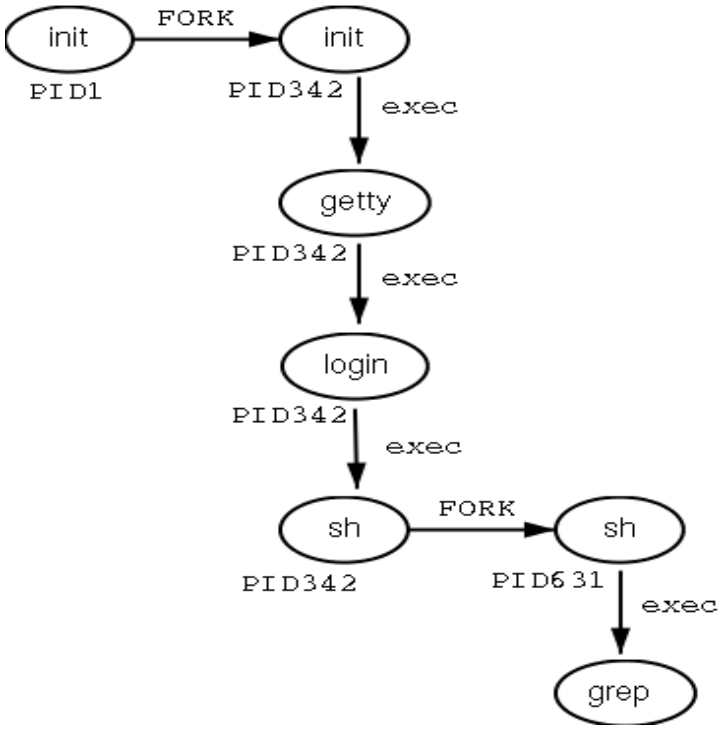
LES PROCESSUS LINUX



CONTRÔLE DES PROCESSUS

Dans un terminal, il est possible d'interagir avec le noyau ou les processus en action.
Il suffit de quelques commandes pour maîtriser la gestion des tâches.

Les commandes qui nous intéressent sont **ps, top, kill, killall, bg, fg.**

```
sh-3.2# ps
  PID TTY          TIME CMD
  190 ttys000    0:00.06 login -pf RAZA_CELMA
  344 ttys000    0:00.02 su
  346 ttys000    0:00.02 sh
  670 ttys001    0:00.04 su
  671 ttys001    0:00.02 sh
  675 ttys001    0:00.00 ps
```

Pour voir un peu ce que cette commande nous propose:

```
sh-3.2# man ps
```

The options are as follows:

-a Display information about other users' processes as well as your own.

-u Display information associated with the following keywords: user,pid, %cpu, %mem, vsz, rss, tt, state, start, time and command.

-x The -u option implies the -r option. Display information about processes without controlling termi-nals.

CONTRÔLE DES PROCESSUS

```
sh-3.2# ps aux
USER      PID %CPU %MEM     VSZ    RSS   TT  STAT STARTED     TIME COMMAND
root        1   0.0  0.1    76512    584   ??  Ss    1Jan70   0:00.82 /sbin/lau
spotlight   763   0.0  0.2    89080   2040   ??  SNs  10:56PM   0:00.28 /System/L
root      762   0.0  0.2    76664   1576   ??  Ss   10:56PM   0:00.04 /usr/sbin
root      671   0.0  0.1    75944    864 s001  S    10:47PM   0:00.10 sh
root      670   0.0  0.1    76536   1032 s001  S    10:47PM   0:00.04 su
RAZA_CELMA   645   0.0  0.1    85948   1116   ??  S    10:43PM   0:00.04 /System/L
```

On obtient donc :

-L'utilisateur,

-le PID de chaque processus: c'est son numéro d'identification,

-l'utilisation du processeur, de la mémoire vive, la date d'exécution, le temps d'exécution,

-le nom de chaque processus lancé.

CONTRÔLE DES PROCESSUS : « ps »

Trier la sortie de "ps"

Pour cibler un processus particuliers dans la sortie de “ps”, il est facile d'utiliser **”grep”** en redirigeant le résultat de “ps”:

```
sh-3.2# ps aux | grep sys
root      13   0.0  0.0    77012    464   ??  Ss    1Jan70   0:00.29 /usr/sbin/syslogd
```

CONTRÔLE DES PROCESSUS: « top »

La commande top

Il existe une commande qui permet d'avoir les mêmes informations que **ps**, mais cette fois-ci en temps réel. Vous l'aurez deviné, c'est la commande **top**.
Attention, la commande est très complexe, il faut lancer …. ! :

```
sh-3.2# top
Processes:  64 total, 3 running, 61 sleeping... 219 threads                                  23:14:15
Load Avg:  0.00,  0.00,  0.02    CPU usage:  0.99% user,  1.98% sys, 97.03% idle
SharedLibs: num =    2, resident =   77M code,     0 data, 5756K linkedit.
MemRegions: num =  5372, resident =  157M +   20M private,   92M shared.
PhysMem:  132M wired,  298M active,   43M inactive,  482M used,  542M free.
VM: 4374M + 142M   42006(0) pageins, 0(0) pageouts

  PID COMMAND      %CPU   TIME   #TH #PRTS #MREGS RPRVT  RSHRD  RSIZE  VSIZE
  847 top          3.7%  0:00.58   1    19    33  784K   212K  1380K    19M
  844 mdworker     0.0%  0:00.28   4    55    36  668K  4488K  2416K    32M
  843 SyncServer   0.0%  0:00.64   2    47    38  792K  4360K  3576K    30M
  837 mdworker     0.0%  0:00.26   3    50    33  632K  3484K  2040K    31M
```

CONTRÔLE DES PROCESSUS: « top »

Par défaut, l'affichage se rafraîchit toutes les 3 secondes.
Pour limiter le nombre ligne, on utilise :

```
sh-3.2# top -d
```

192.168.0.230 - PuTTY

```
Processes:  63 total, 2 running, 61 sleeping... 217 threads                    23:19:04
Load Avg:  0.13,  0.06,  0.04    CPU usage:  0.99% user,  2.97% sys, 96.04% idle
Networks:          1 ipkts/     OK            1 opkts /OK
Disks:             0 reads/     OK            0 writes/OK
VM:                0 pageins                  0 pageouts

  PID COMMAND      %CPU   TIME    FAULTS PGINS/COWS MSENT/MRCVD    BSD/MACH      CSW
  864 top          1.8%  0:01.46    513     0/0      961/958       74/1021       7
  857 mdworker     0.0%  0:00.27      0     0/0        0/0           0/0         0
  856 cupsd        0.0%  0:00.03      0     0/0        0/0           0/0         0
  843 SyncServer   0.0%  0:00.64      0     0/0        0/0           0/0         0
  671 sh           0.0%  0:00.14      0     0/0        0/0           0/0         0
  670 su           0.0%  0:00.04      0     0/0        0/0           0/0         0
  645 AppleSpell   0.0%  0:00.03      0     0/0        0/0           0/0         0
  644 TextEdit     0.0%  0:00.46      0     0/0        0/0           0/0         0
  636 firefox-bi   0.0%  0:49.66      0     0/0        0/0           1/2         1
  632 Preview      0.0%  0:02.73      0     0/0        0/0           0/0         0
```

CONTRÔLE DES PROCESSUS: « top »

Exemple d'options de la commande top:

top -u user_name : permet de n'avoir que les processus de "utilisateur"

top -p1234 : permet de n'avoir que le processus qui a le PID 1234

"top" est un programme interactif, il accepte de nombreuse options et commandes, si vous tapez la lettre "**h**" par exemple, vous aurez accès à l'aide qui liste les commandes disponibles:

```
sh-3.2# top -h
top usage: top [-a | -d | -e | -c <mode>]
            [-F | -f]
            [-h]
            [-i <interval>]
            [-k]
            [-L | -l <samples>]
            [-o <key>] [-O <skey>]
            [-p <format>] [-P <legend>]
            [-R | -r]
            [-s <delay>]
            [-T | -t]
            [-U <user>]
            [-u]
            [-W | -w]
            [-X | x]
            [[-n] <nprocs>]
```

CONTRÔLE DES PROCESSUS: « job,bg,fg »

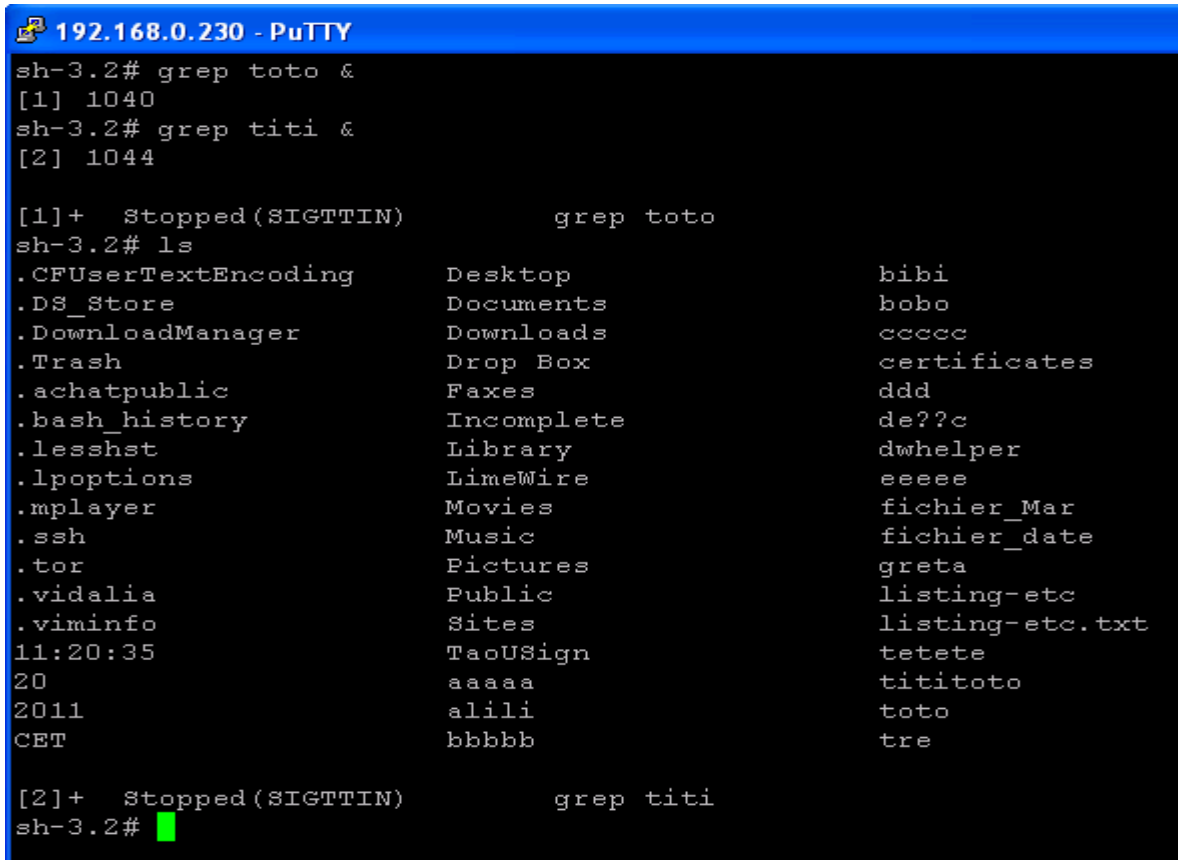
On peut mettre une commande en “arrière-plan”, c'est-à-dire que, une fois lancée, la commande ne nous bloquera pas la console jusqu'à ce qu'elle soit finie (ce qui devrait être le comportement normal) et agira sans que nous soyons obligé de nous en soucier.

Exemple 1 : je suis en console, je veux copier l'intégralité de mon disque 1 vers mon disque 2, et je sais que j'ai de multiples musiques et films libres qui prennent beaucoup de place.
Le transfert sera donc très long et je ne veux pas attendre pour taper d'autres commandes en parallèle.
On utilise l'esperluette & en fin de commande pour qu'elle soit lancée en arrière-plan :

```
sh-3.2# cp -R /dev/sda1 /dev/sd2 &
[1] 1080
<EOF>
sh-3.2#
```

CONTRÔLE DES PROCESSUS: « job,bg,fg »_

Exemple 2 :



On pourra noter que chaque commande est numérotée (ici **1** et **2**).
C'est ce numéro d'identification qui va nous servir par la suite.
Ici pour la commande 1 « grep toto » la tache s'effectue toujours en arrière plan « background » avec un PID [1].
Le PID (Process IDentifier) étant le numéro identifiant la tache en cours de recherche « grep toto »

Ici pour la commande 2 « grep titi » la tache s'effectue toujours en arrière plan « background » avec un PID [2].
Le PID (Process IDentifier) étant le numéro identifiant la tache en cours de recherche « grep titi »].

CONTRÔLE DES PROCESSUS: « job,bg,fg »

Pour maintenant remettre une commande au premier plan, on utiliser la commande fg (pour “foreground”, “premier-plan” en anglais).
Là, trois solutions :
Soit on a qu'une commande en arrière-plan, et on peut utiliser fg sans argument,
Soit on a plusieurs commandes en arrière-plan, et on utilise alors le numéro donné par jobs pour savoir quelle commande on désire récupérer.
Voici donc un exemple. Si on veut revoir au premier plan la recherche de “toto” (donc le job numéro 2):

```
sh-3.2# jobs
[1]   Stopped(SIGTTIN)      grep toto
[2]   Stopped(SIGTTIN)      grep titi
[3]-  Stopped(SIGTTIN)      grep greta
[4]+  Stopped(SIGTTIN)      grep tititoto

sh-3.2# fg %3
grep  greta
```

Notez bien l'utilisation de **%**, c'est obligatoire pour que fg trouve bien le bon processus.

C'est maintenant que les choses se corsent … Maintenant qu’on a récupéré la main sur ma commande de recherche de “titi”, je crois comprendre qu’on va en avoir pour un certains temps. On va donc à nouveau le remettre en arrière-plan. Pour cela, je vais le stopper, pour récupérer la main sur le shell, puis lui dire de continuer en arrière plan, grâce à la commande **bg** (pour “background”, “arrière plan” en anglais).
Voici donc les manipulations :

Pour stopper le processus actuellement en premier-plan, il faut faire contrôle-z (maintenez la touche **ctrl** et appuyez sur Z) noté aussi **^Z**.
Vérifier le numéro assigné à ce processus, à l'aide de **jobs** (vous pouvez remarqué qu'il est indiqué comme “stopped”, c'est-à-dire arrêté) :

CONTRÔLE DES PROCESSUS: « job,bg,fg »

C'est maintenant que les choses se corsent … Maintenant qu'on a récupéré la main sur ma commande de recherche de "titi", je crois comprendre qu'on va en avoir pour un certains temps. On va donc à nouveau le remettre en arrière-plan. Pour cela, je vais le stopper, pour récupérer la main sur le shell, puis lui dire de continuer en arrière plan, grâce à la commande **bg** (pour "background", "arrière plan" en anglais).
Voici donc les manipulations :

Pour stopper le processus actuellement en premier-plan, il faut faire contrôle-z (maintenez la touche **ctrl** et appuyez sur Z) noté aussi **^Z**.
Vérifier le numéro assigné à ce processus, à l'aide de **jobs** (vous pouvez remarqué qu'il est indiqué comme "stopped", c'est-à-dire arrêté) :

```
sh-3.2# jobs
[1]   Stopped(SIGTTIN)       grep toto
[2]   Stopped(SIGTTIN)       grep titi
[3]-  Stopped(SIGTTIN)       grep greta
[4]+  Stopped(SIGTTIN)       grep tititoto
sh-3.2# fg %3

grep greta
^Z
[3]+  Stopped(SIGTSTP)       grep greta
sh-3.2#
```

CONTRÔLE DES PROCESSUS: « job,bg,fg »_

Puis relancez-le en arrière-plan avec **bg** (j'utilise le numéro 2 ici):

```
sh-3.2# bg %3
[3] grep greta &
sh-3.2# jobs
[1]   Stopped(SIGTTIN)      grep toto
[2]   Stopped(SIGTTIN)      grep titi
[3]+  Stopped(SIGTTIN)       grep greta
[4]-  Stopped(SIGTTIN)      grep tititoto
sh-3.2#
```

CONTRÔLE DES PROCESSUS: « kill »

La commande kill

La commande **kill** va nous servir à envoyer un signal à un processus en cours d'exécution. Cette commande est puissante, attention donc à ne pas faire n'importe quoi avec.

Elle peut donc utiliser les numéros donnés par **jobs** et ainsi arrêter un processus.

Pour arrêter ma recherche sur “greta” (voir chapitre ci-dessus), il suffit donc d'écrire :

```
sh-3.2# kill %3

[3]+  Stopped(SIGTTIN)      grep greta
sh-3.2# jobs
[1]   Stopped(SIGTTIN)      grep toto
[2]   Stopped(SIGTTIN)      grep titi
[3]+  Terminated            grep greta
[4]-  Stopped(SIGTTIN)      grep tititoto
sh-3.2#
```

Le processus est arrêté.

```
[3]+  Terminated            grep greta
```

CONTRÔLE DES PROCESSUS: « kill

La commande kill

“kill” peut prendre en argument le PID d'un processus, tel que renvoyé par “top” ou “pgrep”. Prenons l'exemple d'une recherche du PID de “firefox” (navigateur web firefox), et de son utilisation avec “kill”:

```
sh-3.2# top -d 20
Processes:  66 total, 2 running, 64 sleeping... 224 threads                    00:19:27
Load Avg:  0.25,  0.13,  0.05    CPU usage: 15.38% user,  6.73% sys, 77.88% idle
Networks:          1 ipkts/     OK            1 opkts /OK
Disks:             0 reads/     OK            0 writes/OK
VM:                0 pageins                  0 pageouts

  PID COMMAND      %CPU   TIME    FAULTS PGINS/COWS MSENT/MRCVD    BSD/MACH      CSW
 1140 top          1.9%  0:00.23    537      0/0      999/996      77/1062       10
 1127 mdworker     0.0%  0:00.26      0      0/0        0/0          0/0          0
 1126 cupsd        0.0%  0:00.03      0      0/0        0/0          0/0          0
 1121 mdworker     0.0%  0:00.35      0      0/0        0/0          0/0          0
 1120 SyncServer   0.0%  0:00.45      0      0/0        0/0          0/0          0
 1068 grep         0.0%  0:00.00      0      0/0        0/0          0/0          0
 1040 grep         0.0%  0:00.00      0      0/0        0/0          0/0          0
  900 sh           0.0%  0:00.20      0      0/0        0/0          0/0          0
  899 su           0.0%  0:00.02      0      0/0        0/0          0/0          0
  889 bash         0.0%  0:00.01      0      0/0        0/0          0/0          0
  888 sshd         0.0%  0:00.32      0      0/0        0/0          8/0          1
  882 sshd         0.0%  0:00.12      0      0/0        0/0          0/0          0
  645 AppleSpell   0.0%  0:00.03      0      0/0        0/0          0/0          0
  644 TextEdit     0.0%  0:00.50      0      0/0        0/0          0/0          0
  636 firefox-bi  16.6%  1:12.21     17      0/0      311/214      4/1883       398
  632 Preview      0.0%  0:02.74      0      0/0        0/0          0/0          0
  346 sh           0.0%  0:00.02      0      0/0        0/0          0/0          0
  344 su           0.0%  0:00.02      0      0/0        0/0          0/0          0
  194 bash         0.0%  0:00.01      0      0/0        0/0          0/0          0
  190 login        0.0%  0:00.05      0      0/0        0/0          0/0          0
```

```
sh-3.2# kill 636
```

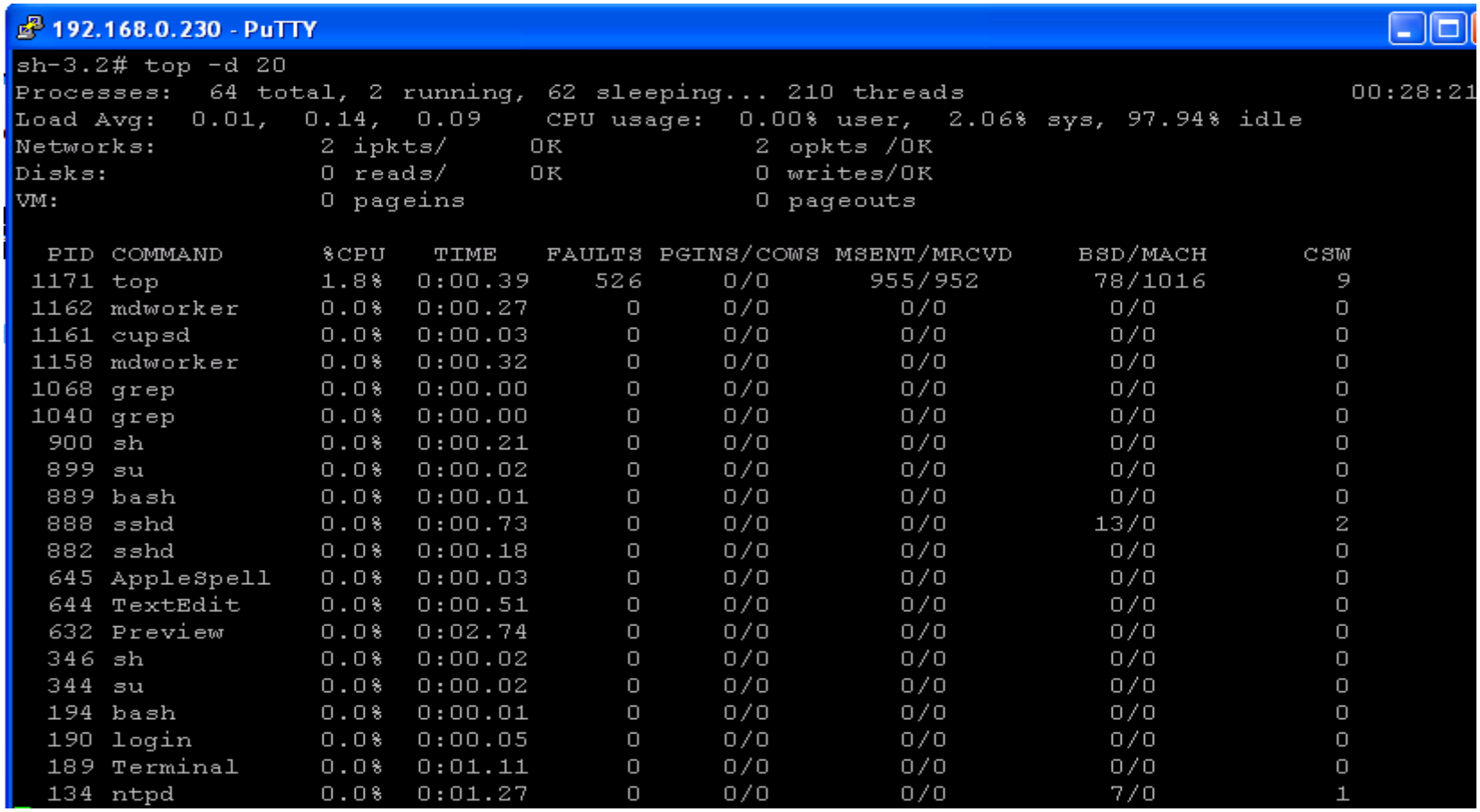
CONTRÔLE DES PROCESSUS: « kill »

La commande kill

On a donc identifier le PID de firefox qui est 636. donc pour arreter le navigateur , il suffit juste de faire un kill sur ce PID comme suivant:

```
sh-3.2# kill 636
```

Resultats :



CONTRÔLE DES PROCESSUS: « kill »

La commande kill

Pour indiquer de manière plus “conviviale” le nom du programme visé au lieu de son PID, on peut utiliser la commande “killall”:

```
sh-3.2# killall firefox
```