Reprenez le contrôle à l'aide de Linux !

30 heures Facile

Mis à jour le 29/06/2021

Le contenu de ce cours n'est plus à jour

Nous avons archivé ce cours et n'actualiserons plus son contenu.

Accédez au contenu le plus récent en découvrant ce cours :

SYSTÈMES & RÉSEAUX

Initiez-vous à Linux

Easy 8 heures

Dans ce cours débutant, découvrez Linux : un système d'exploitation gratuit et fascinant qui vous donnera un contrôle sans précédent sur votre ordinateur ! Créé par des passionnés d'informatique, Linux est un vecteur important de la philosophie du libre et l'alternative parfaite à Windows ou macOS.

VOIR LE NOUVEAU COURS

Surveiller l'activité du système

Comme tous les OS actuels, Linux est un système **multi-tâches** : il est capable de gérer plusieurs programmes tournant en même temps.
Mieux encore, Linux est un système **multi-utilisateurs** : plusieurs personnes peuvent utiliser la même machine en même temps (en s'y connectant via Internet).

Tous ces programmes et ces personnes qui sont sur votre PC peuvent vite donner le tournis. Parfois, l'ordinateur peut se retrouver surchargé à cause d'un programme. Qui a lancé ce programme ? Depuis quand ? Comment arrêter un programme qui ne répond plus ?

Sous Windows, vous avez probablement entendu parler de la commande magique `Ctrl + Alt + Suppr` qui peut parfois vous sortir de bien des situations embarrassantes. Sous Linux, on utilise d'autres outils et d'autres techniques que vous allez apprendre à connaître ici.

w : qui fait quoi ?

Nous allons apprendre dans ce chapitre à utiliser une série de commandes qui nous permettront de savoir ce qui se passe actuellement dans notre ordinateur.

La première commande que je veux vous faire découvrir est très courte et facile à retenir : c'est `w` (comme la lettre, oui, oui).

C'est la première commande que je tape en général quand je me connecte à un serveur surchargé et que je veux essayer de comprendre ce qui se passe. Cela me permet de voir d'un seul coup d'oeil si la machine est vraiment surchargée (et si oui, à quel point) et si quelqu'un d'autre est en train d'intervenir sur la machine.

> Si vous utilisez Linux sur votre ordinateur personnel, tranquillement chez vous, vous êtes seuls à l'utiliser en ce moment. Pour que d'autres personnes puissent se connecter à votre ordinateur via Internet, il faut avoir configuré Linux pour ça. Nous verrons comment faire cela plus tard. On en a principalement besoin sur les serveurs.

Essayons d'utiliser `w` pour voir comment ça marche ; n'ayez pas peur, c'est sans danger :

```
$ w
 16:50:30 up  8:50,  2 users,  load average: 0,08, 0,34, 0,31
USER     TTY      FROM             LOGIN@   IDLE   JCPU   PCPU WHAT
mateo21  :0       -                19Apr08 ?xdm?   3:38m  1.18s /usr/bin/gnome-
mateo21  pts/0    :0.0             16:49    0.00s  0.33s  0.03s w
```

Bon : à première vue, c'est court mais dense, ça n'a pas l'air très clair.

Pourtant, cette commande nous donne un condensé d'informations très utiles que je vais vous présenter dans l'ordre, de gauche à droite et de haut en bas.

L'heure (aussi accessible via `date`)

Ici, l'heure qui nous est donnée est `16:50:30` (16 h 50 mn 30 s).

Cette information est aussi accessible depuis la commande `date` qui nous donne... la date, l'heure et le décalage horaire.

```
$ date
samedi 16 octobre 2010, 17:26:27 (UTC+0200)
```

La commande `date` permet en outre de modifier la date enregistrée dans l'ordinateur. C'est un peu particulier et pas très intéressant, nous ne verrons donc pas comment le faire ici (mais il vous suffit de lire le manuel si vous en avez vraiment besoin.)

L'*uptime* (aussi accessible via `uptime`)

Dans notre exemple plus haut, l'information d'*uptime* est la suivante : `up  8:50` . C'est la durée de fonctionnement de l'ordinateur.

L'uptime peut aussi être obtenu via la commande `uptime` .

En soi, cette information n'a pas l'air très utile mais elle permet quand même de savoir depuis combien de temps l'ordinateur travaille, et donc depuis combien de temps il n'a pas été redémarré.

Notez que, contrairement à Windows, il est extrêmement rare que l'installation d'un programme nous réclame de redémarrer l'ordinateur. En fait, vous avez besoin de le redémarrer principalement quand vous mettez à jour le noyau (le coeur) de Linux. Sinon, ce n'est jamais nécessaire.

Ce mode de fonctionnement est particulièrement adapté sur les serveurs qui, par définition, sont des machines qui doivent être tout le temps allumées pour servir les gens qui en ont besoin. Par exemple, les serveurs du Site du Zéro qui vous délivrent les pages du site 24 h/24 et 7 j/7 sont tout le temps allumés et nous n'avons pratiquement jamais besoin de les redémarrer. Pour preuve, l'uptime de notre serveur au moment où j'écris ces lignes :

```
$ uptime
```

```
17:45:58 up 211 days, 15:24,  1 user,  load average: 2.44, 2.66, 2.28
```

Notre serveur est en fonctionnement depuis 211 jours. Il n'a pas eu besoin d'être redémarré depuis. Cela témoigne notamment de la robustesse de Linux et de sa capacité à « tenir le coup » pendant très longtemps.

La charge (aussi accessible via `uptime` et `tload`)

En haut à droite de notre exemple, nous avons la charge. Ce sont trois valeurs décimales : `load average: 0,08, 0,34, 0,31` .

La charge est un indice de l'activité de l'ordinateur. Il y a trois valeurs :

1. la première correspond à la charge moyenne depuis 1 minute (0,08) ;
2. la seconde à la charge moyenne depuis 5 minutes (0,34) ;
3. la dernière à la charge moyenne depuis 15 minutes (0,31).

Qu'est-ce que ce nombre représente ?

C'est un peu compliqué. Si vous voulez vraiment savoir, la doc nous dit qu'il s'agit du nombre moyen de processus (programmes) en train de tourner et qui réclament l'utilisation du processeur.

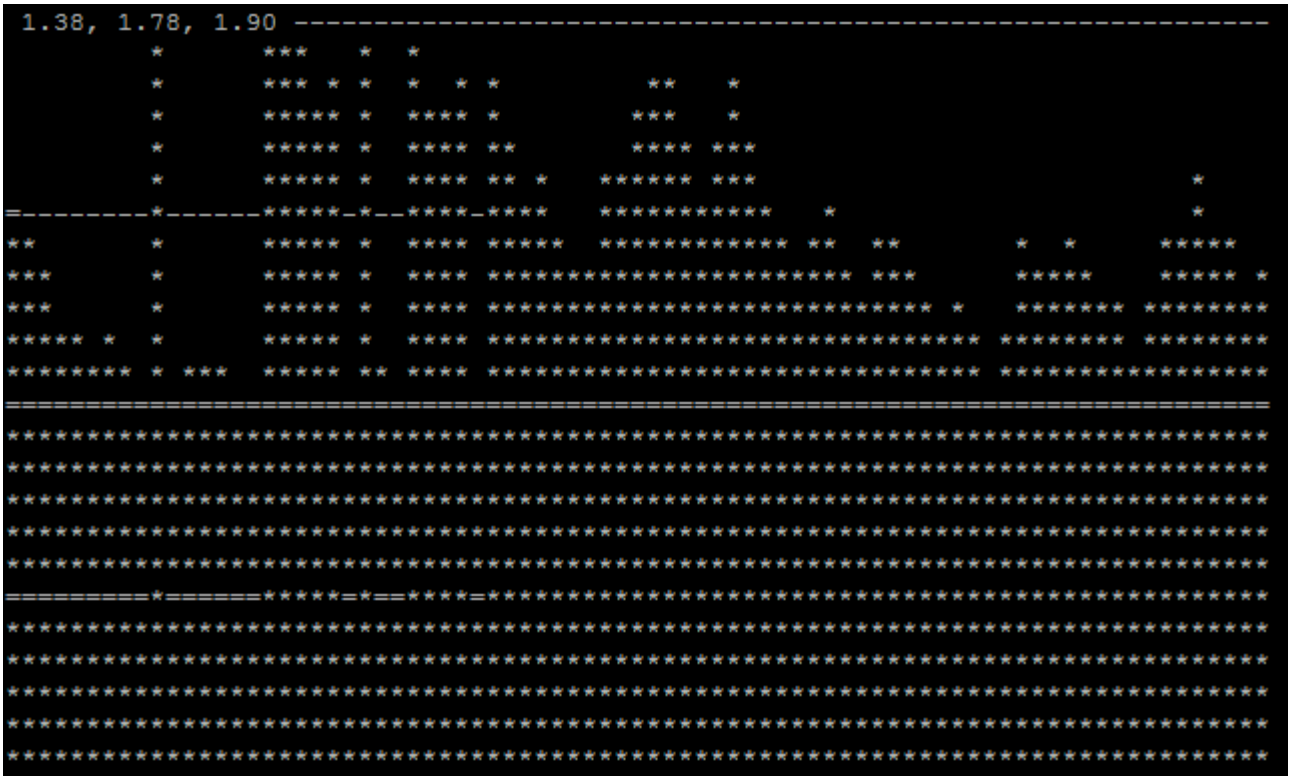
Cela veut dire que, depuis une minute, il y a en moyenne 0,33 processus qui réclament le processeur. Votre processeur est donc actif 33 % du temps.

Mais ce nombre dépend du nombre de processeurs de votre ordinateur. Un ordinateur *dual core* ne sera complètement chargé que lorsque la valeur aura atteint 2. Pour un *quad core* (4 coeurs de processeur), la valeur maximale avant surcharge sera de 4.

Bref, rien ne vous oblige à savoir ce que ce nombre signifie. Vous avez juste besoin de savoir que, lorsqu'il dépasse 1 (si vous avez un processeur), 2 ou 4, alors votre ordinateur est surchargé. J'ai déjà vu des machines avec une charge de 60, et même plus !

Quand la charge est très élevée pendant une longue période, c'est qu'il y a clairement un problème. Il y a trop de programmes qui réclament le processeur et quelque chose ne va pas dans l'ordinateur. Celui-ci aura du mal à répondre en cas de forte charge.

Notez que vous pouvez obtenir un graphique de l'évolution de la charge en console via la commande `tload` . Le graphe évolue au fur et à mesure du temps, il faut patienter un petit peu avant d'avoir quelque chose, comme l'illustre la figure suivante.



Vous pouvez quitter le graphe avec `Ctrl + C` .

La liste des connectés (aussi accessible via `who`)

Enfin, le tableau en bas qui nous est donné par `w` est surtout intéressant sur un serveur (une machine partagée par plusieurs utilisateurs). Il donne la liste des personnes connectées sur la machine, ce qu'ils sont en train de faire et depuis combien de temps.

```
USER     TTY      FROM              LOGIN@   IDLE   JCPU   PCPU WHAT
mateo21  :0       -                 19Apr08 ?xdm?   3:38m  1.18s /usr/bin/gnome
mateo21  pts/0    :0.0              16:49    0.00s  0.33s  0.03s w
```

Là, j'étais sur mon ordinateur personnel sous Ubuntu. Je ne l'ai pas configuré pour qu'on puisse s'y connecter depuis Internet (comme vous, certainement), ce qui explique pourquoi je suis seul. Certes, j'apparais deux fois. Nous allons comprendre pourquoi lorsque nous aurons appris à lire le tableau.

Il n'est pas nécessaire de décrire chacune des colonnes. Sachez qu'en gros vous avez :

- `USER` : le nom de l'utilisateur (son login) ;
- `TTY` : le nom de la console dans laquelle se trouve l'utilisateur. Souvenez-vous que sous Linux il y a en général six consoles (`tty1` à `tty6`) et qu'en plus de ça, on peut en ouvrir une infinité grâce aux consoles graphiques (leur nom commence par `pts` , en général), comme le propose le programme « Terminal » sous Unity ou « Konsole » sous KDE ;

- `FROM` : c'est l'adresse IP (ou le nom d'hôte) depuis laquelle il se connecte. Ici, comme je me suis connecté en local (sur ma propre machine, sans passer par Internet), il n'y a pas vraiment d'IP ;
- `LOGIN@` : l'heure à laquelle cet utilisateur s'est connecté ;
- `IDLE` : depuis combien de temps cet utilisateur est inactif (depuis combien de temps il n'a pas lancé de commande) ;
- `WHAT` : la commande qu'il est en train d'exécuter en ce moment. En général, si vous voyez `bash`, cela signifie que l'invite de commandes est ouverte et qu'aucune commande particulière n'est exécutée.

Dans mon cas, on voit donc deux utilisateurs (deux fois moi). Le premier correspond à la session « graphique » : on le devine notamment grâce à la dernière colonne `WHAT` qui indique que cet utilisateur est en train d'exécuter l'environnement graphique Gnome.

L'autre utilisateur est sur une console (ici, une console « graphique » lancée depuis Gnome). Cet utilisateur est en train d'exécuter... la commande `w` ! En effet, lorsque je lance `w` je me « vois » en train de l'exécuter dans la liste des utilisateurs connectés, c'est parfaitement normal.

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.

Ctrl + C & kill : arrêter un processus

Parfois, rien ne va plus. Un processus s'emballe et ne veut pas s'arrêter. Cela arrive partout, même sous Linux. À la différence de Windows toutefois, vous ne devriez pas avoir le réflexe de redémarrer « pour que ça aille mieux ». Tout peut être résolu en arrêtant les processus qui vous gênent et en les relançant au besoin.

Il y a plusieurs façons d'arrêter un processus, nous allons les étudier ici.

`Ctrl + C` : arrêter un processus lancé en console

La combinaison de touches `Ctrl + C` est à connaître. Cela demande (gentiment) l'arrêt du

programme console en cours d'exécution à l'écran. Ce raccourci se comporte ainsi en mode console seulement. En effet, en mode graphique, le comportement est le même que sous Windows : cela permet d'effectuer une copie dans le presse-papier. Notez que pour copier-coller sous Linux, on utilise souvent une autre technique : on sélectionne du texte et on clique avec la molette de la souris pour le coller ailleurs.

Prenez une commande qui n'en finit plus, comme par exemple un `find` sur l'ensemble du disque. Celui-ci va analyser tout votre disque dur à la recherche du fichier demandé. Si vous trouvez cela trop long et que vous voulez arrêter le programme en cours de route, il vous suffit de taper `Ctrl + C` :

```
# find / -name "*log*"
/dev/log
/bin/login
/sys/module/scsi_mod/parameters/scsi_logging_level
/sys/module/ehci_hcd/parameters/log2_irq_thresh
```

La liste aurait dû être beaucoup plus longue. Mais j'ai demandé l'arrêt du programme avec `Ctrl + C`, ce qui fait que j'ai pu « retrouver » l'invite de commandes rapidement et facilement.

Taper `Ctrl + C` ne coupe pas le programme brutalement, cela lui demande gentiment de s'arrêter, comme si vous aviez cliqué sur la croix pour fermer une fenêtre.

`kill` : tuer un processus

`Ctrl + C` ne fonctionne que sur un programme actuellement ouvert dans la console. De nombreux programmes tournent pourtant en arrière-plan, et `Ctrl + C` n'aura aucun effet sur eux.

C'est là que vous devez utiliser `kill` si vous voulez les arrêter (on dit aussi « tuer », c'est pareil même si ça a l'air violent).

Pour vous en servir, il faudra auparavant récupérer le `PID` du ou des processus que vous voulez tuer. Pour cela, deux solutions :

- `ps` ;
- `top`.

Ces deux commandes que nous venons de voir vous indiquent le `PID` (numéro d'identification) de chaque processus. Par exemple avec `ps` :

```
$ 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 ?        00:00:26 metacity
 5083 ?        00:00:17 gnome-panel

...

25227 pts/1    00:00:00 bash
32617 pts/1    00:00:00 man
32627 pts/1    00:00:00 pager
32703 pts/0    00:00:00 ps
```

Supposons qu'on souhaite arrêter Firefox. On peut filtrer cette longue liste avec `grep` et un pipe que nous avons appris à utiliser.

```
$ ps -u mateo21 | grep firefox
32678 ?        00:00:03 firefox-bin
```

Hop là, on a filtré Firefox de cette longue liste et on a même récupéré son `PID`. Il ne nous reste plus qu'à le tuer, avec la commande suivante :

```
kill 32678
```

Si tout va bien, la commande ne renvoie rien. Sinon, une erreur devrait s'afficher dans la console.

Vous pouvez aussi tuer plusieurs processus d'un seul coup en indiquant plusieurs `PID` à la suite :

```
kill 32678 2768 33071
```

Attention : même si `kill` est par défaut une commande « gentille » qui demande simplement au processus de s'arrêter, évitez de tuer des processus que vous ne connaissez pas. Beaucoup d'entre eux sont essentiels au bon fonctionnement de votre système, surtout ceux qui ont été lancés par root.

> J'ai essayé, mais Firefox a l'air vraiment complètement planté et il refuse de s'arrêter. Il n'y a pas moyen d'être un peu plus... direct ?

Vous voulez tuer un processus sans lui laisser le choix ?
C'est tout à fait possible, mais à n'utiliser que dans le cas d'un programme complètement planté que vous voulez vraiment arrêter !

Avec `kill -9` (comme le chiffre 9, oui, oui), vous demandez à Linux de tuer le processus sans lui laisser le temps de s'arrêter proprement. Cela peut faire le ménage quand rien ne va plus.

```
kill -9 32678
```

... tuera le processus n°32678 (Firefox, dans mon cas) immédiatement sans lui laisser le temps de finir.

`killall` : tuer plusieurs processus

Souvenez-vous : je vous ai dit que certains programmes se dupliquaient en plusieurs processus. Si vous voulez arrêter l'ensemble de ces processus, comment faire ? Heureusement, vous avez des armes pour éradiquer cette vermine.

Vous pourriez, certes, tuer tous les processus en récupérant un à un leur `PID` . Mais il y a plus rapide : `killall` (« tuez-les tous ! »).

Contrairement à `kill` , `killall` attend le nom du processus à tuer et non son `PID` .

Supposons que nous ayons trois processus `find` en cours d'exécution que nous souhaitions arrêter.

```
$ ps -u mateo21 | grep find
  675 pts/1    00:00:01 find
  678 pts/2    00:00:00 find
  679 pts/3    00:00:01 find
```

Pour tous les tuer, il faudra donc taper :

```
$ killall find
```

Si la commande ne renvoie rien, c'est que tout s'est bien passé.

En revanche, si vous avez :

```
$ killall find
find: aucun processus tué
```

... cela signifie qu'il n'y avait aucun processus de ce nom à tuer. Soit le processus n'est plus là, soit vous n'avez pas écrit correctement son nom. Vérifiez ce nom à nouveau avec la commande `ps` .

halt & reboot : arrêter et redémarrer l'ordinateur

Nous venons d'apprendre à arrêter des processus avec `kill` . Je pense que le moment est bien choisi pour découvrir comment arrêter et redémarrer l'ordinateur.

Comme je vous le disais plus tôt, il est assez rare que l'on soit forcé d'arrêter ou de redémarrer l'ordinateur. À moins d'avoir mis à jour le kernel (noyau) de Linux, il n'est jamais nécessaire de redémarrer.
L'arrêt et le redémarrage d'un serveur sous Linux sont réellement des opérations exceptionnelles.

Mais j'ai installé Linux sur mon ordinateur personnel ! Je n'en fais pas un serveur. J'ai le droit de l'arrêter ou de le redémarrer quand même, non ?

En effet, et je suppose que vous n'avez pas attendu ce chapitre pour le faire. ;-)
Vous pouviez arrêter et redémarrer l'ordinateur via l'interface graphique (Unity, KDE, ...). Mais en console, savez-vous le faire ?

`halt` : arrêter l'ordinateur

La commande `halt` commande l'arrêt immédiat de l'ordinateur. Il faut être root pour arrêter la machine ; vous devrez donc taper :

```
$ sudo halt
```

Un message sera affiché dans la console pour annoncer l'arrêt de l'ordinateur.

`reboot` : redémarrer l'ordinateur

De même, il existe la commande `reboot` pour redémarrer l'ordinateur. Il faut à nouveau être root :

```
$ sudo reboot
```

Le redémarrage prend effet immédiatement.

Les commandes `halt` et `reboot` appellent en réalité la commande `shutdown` avec des paramètres spécifiques. N'hésitez pas à lire sa page du manuel, vous verrez que vous pouvez par exemple programmer un arrêt ou un redémarrage à une heure précise ou au bout d'un certain temps.

En résumé

- Linux est multi-tâches (plusieurs programmes peuvent tourner en même temps) et multi-utilisateurs (plusieurs utilisateurs peuvent se servir de la même machine en même temps en s'y connectant via Internet).
- `w` indique quels utilisateurs sont sur la machine, ce qu'ils font et quelques autres statistiques comme la charge de travail de la machine et son uptime.
- `ps` affiche la liste des processus, c'est-à-dire des programmes qui tournent sur la machine. `top` est un équivalent qui met à jour automatiquement la liste au fil du temps.
- La combinaison de touches `Ctrl + C` permet d'arrêter une commande en cours d'exécution dans la console afin de pouvoir reprendre la main.
- `kill` tue un processus, ce qui signifie qu'il lui demande de s'arrêter. Il a besoin du numéro du processus, généralement fourni par `ps` ou `top` . Si le processus ne s'arrête pas, on peut utiliser le paramètre `-9` qui coupe brutalement le processus (avec risque de perte de données).
- `halt` commande l'arrêt de l'ordinateur, `reboot` son redémarrage.

Que pensez-vous de ce cours ?

J'AI TERMINÉ CE CHAPITRE ET JE PASSE AU SUIVANT

‹ LES TUYAUX DE REDIRECTION

EXÉCUTER DES PROGRAMMES EN ARRIÈRE-PLAN ›

Le professeur

Mathieu Nebra

Entrepreneur à plein temps, auteur à plein temps et co-fondateur d'OpenClassrooms :o)

Découvrez aussi ce cours en...