

0.1 Administration du système depuis la console

Ce didacticiel est la suite logique du document Commandes à connaître sous GNU/Linux¹. Je propose ici de faire un zoom sur certaines commandes qui me paraissent particulièrement utiles lorsqu'on administre ou surveille un système GNU/Linux. Mes commentaires sur les commandes sont volontairement courts, cela me permet d'aborder le maximum d'outils tout en conservant un document compact. Si vous avez besoin de plus de détails sur une commande n'hésitez pas à utiliser les pages de manuel (ex : `man ls` donne une description complète de la commande `ls`). Je signale également que la quasi-totalité des commandes abordées ici accepte l'option `-help`. Cette option liste les options qu'accepte ladite commande. Enfin certaines de ces commandes sont considérées comme avancées, c'est pourquoi elles pourraient :

- ne pas être installées, à vous de trouver le bon paquetage et l'installer
- n'être exécutable que par root
- être présente dans un dossier qui n'est pas dans votre PATH (par ex : `/usr/sbin/lsof`, `/sbin/lssusb` ...) à vous de mettre à jour ce dernier.

0.1.1 1. Liste des fichiers ouverts : lsof

La commande `lsof` appelée sans paramètre retourne tous les fichiers actuellement ouverts, ce qui est rarement utile. C'est pourquoi il est conseillé de trier la sortie de cette commande via `grep`. Par exemple, vous pouvez répertorier tous les périphériques en mode caractères utilisés :

```
$ lsof | grep CHR
```

Pour afficher la liste de tous les fichiers ouverts par le processus dont le PID est 13239, tapez :

```
$ lsof -p 13239
```

Pour afficher tous les fichiers utilisés par le shell actuel, entrez :

```
$ lsof -p $$
```

La variable de shell spéciale `$$`, retourne le PID du shell.

0.1.2 2. Utilisateur qui accède aux fichiers : fuser

Vous pouvez utiliser `fuser` pour déterminer les processus ou les utilisateurs qui accèdent actuellement à certains fichiers. Par exemple, pour déterminer la liste des processus qui accèdent au contenu de `/home/Kernel`

```
$ fuser -v /home/Kernel/*
```

0.1.3 3. Propriétés d'un fichier : stat

La commande `stat` affiche les propriétés d'un fichier :

```
$ stat /etc/profile
```

Le paramètre `-filesystem` fournit des détails sur les propriétés du système de fichiers contenant le fichier spécifié :

```
$ stat /etc/profile --filesystem
```

¹<http://www.trustonme.net/didactels/130.html>

0.1.4 4. Périphériques USB : lsusb

La commande `lsusb` répertorie tous les périphériques USB. L'option `-v` permet d'imprimer une liste plus détaillée. Les informations détaillées sont lues à partir du répertoire `/proc/bus/usb/` :

```
# lsusb -v
```

0.1.5 5. Informations relatives à un périphérique SCSI : scsiinfo

La commande `scsiinfo` affiche la liste des informations relatives à un périphérique SCSI. L'option `-l` répertorie tous les périphériques SCSI connus du système (similaire à la sortie de la commande `lsscsi`). Vous trouverez ci-dessous la sortie de la commande `scsiinfo -i /dev/sda`, qui affiche des informations sur un disque dur. L'option `-a` permet d'afficher encore plus d'informations.

```
# scsiinfo -i /dev/sda
```

L'option `-d` diffuse une liste de défauts, avec deux tables qui indiquent les blocs défectueux d'un disque dur : la première est fournie par le fournisseur (table du fabricant) et la seconde répertorie les blocs défectueux qui apparaissent en cours de fonctionnement (table enrichie). Si le nombre d'entrée de la table enrichie augmente, il est conseillé de remplacer le disque dur.

0.1.6 6. Processus : top

La commande `top` (pour " Table of Processes " - table des processus) affiche la liste des processus, qui est rafraîchie toutes les deux secondes. Pour terminer le programme, appuyez sur " `q` ". Le paramètre `-n 1` termine le programme après un seul affichage de la liste des processus :

```
$ top -n 1
```

Si vous appuyez sur `F` pendant l'exécution de la commande `top`, vous voyez apparaître un menu qui permet de modifier très précisément le format de la sortie. Le paramètre `-U` suivi d'un UID (USER ID) surveille uniquement les processus associés à un utilisateur particulier. Par exemple, la commande :

```
$ top -n 1 -U $(id -u Kernel)
```

Retourne la liste des processus associés à l'utilisateur `Kernel`. La commande `id -u Kernel` retourne l'UID de l'utilisateur `Kernel`.

0.1.7 7. Liste des processus : ps

La commande `ps` génère la liste des processus. La plupart des paramètres doivent être écrits sans signe moins. Reportez-vous à `ps --help` pour obtenir de l'aide ou à la page de manuel pour obtenir une aide plus détaillée. Pour répertorier tous les processus avec des informations sur l'utilisateur, la ligne de commande, utilisez :

```
$ ps axu
```

Pour avoir la liste des processus `httpd` en cours d'exécution, utilisez l'option `-p` avec la commande `pidof` et vous obtiendrez une liste d'ID des processus indiqués.

```
$ ps -p $(pidof httpd)
```

Vous pouvez formater la liste des processus en fonction de vos besoins. L'option -L retourne la liste de tous les mots-clés. Entrez la commande suivante pour générer la liste de tous les processus, triés en fonction de la quantité de mémoire qu'ils utilisent :

```
$ ps ax --format pid,rss,cmd --sort rss
```

0.1.8 8. Arborescence de processus : pstree

La commande pstree génère la liste des processus sous forme d'arborescence :

```
$ pstree
```

Le paramètre -p ajoute l'ID du processus au nom indiqué. Pour afficher aussi les lignes de commande, utilisez le paramètre -a :

0.1.9 9. Qui fait quoi : w

A l'aide de la commande w, vous pouvez savoir qui est logué au système et ce que fait chaque utilisateur. Si des utilisateurs d'autres systèmes se sont logués à distance, le paramètre -f affiche les ordinateurs à partir desquels ils ont établi la connexion :

```
$ w -f
```

0.1.10 10. Utilisation de la mémoire : free

L'utilitaire free examine l'utilisation de la mémoire (RAM). Il affiche les détails de la quantité de mémoire libre et utilisée (ainsi que des zones d'échange) :

```
$ free
```

Les options -b,-k,-m,-g affichent les résultats respectivement en octets, Ko, Mo ou Go. Le paramètre -d delay assure le rafraîchissement de l'affichage à l'intervalle (en secondes) fixé par delay. Par exemple, free -d 1.5 procède à une mise à jour toutes les 1,5 secondes.

0.1.11 11. Tampon circulaire du noyau : dmesg

Le noyau Linux conserve certains messages dans un tampon circulaire. Pour afficher ces messages, entrez la commande dmesg :

```
$ dmesg
```

Les événements plus anciens sont consignés dans les fichiers /var/log/messages et /var/log/warn.

0.1.12 12. Les systèmes de fichiers et leur utilisation : mount, df et du

La commande mount indique le système de fichiers (périphérique et type) qui est monté, ainsi que le point de montage correspondant :

```
$ mount
```

Pour obtenir des informations sur l'utilisation totale des systèmes de fichiers, utilisez la commande df. Le paramètre -h (ou --human-readable) transforme la sortie en un texte compréhensible pour un humain :

```
$ df -h
```

Vous pouvez utiliser la commande `du` pour afficher la taille totale de tous les fichiers d'un répertoire donné et de ses sous répertoires. Le paramètre `-s` supprime la sortie des informations détaillées. L'option `-h` transforme à nouveau les données dans un format lisible pour un humain :

```
$ du -hs
```

0.1.13 13. Le système de fichiers /proc

Le système de fichiers `/proc` est un pseudo système de fichiers, dans lequel le noyau stocke les informations importantes sous forme de fichiers virtuels. Par exemple, vous pouvez afficher le type du processeur, à l'aide de la commande suivante :

```
$ cat /proc/cpuinfo
```

Vous pouvez également lancer une requête pour afficher l'allocation et l'utilisation des interruptions, avec la commande suivante :

```
$ cat /proc/interrupts
```

Voici la liste des fichiers important et des informations qu'ils contiennent :

- `/proc/devices` périphériques disponibles
- `/proc/modules` modules de noyau chargés
- `/proc/cmdline` ligne de commande du noyau
- `/proc/meminfo` informations détaillées sur l'utilisation de la mémoire

D'autres informations sont disponibles dans le fichier texte `/usr/src/linux/Documentation/filesystems`. Des informations sur les processus en cours d'exécution figurent dans les répertoires `/proc/NNN`, où `NNN` représente le PID du processus concerné. Chaque processus trouve ses propres caractéristiques dans `/proc/self/` :

```
$ ls -l /proc/self
```

L'assignation d'adresse des fichiers exécutables et des bibliothèques figure dans le fichier `maps` :

```
$ cat /proc/self/maps
```

0.1.14 13.1 procinfo

La commande `procinfo` synthétise des informations importantes provenant du système de fichiers `/proc` :

```
$ procinfo
```

Pour afficher toutes les informations, utilisez le paramètre `-a`. Le paramètre `-nN` met à jour les informations toutes les `N` secondes. Dans ce cas, vous pourrez arrêter le programme en appuyant sur la touche `q`. Par défaut, les valeurs cumulées sont affichées. Le paramètre `-d` génère des valeurs différentielles. Par exemple la commande :

```
$ procinfo -dn5
```

Affiche les valeurs qui ont changé au cours des cinq dernières secondes.

0.1.15 14. Ressources PCI : lspci

La commande lspci répertorie les ressources PCI :

```
# lspci
```

Les périphériques dont l' ID PCI est inconnu sont signalés comme " Unknown device ". Le paramètre -vv génère toutes les informations que le programme a pu obtenir par requête. Pour afficher les valeurs numériques pures, vous devez utiliser le paramètre -n.

0.1.16 15. Appels système d'une exécution de programme : strace

L'utilitaire strace permet de tracer tous les appels système d'un processus en cours d'exécution. Entrez la commande normalement, puis ajoutez strace au début de la ligne :

```
$ strace ls
```

Par exemple, pour tracer toutes les tentatives d'ouverture d'un fichier particulier, utilisez la commande suivante :

```
$ strace -e open ls .bashrc
```

Pour tracer tous les processus enfants, utilisez le paramètre -f. Vous pouvez contrôler très précisément le comportement et le format de sortie de la commande strace. Pour plus d'informations, reportez-vous à man strace.

0.1.17 16. Appels de bibliothèque d'une exécution de programme : ltrace

La commande ltrace permet de tracer les appels bibliothèque d'un processus. Cette commande s'utilise comme la commande strace. Le paramètre -c affiche le nombre et la durée des appels de bibliothèque effectués :

```
$ ltrace -c find
```

0.1.18 17. Spécification de la bibliothèque requise : ldd

Vous pouvez utiliser la commande ldd pour connaître les bibliothèques que chargent dynamiquement l'exécutable spécifié comme argument :

```
$ ldd /bin/ls
```

Les fichiers binaires statiques n'ont besoin d'aucune bibliothèque dynamique.

0.1.19 18. Informations supplémentaires sur les fichiers binaires ELF

Vous pouvez lire le contenu des fichiers binaires avec l'utilitaire readelf. Cette commande fonctionne même avec des fichiers ELF construits pour d'autres architectures matérielles :

```
$ readelf --file-header /bin/ls
```

0.1.20 19. Communication entre processus : ipcs

La commande ipcs génère la liste des ressources IPC en cours d'utilisation :

```
$ ipcs
```

0.1.21 20. Calculs de durée avec time

Vous pouvez déterminer le temps d'exécution des commandes avec l'utilitaire `time`. Cet utilitaire offre deux versions : un module intégré au shell et un programme distinct (`/usr/bin/time`). Par exemple, la commande :

```
$ time find *.txt > /dev/null
```

Affichera uniquement le temps d'exécution de la commande `find *.txt`