LES PROCESSUS LINUX

Lister les processus, la charge mémoire et CPU :

Pour lister les processus, il existe 3 commandes qui sont top, ps auf et pstree.

Commande top :

top est la commande qui affiche le plus d'informations sur les processus.

Ci dessous, on voit qu'il y a 105 tâches (Tasks) , 2 sont en cours d'exécution (running), 103 en attentes (sleeping).

Le taux de charge du processeur est de 27,2%.

Sur un total de 507 Mo de mémoire, 488 Mo sont utilisés, il reste 19 Mo de disponible. Le reste étant utilisé par les buffers.

Sur un total de 570 Mo de swap, 33 Mo sont utilisés, 536 Mo de disponible. Le reste sert de cache.

A quoi sert le PID ?
Le PID c'est l'identifiant unique du processus. Ce PID est nécessaire pour mettre fin à un processus.

```
cyrille@cyrille:~$ top
top - 20:10:20 up  1:19,  3 users,  load average: 0.00, 0.01, 0.00

Tasks: 105 total,   2 running, 103 sleeping,   0 stopped,   0 zombie
Cpu(s): 27.2%us,  5.0%sy,  0.3%ni, 67.4%id,  0.0%wa,  0.0%hi,  0.0%si,  0.0%st
Mem:    507728k total,   488172k used,    19556k free,    44592k buffers
Swap:   570296k total,    33860k used,   536436k free,   133436k cached

  PID USER      PR  NI  VIRT  RES  SHR S %CPU %MEM    TIME+  COMMAND

 4672 root      15   0  457m  79m 7796 S 22.6 16.1   1:32.65 Xorg
 5934 cyrille   15   0 78164  17m  10m R  7.0  3.5   0:03.33 gnome-terminal
 5256 cyrille   16   0 49432 9116 7444 S  0.7  1.8   0:20.10 gnome-cups-icon
 5848 cyrille   15   0  116m  11m 5732 S  0.7  2.4   0:08.16 beryl
 5862 cyrille   15   0  317m  78m  58m S  0.3 15.9   0:26.99 soffice.bin
 6504 cupsys    26  10  4688 2060 1532 S  0.3  0.4   0:01.17 cupsd
 8282 cyrille   15   0  2316 1160  884 R  0.3  0.2   0:00.06 top
    1 root      17   0  2912 1848  524 S  0.0  0.4   0:01.48 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   0:00.00 ksoftirqd/0
```

La commande ps auf :

La présentation différe par rapport à top.

```
cyrille@cyrille:~$ ps auf
USER       PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND

cyrille   6499  0.0  0.6   5616  3096 pts/0    Ss   17:23   0:00 bash
cyrille   6683  0.0  0.1   2560   936 pts/0    R+   17:29   0:00  \_ ps auf
root      4753  0.2  3.7  25232 19096 tty7     Ss+  16:48   0:07 /usr/X11R6/bin/
root      4102  0.0  0.1   1648   508 tty6     Ss+  16:48   0:00 /sbin/getty 384
root      4101  0.0  0.1   1652   512 tty1     Ss+  16:48   0:00 /sbin/getty 384
root      4100  0.0  0.1   1652   512 tty3     Ss+  16:48   0:00 /sbin/getty 384
root      4099  0.0  0.1   1648   508 tty2     Ss+  16:48   0:00 /sbin/getty 384
root      4096  0.0  0.1   1652   512 tty5     Ss+  16:48   0:00 /sbin/getty 384
root      4095  0.0  0.1   1652   508 tty4     Ss+  16:48   0:00 /sbin/getty 384
cyrille@cyrille:~$
```

La commande pstree :

pstree affiche les processus sous forme d'un arbre qui facilite la reconnaissance des processus père et fils.

Sur la capture ci dessous, on observe qu'il existe un seul processus père d'origine qui est init.
Tous les processus pères et fils ont pour père init.

Si on tue un processus père tous ses enfants meurent.

```
cyrille@cyrille:~$ pstree
init─┬─NetworkManager───2*[{NetworkManager}]
     ├─NetworkManagerD
     ├─acpid
     ├─atd
     ├─avahi-autoipd───avahi-autoipd
     ├─avahi-daemon───avahi-daemon
     ├─bonobo-activati───{bonobo-activati}
     ├─cron
     ├─cupsd───ipp
     ├─2*[dbus-daemon]
     ├─dbus-launch
     ├─dd
     ├─dhcdbd
     ├─dhclient3
cyrille@cyrille:~$
```

Les priorités des processus :

Voir les priorités :

Ci dessous, on observe que bash et ps ont une priorité de 0 :

```
cyrille@cyrille:~$ ps -l
F S   UID   PID  PPID  C PRI  NI ADDR SZ WCHAN  TTY  TIME CMD
0 S  1000 12072  6496  0  75   0 -  1405 wait   pts/1    00:00:00 bash
0 R  1000 12129 12072  0  76   0 -   588 -      pts/1    00:00:00 ps
cyrille@cyrille:~$
```

Modifier les priorités :

Seul root peut modifier les priorités avec la commande nice. Une priorité inférieure à 0 est prioritaire alors qu'une priorité supérieure à 0 n'est pas prioritaire.

La priorité est ajustable de -20 à +19.

Ci dessous, j'ai lancé un ps -l avec une priorité de 19 :

```
cyrille@cyrille:~$ sudo nice -n 19 ps -l
F S   UID   PID  PPID  C PRI  NI ADDR SZ WCHAN  TTY          TIME CMD
4 R     0 12595 12473  0  96  19 -   586 -      pts/0    00:00:00 ps
cyrille@cyrille:~$
```

Exécuter un processus :

Exécution au premier plan :

Il suffit de saisir que la commande :

```
cyrille@cyrille:~$ ls -l
total 376
-rw-r--r-- 1 root    root       280 2007-08-23 17:07 ;
drwx------ 2 cyrille cyrille   4096 2007-07-11 15:11 amsn_received
drwxr-xr-x 4 cyrille cyrille   4096 2007-10-18 20:29 Desktop
-rwxr--r-- 1 cyrille cyrille    179 2007-07-25 20:32 essai
-rw-r--r-- 1 cyrille cyrille   9414 2007-10-18 20:10 essai.txt
```

Exécution en arrière plan :

Dans ce contexte, le processus n'est exécuté qu'uniquement lorsque le processeur a du temps à allouer.
Il faut que la commande soit terminée par &.

```
cyrille@cyrille:~$ ls -l &
[1] 12981
cyrille@cyrille:~$ total 376
-rw-r--r-- 1 root    root       280 2007-08-23 17:07 ;
drwx------ 2 cyrille cyrille   4096 2007-07-11 15:11 amsn_received
drwxr-xr-x 4 cyrille cyrille   4096 2007-10-18 20:29 Desktop
-rwxr--r-- 1 cyrille cyrille    179 2007-07-25 20:32 essai
-rw-r--r-- 1 cyrille cyrille   9414 2007-10-18 20:10 essai.txt
lrwxrwxrwx 1 cyrille cyrille     26 2007-07-10 17:24 Examples -> /usr/share/example-content
```

Voir un processus en arrière plan:

Sur l'exemple ci dessous, on voit que la commande sleep 200 & (&=en arrière plan) est en cours d'exécution.

```
cyrille@cyrille:~$ jobs
[1]+  Running                 sleep 200 &
cyrille@cyrille:~$
```

Mettre une tâche en arrière plan :

Ctrl Z puis bg numéro du job.

Mettre une tâche en premier plan :

fg numéro du job.

Exemple :
On va lancer la commande sleep en arrière plan, puis on va la visualiser afin de la repasser en premier plan puis la remettre en arrière plan.

Lancement de la commande sleep en arrière plan :
cyrille@cyrille:~$ sleep 300 &
[1] 13639
Visualisation des processus en arrière plan :
cyrille@cyrille:~$ jobs
1]+ Running sleep 300 &

Basculement du processus d'arrière plan au premier plan :
cyrille@cyrille:~$ fg 1
sleep 300

Basculement du processus en arrière plan (ctrl z arrête le processus) :
[1]+ Stopped sleep 300

Lancement du processus en arrière plan :
cyrille@cyrille:~$ bg 1
[1]+ sleep 200 &
cyrille@cyrille:~$

Exécution de plusieurs processus :

Linux permet d'enchaîner des commandes à l'aide de séparateur.

Le séparateur ; :

Le séparateur ; placé entre 2 commandes exécute la commande 1 puis la commande 2.

Exemple : Ci dessous, on voit le résultat de ls puis de date.
cyrille@cyrille:~$ ls /home/cyrille/ ; date
; essai fichiers.doc minicom.log q
amsn_received essai.txt formation.doc monfichier
Desktop Examples laboporteoceane.doc processus.doc

mardi 23 octobre 2007, 21:30:28 (UTC+0200)
cyrille@cyrille:~$

Le séparateur && :

Le séparateur && (et) exécute la commande 2 uniquement si la commande 1 s'est bien terminée.

Exemple 1 : Cas où la commande 1 ne s'exécute pas. J'ai lancé un ls sur un répertoire qui n'existe pas.
cyrille@cyrille:~$ ls /home/cyr/ && date
ls: /home/cyr/: Aucun fichier ou répertoire de ce type
cyrille@cyrille:~$

Exemple 2 : Cas où la commande 1 s'exécute correctement. On observe que les 2 commandes se sont executés.

```
cyrille@cyrille:~$ ls /home/cyrille/ && date
;           essai     fichiers.doc       minicom.log    q
amsn_received  essai.txt  formation.doc       monfichier
Desktop       Examples   laboporteoceane.doc  processus.doc

mardi 23 octobre 2007, 21:38:20 (UTC+0200)
cyrille@cyrille:~$
```

Le séparateur || :

Le séparateur || (ou), si la première commande s'est exécutée sans erreur, la deuxième commande ne s'exécute pas. Si la première ne s'est pas exécutée correctement, la deuxième commande s'exécute.

Exemple 1 : Cas où la commande 1 s'exécute. On observe que la commande date ne s'est pas exécutée.

```
cyrille@cyrille:~$ ls /home/cyrille/ || date
;           essai     fichiers.doc       minicom.log    q
amsn_received  essai.txt  formation.doc       monfichier
Desktop       Examples   laboporteoceane.doc  processus.doc
cyrille@cyrille:~$
```

Exemple 2 : Cas où la commande 1 ne s'exécute pas. On observe que la commande date s'est exécutée.

```
cyrille@cyrille:~$ ls /home/cyr/ || date
ls: /home/cyr/: Aucun fichier ou répertoire de ce type
mardi 23 octobre 2007, 21:45:51 (UTC+0200)
cyrille@cyrille:~$
```

Mettre fin à un processus :

Pour mettre fin à un processus, il faut son PID (identifiant unique processus). On l'obtient grâce à top ou ps -aux. Voir chapitre Lister les processus, la charge mémoire et CPU.

Tuer un processus proprement :

kill -15 PID

Forcer un processus à mourir :

A n'utiliser que pour les processus qui ne meurent pas avec kill -15.

kill -9 PID

Si vous êtes bloqués dans un processus faîtes Ctrl d (équivalent au kill -15), sinon faîtes Ctrl c (équivalent au kill -9)

Exemple :

On veut tuer proprement le processus ping 192.168.1.1. Lancer 2 terminals, dans le premier, lancer ping 192.168.1.1.

Dans le deuxième, effectuer un ps aux :

ps aux pour trouver le PID
cyrille@cyrille:~$ ps aux
cyrille 14380 0.0 0.5 5556 3032 pts/2 Ss 18:54 0:00 bash
cyrille 14600 0.0 0.1 1812 540 pts/2 S+ 19:01 0:00 ping 192.168.1.1
cyrille 14605 0.0 0.1 2564 996 pts/1 R+ 19:02 0:00 ps aux

Puis kill -15 PID du ping
cyrille@cyrille:~$
kill -15 14600

Dans le 1er terminal, le ping s'est arrêté. Si on refais un ps aux, ping n'apparaît plus.

Trouver un processus :

On peut trouver un processus par son nom grâce l'utilisation combinée des commandes ps aux et grep.

ps aux | grep nom du processus

Exemple : Ouvrons 2 terminaux, dans le premier lançons un ping 192.168.1. Dans le deuxième terminal, tapez ps aux | grep ping (capture ci dessous) pour trouver le processus ping:

cyrille@cyrille:~$ ps aux | grep ping
cyrille@cyrille:~$ ps aux | grep ping
cyrille 5364 0.0 0.1 2456 884 ? S 16:49 0:00 /usr/lib/nautilus-cd-burner/mapping-daemon
cyrille 15143 0.0 0.1 1808 536 pts/2 S+ 19:18 0:00 ping 192.168.1.1
cyrille 150144 0.0 0.1 2900 764 pts/1 R+ 19:18 0:00 grep ping
cyrille@cyrille:~$

Arrêter le processus ping :
cyrille@cyrille:~$ kill -15 15143

Nouvelle recherche du processus ping, il n'apparaît plus vu que le processus est mort :
cyrille@cyrille:~$ ps aux | grep ping
cyrille 5364 0.0 0.1 2456 884 ? S 16:49 0:00 /usr/lib/nautilus-cd-burner/mapping-daemon
cyrille 15166 0.0 0.1 2896 760 pts/1 R+ 19:21 0:00 grep ping
cyrille@cyrille:~$ cyrille@cyrille:~$ ps aux | grep ping

Dupliquer données d'une commande vers un fichier :

Exemple : Nous allons afficher dans le fichier resultat les fichiers qui contiennent f1 dans leur nom.
Créer 2 fichiers f1f2 et f2f1

```
cyrille@cyrille:~$ ls | grep f1 | tee resultat
f1
f1f2
f2f1
cyrille@cyrille:~$
```

Le résultat est stocké dans le fichier resultat. Faîtes vi resultat.