# Imprimeurs on vous « spoolie »

8h30 (bon d'accord, 11h30 si j'avoue la vérité), je pose le PowerBook sur mon bureau et dans le quart d'heure qui suit, lance impression sur impression : factures, pages Web (depuis des fenêtres aussitôt refermées), courriers électroniques, etc. Je me rends à l'imprimante à l'autre bout du bâtiment et là, stupéfaction et effroi, aucun papier ! Personne aux alentours ayant emprunté par mégarde les précieux documents. Pas de message d'erreur sur l'imprimante. Il faut se rendre à l'évidence, je réalise que j'ai tout envoyé sur l'imprimante de la maison. Que faire ? Rentrer chez moi et passer pour le dernier des idiots ? Peiner à retrouver tous les documents concernés pour les imprimer à nouveau ?
Fi de tout cela, ces solutions ne sont pas dignes d'un techos !

Travaux en attente

```
$ lpstat
```

Tout simplement ! Sur quatre colonnes, vous avez respectivement les identifiants de requêtes (constitués a priori d'un numéro d'ordre préfixé du nom de la file d'impression), le nom du propriétaire, la taille du document source et enfin un horodatage.

Déplacement d'une requête d'impression

```
$ lpmove 577 imprimante_bureau
```

lpmove affecte une requête non traitée à une autre file d'impression. Il suffit pour cela de lui communiquer le numéro d'ordre de la requête et la désignation de la file pour la nouvelle destination. Ne trouvant rien dans les menus des outils d'impression dans l'environnement graphique, j'ai longtemps cru qu'Apple avait fait l'impasse sur cette fonction. Il n'en est rien et je vous laisse le soin de découvrir la solution, on ne peut plus évidente au demeurant.

Déplacement de plusieurs requêtes

```
$ for requete in $( jot - 578 603 1 ); do lpmove $requete
imprimante_bureau; done
```

Si vous avez de nombreuses requêtes à réaffecter, répéter la commande lpmove va rapidement devenir lassant. Une boucle constitue une bonne solution surtout quand elle offre le prétexte pour introduire la commande jot ainsi qu'une nouvelle syntaxe. jot est un générateur de nombres polyvalent. Dans l'expression ci-dessus, les arguments représentent les nombre de valeurs à produire (- ici indique que cet argument n'est pas significatif), la borne inférieure (578), la borne supérieure (603) et enfin le pas (on ajoute 1 à chaque fois). La syntaxe composée par les délimiteurs $( et ) est identique à celle utilisant les apostrophes inverses « ` » qui entourent une expression que l'on demande à l'interpréteur d'évaluer quelque soit sa position, y compris au cœur d'une autre expression. Ici donc, la liste de nombres fournie par la commande jot servira d'argument pour la boucle for. $() est une écriture plus moderne que `` et permet l'imbrication que la seconde formulation n'autorise pas.

Désactivation d'une file d'impression

```
$ disable imprimante_maison
```

Un empêchement subsiste avec les manipulations précédentes, le déplacement de la toute première requête qui est considérée comme étant en cours d'impression même si, par la force des choses, aucune feuille ne peut sortir. Pour contourner proprement ce blocage, il convient de préciser à la file d'impression que ses accès à l'imprimante sont momentanément hors service (on ne supprime pas la file d'impression). À ce stade on est libre d'agir sur la première requête comme sur toutes les autres.

Rétablissement de la file

```
$ /usr/bin/enable imprimante_maison
```

Il semblait évident d'utiliser enable pour réactiver une file d'impression. Manque de chance, enable désigne aussi et en priorité une commande interne de notre interpréteur de commande par défaut, Bash. Pour lever l'ambiguïté, on donne le chemin d'accès absolu du programme que l'on veut exécuter. (La commande which nous aura aidé à le trouver très facilement.)