TD N°2
Contrôle des processus

Partie I : Systèmes de gestion de fichiers (suite)

1.1 Voir un fichier (cat et more)

La commande cat permet de lire des fichiers. Nous avons vu dans le TD1 que le répertoire /root contenait des fichiers de configuration. Ces fichiers sont simplement des fichiers textes avec un agencement et une syntaxe particulière. Regardons le contenu du fichier .bashrc qui permet de configurer à souhait son shell :

```
[root@mistra /root]# cat .bashrc
# .bashrc

# User specific aliases and functions

# Source global definitions
if [ -f /etc/bashrc ]; then
. /etc/bashrc
fi
source .sd.sh
[root@mistra /root]#
```

Une option utile de cat est -n qui permet de numéroter les lignes (ne pas oublier que cat permet de lire et non de modifier un fichier. Ainsi la numérotation de ligne apparaît à l'écran mais le fichier .bashrc n'en est pas pour autant modifié).

```
[root@mistra /root]# cat -n .bashrc
1 # .bashrc
2
3 # User specific aliases and functions
4
5 # Source global definitions
6 if [ -f /etc/bashrc ]; then
7 . /etc/bashrc
8 fi
9 source .sd.sh
[root@mistra /root]#
```

Si vous souhaitez connaître les autres options de cat, tapez au prompt "cat --help".

Vous pouvez utiliser la commande **more** pour visualiser un fichier. La commande more a l'avantage d'afficher le fichier page par page. Pour passer d'une page à l'autre, tapez sur la touche ESPACE.

1.2 Les commande head et tail

La commande **tail** est tout simplement inévitable.
Elle permet d'afficher les dernières lignes d'un fichier. Jusque là on pourrait se dire qu'après tout il suffit d'éditer le fichier et de se déplacer à la fin. D'une part c'est une méthode fastidieuse mais d'autre part, l'option -f va définitivement vous convaincre de l'utiliser :
L'option -f demande à tail de ne pas s'arrêter lorsqu'elle a affiché les dernières lignes du fichier et de continuer à afficher la suite du fichier au fur et à mesure que celui-ci grossit jusqu'à ce que l'utilisateur interrompe la commande avec la combinaison de touches d'interruption Ctrl-c.

la commande **head** réalise la même chose que tail mais elle affiche les premières lignes du fichier au lieu d'afficher les dernières. tail et head ont une option commune qui permet d'afficher le nombre de ligne que l'on souhaite :
"tail -5 nom_du_fichier" affichera les 5 dernières lignes du fichier

"head -15 nom_du_fichier" affichera les 15 premières lignes du fichier.

Par défaut, tail et head affichent 10 lignes.

1.3 La commande find

La commande find permet de trouver des fichiers dans un répertoire.

Exemple simple : comment trouver un fichier portant un nom donné ?
find directory -name targetfile –print

```
[user@myHome]$ find / -name linux-test2 -print
/home/delcros/linux-test2
```

Décomposition de la commande de l'exemple :
"***/***" indique que nous voulons chercher à partir de la racine notre fichier.
"-name " est l'option qui indique ici que nous voulons spécifier le nom d'un fichier.
"-print" demande à find d'afficher le résultat.

Un peu long n'est ce pas pour trouver la réponse dans tout cette grosse arborescence ? En général on recherche rarement un fichier depuis la racine.

Pour chercher tous les fichiers commencant par "linux-tes" et définir à partir de quel répertoire on souhaite effectuer la recherche on utilise cette syntaxe :

[user@myHome]$find /home/user -name 'linux-tes*' -print

Le nombre d'options de find est impressionnant. En voici quelques unes :

- -type permet d'indiquer le type de fichier que l'on recherche. Si vous cherchez seulement un répertoire et non pas un fichier vous pourrez utilisez cette option :

  [user@myHome]$find /usr -type d -name bin -print

  Ici, on demande à find de trouver les répertoires (l'argument "d" (comme "directory") de l'option -type indique que l'on cherche un répertoire) du nom de "bin" à partir du répertoire /usr.

- -exec ou -ok permet d'exécuter une commande sur les fichiers trouvés. La différence entre -exec et -ok est que la deuxième vous demandera pour chaque fichier trouvé si vous souhaitez réellement réaliser l'opération :

  [user@myHome]$find -name 'linux-tes*' -print -ok rm {} \;

  ./linux-test
  rm ... ./linux-test ? y
  [user@myHome]$

  Dans l'option -exec, la paire d'accolades se substitue aux fichiers trouvés, et l'anti-slash lié au point virgule forme une séquence d'échappement.

Exercice :
Dans le répertoire d'accueil, créez un répertoire **rep**. Dans ce sous répertoire, créez trois fichiers toto.c, tata.h et lala.o.
A partir du répertoire d'accueil, en utilisant la commande **find** affichez tous les fichiers du répertoire **rep** commencent par la lettre t.
3) Quelle est la commande à utiliser ? **find . –name 't*' –print ***

De même, toujours à partir du répertoire d'accueil, en utilisant la commande find, affichez tous les fichiers du répertoire **rep** de type .c
4) Quelle est la commande à utiliser ? **find rep/ –name '*.c' –print ***

De même, toujours à partir du répertoire d'accueil, en utilisant la commande find, affichez à l'écran le contenu de tous les fichiers du répertoire **rep** de type .c
4) Quelle est la commande à utiliser ? **find . –name '*.c' –exec cat '{}'\;**

1.3 La commande locate

La commande locate a la même mission que find. Pourtant vous verrez qu'en utilisant la commande locate, le fichier sera trouvé beaucoup plus rapidement. Pourquoi ? Parce que locate ne va pas chercher le fichier dans toute l'arborescence des répertoires mais

va localiser la position du fichier dans une base de données qui contient la liste des fichiers existants. Cette base de données est en général automatiquement générée une fois par jour par le système grâce à une commande appelée updatedb. Sur un système Linux, cette base de donnée se trouve dans le répertoire /usr/lib et se nomme locatedb.

La syntaxe est donc simple:
[user@myHome]$ locate nom_du_fichier

Bien que la commande locate soit très intéressante, elle ne possède pas la puissance des options de find. De plus, si vous créez des fichiers pendant la journée et que vous les recherchez avec la commande locate, il n'est pas sûr que la base de donnée ait été remise à jour. Bref, locate est un complément de find.

1.4 La commande which

which vous permet simplement de connaître le chemin d'un exécutable. Exemple:

[user@myHome]$ which ls
/bin/ls
[user@myHome]$

1.5 La commande file

La commande file permet d'afficher le type du fichier (exécutable, répertoire, ASCII, …)

file /bin/ls
file /etc/passwd
file /usr

1.6 Trouver du texte dans un fichier : commande grep

La commande grep est un pivot des commandes UNIX. Elle cherche une expression rationnelle dans un ou plusieurs fichiers, exemple :
[user@myHome]$grep fouille linux-commande.html

grep est la commande qui vous fouille les fichiers

La commande a donc affiché la ligne qui contient le mot "fouille" dans le fichier linux-commande.html.

La richesse de la commande grep permet de faire des recherches sur plusieurs fichiers et d'avoir un format de sortie adéquat.
Par exemple, le fichier linux-commande.html est déjà assez important et il serait agréable de savoir où se trouve cette ligne qui contient le mot fouille dans le fichier :
[user@myHome]$grep -n fouille linux-commande.html

902: Grep, la commande qui vous fouille les fichiers

Le mot fouille se trouve à la ligne numéro 902 et c'est l'option -n qui nous a permis de connaître ce numéro.

Une autre option très utile est -l qui permet de n'afficher que les noms des fichiers contenant ce que l'on cherche :
[user@myHome]$grep -l fouille /home/myHome/personnel/html/*

/home/myHome/personnel/html/linux-commande.html

Ici, j'ai demandé à la commande grep de chercher l'occurrence "fouille" dans les fichiers du répertoire /home/myHome/personnel/html/. Le résultat est le nom des fichiers qui contiennent l'occurrence. Ici, seul le fichier "linux-commande.html" dans le répertoire contient le mot "fouille". Quelques-unes des autres options :

- -c donne le nombre de fois où l'expression rationnelle a été rencontrée dans le fichier :
  [myHome]$ grep -c fouille linux-commande.html
  10

- -n est utile lorsque vous cherchez une expression rationnelle qui commence par un tiret car si vous n'utilisez pas l'option -n, grep la considèrera comme une option !

Exercice :
Copiez le fichier stdio.h situé dans /usr/include dans votre répertoire courant.
5) Quelle commande faut-il utiliser ?

A l'aide de la commande **grep**, affichez les lignes du fichier stdio.h qui contiennent la chaîne ***define***.
6) Quelle commande utiliser ? **grep 'define' studio.h**

Utilisez la commande grep pour isoler la ligne dans le fichier /etc/passwd qui contient vos détails.
7) Quelle est la commande à utiliser et quel est le résultat obtenu ? **grep 'login' /etc/passwd**

En utilisant la commande find ainsi que la commande grep, affichez la liste de tous les fichiers du répertoire courant et de ses sous répertoires qui ont une extension .txt.
8) Quelle est la commande à utiliser ? **find . | frep '*.txt' ***

1.6 La commande wc
La commande wc (word count) lit à partir du clavier les caractères, lignes et fichiers, en effectuant le décompte des caractères, des mots et des lignes.
Wc NomDeFichier

- Quelle commande utiliser pour afficher le nombre de caractères dans le fichier /etc/passwd ? **wc –c fichier**

- Quelle commande utiliser pour le nombre de mots dans le fichier /etc/passwd ? **wc –w fichier**

- Quelle commande utiliser pour le nombre de lignes dans le fichier /etc/passwd ? **wc –l fichier**

1.7 La commande diff

diff nous permet de découvrir les differences entre deux versions d'un fichier. Pour situer les différences dans leurs contextes, on vous conseille d'utiliser l'option -c.
diff [-c] fichier fichier

$ diff -c prog.c old_prog.c

Les lignes marquées par un point d'exclamation ! sont les lignes qui ne sont pas identiques.

Parce diff risque d'afficher beaucoup plus de lignes que l'écran (ou fenêtre) puisse accommoder, on fait souvent un tube (pipe) qui dirige les résultats de diff à la commande more. Par exemple:
$ diff -c prog.c old_prog.c | more

1.8 Redirection entrée/sorties et pipes

La connexion de plusieurs commandes : les pipes

Qu'est ce qu'un "pipe" (parfois appelé « tube ») ? Si on le décrit ce n'est rien d'autre que cette barre verticale que vous pouvez obtenir avec la combinaison de touches "Altgr + 6" sur les clavier français classiques. Un tube permet de passer le résultat d'une commande à autre commande. Un exemple permettra de comprendre tout cela beaucoup plus facilement

Je veux savoir quels sont tous les processus "bash" qui fonctionnent sur le système, mais je veux que la commande ps aux ne me fournisse que les lignes qui contiennent le mot "bash" pour m'éviter d'avoir à parcourir toute la longue liste qu'affiche ps aux :

```
[user@myHome]$ ps aux | grep bash
user   367   0.0  1.8  1600  568  p2  S  18 : 14  0 : 00  bash
user   426   0.0  2.2  1624  704  p3  S  18 : 17  0 : 00  bash
user  1261  0.0  2.2  1608  692  p6  S  21 : 22  0 : 00  bash
user  1332  0.0  2.4  1616  772  ?   S  21 : 41  0 : 00  bash
```

```
user   1582   0.0   2.7   1604   844   p8   S   22 : 30   0 : 00   bash -rcfile .bashrc
user   2796   0.0   0.9    908   300   p3   S   02 : 17   0 : 00   grep bash
root      1162   0.0   2.1   1596   664   ?    S   21 : 06   0 : 00 bash
```

On peut dire que l'on a "connecté" deux commandes entre elles. Mais vous pouvez ainsi en connecter autant que vous voulez en utilisant cette syntaxe :
commande1 | commande2 | commande3 ... | commandeN Si on se rend compte de l'utilité des pipes, progressivement on les utilise et on fini par ne plus s'en passer.

1.9 Connaître l'espace disque utilisé (df et du)

La commande df permet de connaître l'emplacement de montage des systèmes de fichiers (partitions utilisables pour stocker des fichiers) accessibles sur votre système et les capacités restantes sur chacun d'eux.

```
[user@myHome]$ df
Filesystem         1024-blocks  Used Available Capacity Mounted on
/dev/sda5            298762  119387   163945     42%   /
/dev/sda1             41166   17116    24050     42%   /dos
/dev/sda6           1745186 1163946   491042     70%   /usr
[user@myHome]$
```

La commande du permet de connaître l'utilisation disque en kilo-octet par le répertoire spécifié et ses sous répertoires.

```
[user@myHome]$ du
56      ./config
224     ./images
185     ./commandes
28      ./.xvpics
2       ./docs/preparation_debutantlinux
203     ./docs
875     .
[user@myHome]$
```

Partie II : Contrôle des processus

Il se peut qu'une commande ne termine pas ou dure trop longtemps. Cela arrivera certainement par erreur durant les travaux dirigés de programmation !
Il y aussi des commandes comme xterm qui ne se terminera que si l'on termine son propre shell (par exit).
Enfin certaines commandes sont volontairement conçues pour ne pas terminer, comme yes ou xeyes, par exemple.

Normalement, l'interpréteur de commandes attend la fin de la commande en cours avant d'en accepter une nouvelle et il se trouve donc bloqué dans les situations envisagées ci-dessus. Heureusement, on dispose de plusieurs moyens d'action sur la commande en cours.
Le plus brutal, est C-c (Control c) qui interrompt définitivement la commande, sans possibilité de reprise.
On peut aussi suspendre la commande courante par C-z.
Une commande que l'on vient de suspendre peut être relancée par la commande fg (comme foreground).

On peut également la relancer en arrière-plan par la commande bg (comme background). Une commande qui s'exécute en arrière-plan laisse l'interpéteur de commande disponible.
Une commande peut être directement lancée en arrière-plan en ajoutant un & à la fin. Par exemple, la commande :
xterm &
est équivalente à la séquence suivante :
xterm
C-z
bg

Et si on a lancé une commande comme sleep 1000 &, que peut-on faire ? Il est clair que C-c ou C-z n'agissent que sur la commande de premier-plan.
Et, comme on peut avoir plusieurs commandes en arrière-plan, il faut pouvoir désigner celle sur laquelle on veut agir. On lance alors la commande ps qui nous donne le pid du processus concerné que l'on utilise ensuite dans une commande kill appropriée, voir ps et kill ci-dessous.

On remarquera que l'on peut ainsi tuer son tcsh ... mais on s'en lasse vite !

Les commandes indispensables :

C-c
Interrompt définitivement la commande de premier plan, sans possibilité de reprise.

C-z
Suspend la commande de premier plan. Elle peut être relancée par les commandes fg ou bg.

fg
La commande fg permet de relancer au premier plan la dernière commande, qu'elle soit suspendue ou non (cas d'une commande d'arrière-plan).

bg
La commande bg permet de relancer en arrière-plan la dernière commande suspendue.

ps

La commande ps, utilisée sans argument, liste tous les processus attaché au terminal. Chaque ligne affichée commence par le PID qui est le numéro qui identifie chaque processus dans le système. On repère ainsi le pid d'un processus que l'on pourra supprimer par la commande kill.

kill
La commande kill s'utilise en général avec une option qui indique l'action voulue et prend en paramètre le pid du processus concerné. On utilise surtout :
kill -KILL pid qui interrompt définitivement
ou
kill -STOP pid qui suspend, on peut relancer ensuite par kill -CONT pid.
Une commande instructive
La commande top permet d'observer la vie d'un système Unix et de détecter les processus "gourmands". L'essai est particulièment instructif sur la machine poly mais il ne faut pas en abuser car c'est une commande qui charge fortement le système.

La commande "top" :
La commande top vous permet d'afficher des informations en continu sur l'activité du système. Elle permet surtout de suivre les ressources que les processus utilisent (quantité de RAM, pourcentage de CPU, la durée de ce processus depuis son démarrage).
Vous pourrez utiliser l'option -d pour spécifier des délais de rafraîchissement (en secondes).
En cours d'utilisation de top, il est possible de stopper un process de manière interactive en tapant k. top demande ensuite quel signal il doit envoyer : 15 (SIGTERM) est le signal par défaut qui met fin à un process, 9 (SIGKILL) est plus brutal.
Pour quitter top, appuyer simplement sur la touche "q".

La commande "ps" :
La commande ps permet de connaître les processus actifs à un moment donné :
[user@myHome]$ ps

```
PID   TTY STAT  TIME COMMAND
341   p1    S     0 : 00    bash
344   p2    S     0 : 00    bash
1039   p3    S     0 : 00    bash
1219   p3    R     0 : 00    ps
```

Le "PID" est l'identificateur d'un processus, c'est un nombre. Chaque processus est identifié dans le système par un nombre unique.

Le "TTY" indique à quel port de terminal est associé le processus.

"STAT" indique l'état dans lequel se trouve le processus. Dans l'exemple, trois processus sont endormis (S comme "sleep"), et un processus en cours d'exécution (R comme "run"). Le processus qui est en cours d'exécution n'est autre que la commande "ps" que nous venons de lancer.

Le "TIME" indique depuis combien de temps le processus utilise les ressources du microprocesseur.

Le "COMMAND" précise, comme son nom l'indique, la commande dont l'état est décrit par PID, TTY, STAT et TIME.

Ceci dit, une simple commande "ps" n'indique pas tous les processus du système. Le simple fait de lancer ps nous a juste indiquer les processus associés à un terminal et qui dépendent de l'utilisateur courant (ici " user ").
En fait, il est tout a fait probable que d'autres processus non liés à un terminal aient été lancés par " user ". J'en suis d'ailleurs sur, puisque actuellement j'utilise emacs pour réaliser cette modeste page de documentation et que pour visualiser le résultat, j'utilise netscape :
[user@myHome]$ ps -x

```
  PID TTY STAT  TIME COMMAND
  240  ?  S    0:01 /usr/X11R6/bin/fvwm2
  246  ?  S    0:00 /usr/X11/bin/xautolock -corners ++++ -time 5 -locker /usr/X
  247  ?  S    0:00 /usr/X11/bin/unclutter -idle 3
  253  ?  S    0:00 /usr/local/bin/Periodic
  254  ?  S    7:34 emacs --background grey79 -geometry 80x58+-4+-11
  257  p0 S    0:00 bash
  258  p2 S    0:00 bash
  259  p1 S    0:00 bash
  272  ?  S    0:00 /usr/lib/emacs/19.34/i386-gnu-linux/emacsserver
 2134  ?  S    0:00 /usr/bin/ispell -a -m -d francais
 6431  p0 S    1:03 /usr/lib/netscape/netscape-navigator
 6441  p0 S    0:00 (dns helper)
 6741  p0 R    0:00 ps -x
```

Les commandes qui ne sont pas associées à un terminal sont reconnaissable par le point d'interrogation qui rempli le champs TTY.

Si vous voulez connaître tous les processus de la machine de tous les utilisateurs, il suffit d'utiliser l'option ax. Si en plus vous voulez connaître les utilisateurs associés à chaque processus, il vous suffit d'utiliser l'option aux. Vous verrez alors plusieurs colonnes s'ajouter dont "USER" qui indique à quel utilisateur appartient le processus. "%CPU" indique en pourcentage les ressources du microprocesseur utilisées par le processus. "%MEM" montre en pourcentage les ressources en mémoire vive utilisées par le processus. "RSS" donne réellement la mémoire utilisée en kilobytes par le processus. "START" indique l'heure à laquelle le processus a été lancé.

La commande "kill" :
La commande "kill" permet d'expédier un signal à un processus en cours.
Sa syntaxe est la suivante :
kill [options] PID

Par exemple, si j'ai lancé une connexion à l'Internet en PPP, un processus pppd sera en cours. Pour tuer le processus, je peux d'abord faire un ps -ax pour connaître le numero du PID de pppd et ensuite si par exemple le PID est 592, je peux tuer la connexion en faisant :
[root@ myHome]# kill 592

Vous remarquerez que je suis logué en utilisateur "root" pour faire ceci, en effet le processus pppd appartenait à l'utilisateur "root" et un autre utilisateur ne peut pas lui expédier de signal.

Si un processus vous résiste, c'est à dire que vous n'arrivez pas à le tuer, vous devez utiliser la commande : kill -9 PID (PID étant toujours le numéro de de processus).

La commande "killall" permet aussi de tuer un processus mais au lieu d'indiquer le PID vous indiquerez le nom du processus.

Mais attention, plusieurs processus peuvent utiliser la même commande. Ainsi, si vous tapez :
[root@ myHome]# killall grep

Vous tuerez tous les processus qui contiennent la commande grep. Je vous recommande donc d'utiliser l'option "-i" qui vous demande une confirmation avant de tenter d'arrêter un processus..