# Un mini guide pour 'bash'

Méthodologie de la programmation

Anna Pappa

# Un peu de … shell

- Le Unix shell interpète des commandes d'utilisateur soit directement entrées par le “user” soit par un “file” qui s'appelle *script* ou *programme shell*.
- Qqs types de shell :
  - Sh ou Bourne Shell : le shell original qui est tjrs utilisé en Unix.
  - Bash ou Bourne Shell (encore) : standard GNU shell.
  - Csh ou C shell : la syntaxe ressemble au langage C.
  - Tcsh ou Tenex C shell : on l'appelle aussi turbo C shell.
  - Ksh ou Korn shell : c'est pour les gens qui sont déjà initiés à Unix.

Dans le fichier /etc/shells on peut voir les shells :

/bin/bash

/bin/csh

/bin/ksh

/bin/sh

/bin/tcsh

/bin/zsh

# shell

- Comment savoir quel est le shell par défaut :
- aller voir dans le file : /etc/passwd
- root:/bin/sh
- pour changer il suffit de taper le nom dans la ligne de commande :
- ap$ bash
- bash-3.2$
- Exemple script :

```
bash-3.2$ cat myscript.sh
#!/bin/bash
clear
echo "le programme commence maintenant."
echo "bonjour $USER"
echo
echo "ces utilisateurs sont actuellement connectes :"
w | cut -d " " -f 1 - | grep -v USER  | sort -u
echo
echo "ceci est une information uptime :"
uptime
Bash-3.2$
```

- pour connaître la version de bash :

```
bash-3.2$ bash --version
GNU bash, version 3.2.57(1)-release (x86_64-apple-darwin16)
Copyright (C) 2007 Free Software Foundation, Inc.
bash-3.2$
```

# les commandes du script :

- `uptime` : donne des informations sur l'état actuel du système comme l'heure courante, le temps depuis le dernier démarrage, le nombre d'utilisateurs actuellement connectés, et la charge de la machine.
- `uname [-a] [-m] [-r] [-s] [-v]:` informations sur la version du système, -m le type de matériel -r le numéro de version du système, -s le type de système (os), -v la version de l'os et -a toutes les informations. Sur certains Unix -p donne le type de processeur.
- `w` : statistiques sur le système : affiche une entête comme uptime puis la liste des utilisateurs connectés.

# Encore un script ...

- Cette fois on utilise vim :

```
bash-3.2$ vim script2.sh

#!/bin/bash
clear
echo "Ce script commence mainteant."
echo "Bonjour $USER !"
echo
echo "et voici une liste des utilisateurs connectés :"
echo
w
echo
echo "on va définir deux variables

COLOUR="blue"
VALUE="9"
echo "ceci est une chaine de caractères : $COLOUR"
echo "et ceciest un nombre : $VALUE"
echo
echo "et maintenant retour à la ligne de commandes."

"script2.sh" 22L, 366C
```

# Et on l'exécute :

```
bash-3.2$ sh ./script2.sh

Ce script commence mainteant.
Bonjour ap !

et voici une liste des utilisateurs connectés :

22:00  up 7 days, 19:54, 3 users, load averages: 1.18 1.40 1.51
USER     TTY      FROM              LOGIN@  IDLE WHAT
ap       console  -                 18Sep17 7days -
_mbsetupuser console  -                 18Sep17 7days -
ap       s000     -                 19:49       - w

on va définir deux variables
ceci est une chaine de caractères : blue
et ceciest un nombre : 9

et maintenant retour à la ligne de commandes.
bash-3.2$
```

# Qqs commandes :

- date : affiche la date et l'heure
- who : liste des utilisateurs connectés (ou whoami )
- cal : affiche le calendrier
- man : mode d'emploi (man cal)

On peut lier les commandes :

- who ; date

Pour se déconnecter, il suffit de taper :

- Exit (ou Ctrl D)

# Encore commandes :

- Pour changer son mot de passe :

```
$ passwd
Old password:
New password: Reenter password
```

- Pour obtenir son nom de connexion :

```
bash-3.2$ logname

ap

bash-3.2$
```

- Deux autres commandes utiles :

```
cat : concate nation de fichiers

touch : permet de cre er un fichier s'il n'existe pas, et
s'il existe de modifier sa date d'acce s et sa date de
modification, touch toto cre e le fichier toto s'il n'existe
pas
```

# Commandes suite ...

- Créer un répertoire :

```
mkdir nom_du_repertoire
```

La commande mkdir accepte un parame tre « -p » permettant de cre er une arborescence. Dans l'exemple pre ce dent, si je veux cre er DOCS/EXOS et que DOCS n'existe pas, alors :

```
$ mkdir -p DOCS/EXOS
```

va cre er a  la fois DOCS et EXOS. C'est valable pour tous les re pertoires de niveau supe rieur :

```
$ mkdir -p DOCS/EXOS/perso
```

va cre er les re pertoires DOCS, EXOS et perso s'ils n'existent pas. S'il existent ils ne sont pas modifie s.

- Pour supprimer un répertoire :

```
$ cd DOCS
$ rmdir EXOS/perso
```

# Commandes ...

- La commande cp (copy) copie un ou plusieurs fichiers vers un autre fichier ou vers un répertoire. cp fic1 fic2

  cp fic1 [fic2 ... ficn] rep1

- Exemple :

```
$ cd
$ pwd
/home/ILI123
$ touch cv.txt calc.xls image.jpg
$ ls
cv.txt calc.xls image.jpg documents
$ cp cv.txt DOCS/EXOS
$ cp calc.xls DOCS/calcul/calcul_paie.xls $ cd DOCS
$ touch fichier
$ cp fichier ..
$ cp texte/cv.txt .
```

# Commandes ...

- La commande mv (move) permet de déplacer et/ou de renommer un fichier.
- Exemple :

```
cd
$ mv cv.txt cv_toto.txt
$ mv image.jpg DOCS/images/photo_toto_cv.jpg
```

- Les options pour chercher suivant critères de nom du fichier:

```
ls a* : tous les fichiers commenc ant par a
ls a?? : tous les fichiers de trois caracte res commenc ant par a
ls a??* : tous les fichiers d'au moins trois caracte res et commenc ant par a
ls [aA]* : tous les fichiers comenc ant par a ou A
ls [a-m]?*txt : tous les fichiers commenc ant par les lettres de a a  m, posse dant au moins un second caracte re avant la terminaison txt.
```

# Les droits d'accès ...

- Sortie de la commande ls -l :

  `rwx | r-x | r--`

  propriétaire | Groupe | Autres

- La syntaxe est la suivante :

  `chmod modifications Fic1 [Fic2...]`

- S'il faut modifier les droits de l'utilisateur, on utilisera le caractère « u ». pour les droits du groupe, le caractère « g », pour le reste du monde le caractère « o », pour tous le caractère « a ».
- Pour ajouter des droits, on utilise le caractère «+», pour en retirer le caractère «-», et pour ne pas tenir compte des paramètres précédents le caractère «=».
- Enfin, le droit d'accès par lui-meme : « r » , « w » o u « x » .

  Exemple :

  **`$ chmod u=rwx,g=x,o=rw fichier1`**

# La commande 'grep'

- La syntaxe de la commande grep est

```
grep [Options] mode le [Fichier1...]
```

- Le modèle se compose de critères de recherche ressemblant beaucoup aux critères déjà exposés pour vi par exemple. Il ne faut pas oublier que ces critères doivent être interprétés par la commande grep et pas par le shell. Il faut donc verrouiller tous les caractères.

```
$ cat fichier4

Cheval
Veau
Boeuf
mouton
Mouton
boeuf


$ grep \^[\b\B\] fichier4

Boeuf
boeuf
```

# Archivage et compression

- `compress [-c] [-d] [-f], uncompress [-c] et zcat :`
  permettent de compresser, décompresser et afficher un fichier compressé au format d'encodage Lempel-Ziv. Cette méthode est moins performante que la suivante. Avec l'option -c le résultat (compressé ou décompressé) est envoyé par le canal de sortie standard. L'option -d décompresse le fichier, l'option -f force la compression même si la taille du fichier ne varie pas. L'extension par défaut du fichier est « .Z ».
- `gzip [-c] [-d] [-f] [-1->9] [-r] et gunzip :`
  version Gnu améliorée de compress et uncompress. La méthode utilisée est la même mais améliorée. Les options -c et -d sont identiques. L'option -f évite la demande de confirmation avant écrasement des fichiers, les options de -1 (fast) à -9 (best) déterminent le taux de compression. L'option -r indique la récursivité. L'extension par défaut est « .gz ».
- `tar [-c] [-x] [-t] [-v] [-f] :`
  manipulation d'archives cassette, par extension archivage par concaténation d'une série de fichiers compressés ou non. L'option -c permet la création de l'archive, -x l'extraction, -t liste le contenu de l'archive, -v inhibe le mode silence, -f permet de préciser le nom de l'archive. Par défaut la commande tar est récursive. L'extension par défaut est « .tar ». La version Gnu de tar propose l'option -z qui compresse l'archive tar avec gzip. Ex : `tar cvf archive.tar /home/toto ; gzip archve.tar`.
- `cpio [-o] [-i] [-p] [-t] [-v] [-d] [-B] [-c]:`
  copie de fichiers depuis et vers une archive. On a tendance aujourd'hui à utiliser tar car la syntaxe de cpio est un peu plus complexe. Cependant il est utile de la connaître car beaucoup de sauvegardes fonctionnent de cette manière. Les options sont -o (output) -i (input) -p (porting) -B (travailler avec des blocs de 5120 octets au lieu de 512) -c (créé une entête ASCII) -t (créé une table des matières) -v (affiche les noms des fichiers) -d (force la création de l'arborescence lors de la copie). Ex : find . -print | cpio -ocv > archive.cpio.

# Les processus

- La commande `ps` (process status) permet d'avoir des informations sur les processus en cours.

  Ex :

```
bash-3.2$ ps

  PID TTY           TIME CMD
 4662 ttys000    0:00.03 -bash
 4720 ttys000    0:00.02 bash
 4761 ttys000    0:00.02 sh
 4766 ttys000    0:00.06 bash
 4889 ttys000    0:00.09 emacs myscript.sh
 5209 ttys000    0:00.01 script
 5210 ttys001    0:00.01 /bin/bash -i
bash-3.2$
```

- Pour arrêter un processur il faut utiliser la commande **`kill`**.

  Ex : d'abord on lance **`$ps aux`**

  ensuite **`$kill xxxx`** (le PID du processus)

# Recherche de files : `find`

- La commande `find` permet de rechercher des fichiers dans l'arborescence du système de fichiers, à l'aide de critères et la possibilité d'agir sur les résultats retournés.

  **`find repertoires crite res options`**

- L'option de base est `-print` (souvent implicite sur la plupart des Unix) qui permet d'afficher sur écran les résultats.

  **`$ find . -print`**

- L'option **`-name`** permet une sélection par noms de fichiers.

  Ex.:

```
$ find . name "fic*" -print ./fic1
./fic2
./fic3
./fic4
```

# Expliquer les commandes (rappel)

- pwd :
- man :
- which :
- mkdir :
- cp :
- who :
- clear :
- touch :
- logout :
- tar :
- q :
- wc :
- grep -c :
- find -name :
  grep -n :
- file :
- ps :
- cat et more :
- Echo
- date
- Ls -al
- exit
- rm :
- mv :