ps & top : lister les processus

La commande `w` nous a permis de faire rapidement le point sur l'état du système. Allons plus loin, maintenant : nous allons apprendre à lister les processus qui tournent sur votre machine.

Pour faire simple, dites-vous qu'un **processus** est un programme qui tourne en mémoire. La plupart des programmes ne font tourner qu'un processus en mémoire (une seule version d'eux-mêmes). C'est le cas d'OpenOffice par exemple. D'autres lancent des copies d'eux-mêmes, c'est le cas du navigateur Google Chrome qui crée autant de processus en mémoire que d'onglets ouverts.

Sur un serveur web, on utilise en général le logiciel Apache qui délivre les pages web aux internautes. Ce logiciel crée beaucoup de processus pour séparer ses activités. Il en va de même pour les systèmes de gestion de bases de données, comme MySQL et PostgreSQL.
Il ne faut pas s'inquiéter si un programme génère beaucoup de processus, cela n'est pas anormal.

Si vous faites la liste des processus qui tournent sur votre machine, vous risquez d'être surpris. Vous en reconnaîtrez certains, mais vous en verrez beaucoup d'autres qui ont été lancés par le système d'exploitation et dont vous n'avez jamais eu connaissance.

Pour lister les processus qui tournent sous Windows, on utilise `Ctrl + Alt + Suppr` et on va dans l'onglet « Processus ».
Sous Linux, on peut utiliser deux commandes différentes : `ps` et `top`.

ps : liste statique des processus

`ps` vous permet d'obtenir la liste des processus qui tournent au moment où vous lancez la commande. Cette liste n'est pas actualisée en temps réel, contrairement à ce que fait `top` et qu'on verra plus tard.

Essayons d'utiliser `ps` sans paramètre :

```
$ ps
  PID TTY          TIME CMD
23720 pts/0    00:00:01 bash
29941 pts/0    00:00:00 ps
```

On distingue quatre colonnes.

- `PID` : c'est le numéro d'identification du processus. Chaque processus a un numéro unique qui permet de l'identifier. Ce numéro nous sera utile plus tard lorsque nous voudrons arrêter le processus.
- `TTY` : c'est le nom de la console depuis laquelle a été lancé le processus.
- `TIME` : la durée d'exécution du processus. Plus exactement, cela correspond à la durée pendant laquelle le processus a occupé le processeur depuis son lancement.
- `CMD` : le programme qui a généré ce processus. Si vous voyez plusieurs fois le même programme, c'est que celui-ci s'est dupliqué en plusieurs processus (c'est le cas de MySQL, par exemple).

Dans mon cas, on distingue deux processus : `bash` (qui correspond à l'invite de commandes qui gère les commandes) et `ps` que je viens de lancer.

Deux processus, c'est tout ?

En fait, quand on utilise `ps` sans argument comme on vient de le faire, il affiche seulement les processus lancés par le même utilisateur (ici « mateo21 ») dans la même console (ici « pts/0 »). Cela limite énormément les processus affichés, car beaucoup sont lancés par root (l'utilisateur administrateur de la machine) et ne sont pas lancés depuis la même console que la vôtre.

La commande `ps` vous permet d'utiliser énormément d'options. Regardez le manuel pour avoir une petite idée de tout ce que vous pouvez faire avec, vous allez prendre peur.

Plutôt que de faire une longue liste des paramètres possibles, je vous propose quelques combinaisons de paramètres utiles à retenir.

`ps -ef` : lister tous les processus

Avec `ps -ef`, vous pouvez avoir la liste de tous les processus lancés par tous les utilisateurs sur toutes les consoles :

```
$ ps -ef
UID        PID  PPID  C STIME TTY          TIME CMD
root         1     0  0 01:01 ?        00:00:01 /sbin/init
root         2     1  0 01:01 ?        00:00:00 [migration/0]
root         3     1  0 01:01 ?        00:00:00 [ksoftirqd/0]
root         4     1  0 01:01 ?        00:00:00 [watchdog/0]
root         5     1  0 01:01 ?        00:00:00 [events/0]
root         6     1  0 01:01 ?        00:00:00 [khelper]
root         7     1  0 01:01 ?        00:00:00 [kthread]
root        30     7  0 01:01 ?        00:00:00 [kblockd/0]
root      2462     1  0 01:01 ?        00:00:00 /sbin/udevd --daemon
root      3292     7  0 01:01 ?        00:00:00 [kpsmoused]
root      3448     7  0 01:01 ?        00:00:00 [kgameportd]
root      4021     1  0 01:02 tty4     00:00:00 /sbin/getty 38400 tty4
root      4022     1  0 01:02 tty5     00:00:00 /sbin/getty 38400 tty5
root      4024     1  0 01:02 tty2     00:00:00 /sbin/getty 38400 tty2
root      4027     1  0 01:02 tty3     00:00:00 /sbin/getty 38400 tty3
root      4030     1  0 01:02 tty1     00:00:00 /sbin/getty 38400 tty1
root      4040     1  0 01:02 tty6     00:00:00 /sbin/getty 38400 tty6
root      4266     1  0 01:02 ?        00:00:00 /usr/sbin/acpid -
c /etc/acpi/eve
root      4363     1  0 01:02 ?        00:00:00 /sbin/syslogd
root      4417     1  0 01:02 ?        00:00:00 /bin/dd bs 1 if /proc/kmsg of /v
klog      4419     1  0 01:02 ?        00:00:00 /sbin/klogd  -
P /var/run/klogd/km
103       4440     1  0 01:02 ?        00:00:00 /usr/bin/dbus-daemon --system
107       4456     1  0 01:02 ?        00:00:03 /usr/sbin/hald

...
```

Il y en a vraiment beaucoup, je n'ai pas recopié la liste complète ici.

Vous noterez l'apparition de la colonne `UID` (*User ID*) qui indique le nom de l'utilisateur qui a lancé la commande. Il y en a beaucoup, lancés par root automatiquement au démarrage de la machine, dont vous n'avez jamais entendu parler.

`ps -ejH` : afficher les processus en arbre

Cette option intéressante vous permet de regrouper les processus sous forme d'arborescence. Plusieurs processus sont des « enfants » d'autres processus, cela vous permet de savoir qui est à l'origine de quel processus.

```
$ ps -ejH
  PID  PGID   SID TTY          TIME CMD
    1     1     1 ?        00:00:01 init
    2     1     1 ?        00:00:00   migration/0
    3     1     1 ?        00:00:00   ksoftirqd/0
    4     1     1 ?        00:00:00   watchdog/0
    5     1     1 ?        00:00:00   events/0
    6     1     1 ?        00:00:00   khelper
<surligne>    7     1     1 ?        00:00:00   kthread</surligne>
   30     1     1 ?        00:00:00     kblockd/0
   31     1     1 ?        00:00:00     kacpid
   32     1     1 ?        00:00:00     kacpi_notify
   93     1     1 ?        00:00:00     kseriod
  118     1     1 ?        00:00:04     pdflush
  119     1     1 ?        00:00:00     pdflush
  120     1     1 ?        00:00:01     kswapd0
  121     1     1 ?        00:00:00     aio/0
 1930     1     1 ?        00:00:00     ksuspend_usbd
 1931     1     1 ?        00:00:00     khubd
 2061     1     1 ?        00:00:00     ata/0
 2062     1     1 ?        00:00:00     ata_aux
 2094     1     1 ?        00:00:00     scsi_eh_0
 2263     1     1 ?        00:00:09     kjournald
 3292     1     1 ?        00:00:00     kpsmoused
 3448     1     1 ?        00:00:00     kgameportd
 4521  4521  4521 ?        00:00:00   NetworkManager
 4538  4538  4538 ?        00:00:01   avahi-daemon
 4539  4539  4539 ?        00:00:00     avahi-daemon
 4556  4556  4556 ?        00:00:00   NetworkManagerD
 4569  4569  4569 ?        00:00:00   system-tools-ba
 4570  4569  4569 ?        00:00:00     dbus-daemon
 4593  4593  4593 ?        00:00:00   gdm
 4594  4594  4593 ?        00:00:00     gdm
 4625  4625  4625 tty7     00:05:56       Xorg
 5012  5012  5012 ?        00:00:01       gnome-session
 5057  5057  5057 ?        00:00:00         ssh-agent
 5080  5012  5012 ?        00:00:25         metacity
 5083  5012  5012 ?        00:00:16         gnome-panel
 5089  5012  5012 ?        00:00:31         nautilus
 5098  5012  5012 ?        00:00:01         update-notifier
 5102  5012  5012 ?        00:00:01         evolution-alarm
 5107  5012  5012 ?        00:00:02         nm-applet
 5112  5012  5012 ?        00:01:18         gnome-cups-icon
```

```
 4640  4640  4640 ?        00:00:05   cupsd
 4672  4672  4672 ?        00:00:00   hpiod
```

Dans cette liste, vous pouvez voir que `kthread` (ici surligné) a lancé lui-même de nombreux processus, comme `kacpid`, `pdflush`…
Autre exemple : `gdm` (Gnome Desktop Manager) lance `Xorg` ainsi que gnome-session qui lui-même lance `nautilus`, `gnome-panel`, etc.

`ps -u UTILISATEUR` : lister les processus lancés par un utilisateur

Pour filtrer un peu cette longue liste, on peut utiliser `-u` afin d'obtenir par exemple uniquement les processus que l'on a lancés nous-mêmes.

```
$ ps -u mateo21
  PID TTY          TIME CMD
 5012 ?        00:00:01 gnome-session
 5057 ?        00:00:00 ssh-agent
 5060 ?        00:00:00 dbus-launch
 5061 ?        00:00:00 dbus-daemon
 5063 ?        00:00:03 gconfd-2
 5066 ?        00:00:00 gnome-keyring-d
 5068 ?        00:00:03 gnome-settings-
 5075 ?        00:00:00 sh
 5076 ?        00:00:00 esd
 5080 ?        00:00:25 metacity
 5083 ?        00:00:16 gnome-panel
 5089 ?        00:00:31 nautilus
```

Ici, j'obtiens uniquement les processus lancés par l'utilisateur « mateo21 », ce qui filtre déjà pas mal les autres processus système lancés par root.

`top` : liste dynamique des processus

La liste donnée par `ps` a un défaut : elle est **statique** (elle ne bouge pas). Or, votre ordinateur, lui, est en perpétuel mouvement. De nombreux processus apparaissent et disparaissent régulièrement.

Comment avoir une liste régulièrement mise à jour ? Avec la commande `top` !

Essayez-la :

```
top - 13:31:30 up 12:30,  3 users,  load average: 0.01, 0.07, 0.11
Tasks:  96 total,   3 running,  93 sleeping,   0 stopped,   0 zombie
Cpu(s):  1.8%us,  0.6%sy,  0.0%ni, 97.5%id,  0.0%wa,  0.0%hi,  0.0%si,  0.0%st
Mem:    515984k total,   453652k used,    62332k free,    69036k buffers
Swap:   240932k total,    31496k used,   209436k free,   246404k cached

  PID USER      PR  NI  VIRT  RES  SHR S %CPU %MEM    TIME+  COMMAND
 4625 root      15   0 38572  14m 6676 R  1.2  2.9   6:01.00 Xorg
 5068 mateo21   15   0 29760 9.8m 8008 S  0.6  1.9   0:03.69 gnome-settings-
 5112 mateo21   15   0 48612 8440 6844 S  0.6  1.6   1:19.45 gnome-cups-
icon
```

```
    1 root      18   0  2908 1848  524 S  0.0  0.4   0:01.50 init
    2 root      RT   0     0    0    0 S  0.0  0.0   0:00.00 migration/0

    3 root      34  19     0    0    0 S  0.0  0.0   0:00.01 ksoftirqd/0
    4 root      RT   0     0    0    0 S  0.0  0.0   0:00.00 watchdog/0

    5 root      10  -
5     0    0     0 S  0.0  0.0   0:00.66 events/0
    6 root      10  -
5     0    0     0 S  0.0  0.0   0:00.02 khelper
    7 root      10  -
5     0    0     0 S  0.0  0.0   0:00.00 kthread
   30 root      10  -
5     0    0     0 S  0.0  0.0   0:00.55 kblockd/0
   31 root      20  -
5     0    0     0 S  0.0  0.0   0:00.00 kacpid
   32 root      20  -
5     0    0     0 S  0.0  0.0   0:00.00 kacpi_notify
   93 root      10  -5     0    0    0 S  0.0  0.0   0:00.02 kseriod
  118 root      15   0     0    0    0 S  0.0  0.0   0:04.84 pdflush
  119 root      15   0     0    0    0 S  0.0  0.0   0:00.20 pdflush

  120 root      10  -5     0    0    0 S  0.0  0.0   0:01.29 kswapd0
```

Cette liste est **interactive** et régulièrement mise à jour.

En haut, vous retrouvez l'uptime et la charge, mais aussi la quantité de processeur et de mémoire utilisée. Nous n'entrerons pas dans les détails à ce niveau car cela demanderait un peu trop d'explications avancées sur le fonctionnement du système d'exploitation. Néanmoins, si vous savez lire la charge et la mémoire disponible, vous pouvez déjà vous faire une idée de ce qui se passe.

En dessous, vous avez la liste des processus.

Pourquoi y a-t-il si peu de processus ?

`top` ne peut pas afficher tous les processus à la fois, il ne conserve que les premiers pour qu'ils tiennent sur une « page » de la console.

Par défaut, les processus sont triés par taux d'utilisation du processeur (colonne `%CPU`). Les processus que vous voyez tout en haut de cette liste sont donc actuellement les plus gourmands en processeur. Ce sont peut-être eux que vous devriez cibler en premier si vous sentez que votre système est surchargé.

On navigue à l'intérieur de ce programme en appuyant sur certaines touches du clavier. En voilà au moins deux à connaître :

- **`q`** : ferme `top` ;
- **`h`** : affiche l'aide, et donc la liste des touches utilisables.

Attention à la différence entre majuscules et minuscules ! Taper « h » n'a pas le même effet que de taper « H » !

Mis à part cela, voici quelques commandes à connaître au sein de `top` qui peuvent vous être utiles.

- **`B`** : met en gras certains éléments.
- **`f`** : ajoute ou supprime des colonnes dans la liste.
- **`F`** : change la colonne selon laquelle les processus sont triés. En général, laisser le tri par défaut en fonction de `%CPU` est suffisant.
- **`u`** : filtre en fonction de l'utilisateur que vous voulez.
- **`k`** : tue un processus, c'est-à-dire arrête ce processus. Ne vous inquiétez pas, en général les processus ne souffrent pas. On vous demandera le numéro (`PID`) du processus que vous voulez tuer. Nous reviendrons sur l'arrêt des processus un peu plus loin.
- **`s`** : change l'intervalle de temps entre chaque rafraîchissement de la liste (par défaut, c'est toutes les trois secondes).

Vous voilà parés à utiliser `top` ! ;-)
Je l'utilise principalement pour voir la charge évoluer régulièrement tout en surveillant les processus les plus gourmands qui peuvent poser un problème.