd 4th 5th 6th
vxrestore 2nd 3rd
vxsd
vxstat
vxtrace
vxunrelocate
VxVm
VxVM 2nd 3rd
Commands
vxvol

[Team LiB]

[SYMBOL] [A] [B] [C] [D] [E] [F] [G] [H] [I] [J] [K] [L] [M] [N] [O] [P] [R] [S] [T] [U] [V] [**W**] [X] [Y]

wait
wc 2nd 3rd 4th 5th
while 2nd 3rd 4th
while Loop
Wild Cards and File Name Expansion
WINSTALL 2nd
WLM
Workload Manager (WLM)
www.docs.hp.com
www.hp.com
www.samba.org

[Team LiB]

X Window System
  sys.x11start
  Xlib
  Xt
X*devices
X*pointerkey
X*screens 2nd
Xaccess
Xconfig 2nd
Xfailsafe
xosview 2nd
xprm 2nd 3rd 4th 5th 6th 7th 8th 9th 10th 11th 12th 13th 14th 15th 16th 17th 18th 19th
Xreset
Xresources 2nd
Xservers
Xsession 2nd
Xstartup 2nd
xterm 2nd
xwd

[Team LiB]

[Team LiB]

yppasswd

[Team LiB]