Chapitre 10 – Le shell, les commandes

Une **commande shell** est une chaîne de caractères en minuscules qui peut être invoquée au travers d'une **invite de commande** (console) ou d'un **script**. Des options et des arguments peuvent la compléter. Ceux-ci sont généralement appelés paramètres de la commande.

1.1. Le shell.

Le **shell** ou **interpréteur de commande** permet d'activer des commandes soit manuellement avec la **console** soit de manière programmée sous la forme d'un **script** qui automatise l'exécution d'un ensemble de commandes.

Sh (Bourne Shell) est l'ancêtre de tous les shells. **Bash** (Bourne Again Shell) est le shell par défaut de la plupart des distributions Linux ainsi que celui du terminal de Mac OS X. Il s'agit d'une amélioration du sh.
D'autres shells existent comme **Ksh** (Korn Shell), **Csh** (C Shell), **Tcsh** (Tenex C Shell) ou encore **Zsh** (Z Shell).
On télécharge et installe un nouveau shell comme n'importe quel paquet avec la commande **apt-get install** (environnement Debian) ou **dnf install** (ex **yum install** de l'environnement RedHat).

- Le **shell** est donc le programme qui gère l'**invite de commandes**. Il attend que l'utilisateur rentre des commandes. Il est capable :
  - de se souvenir des commandes précédemment tapées (on remonte dans l'historique avec la touche flèche **Haut**) ;
  - d'autocompléter une commande ou un chemin lorsque l'on appuie sur la touche **Tab** ;
  - de rediriger et chaîner des commandes (caractères >, >>, |) ;
  - de définir des alias.
- Les **scripts shell** (programmation shell) sont des fichiers qui permettent d'**automatiser une série de commandes**. La session d'un utilisateur peut, par exemple, être paramétrée par des scripts qui sont exécutés automatiquement en début de session.
  Un script shell dépend d'un shell précis. Le langage n'est, en effet, pas tout à fait le même suivant que l'on utilise Sh, Bash ou un autre shell.
  Si le shell utilisé est Bash (script Bash), le fichier de script doit commencer par la ligne **#!/bin/bash**. Elle indique quel est le shell utilisé et où il se trouve (#! est appelé le sha-bang).
  Ensuite, les commandes à exécuter les unes après les autres sont écrites chacune sur une ligne différente.

1.2. Configurer sa console avec .bashrc.

Le fichier de configuration de la console **.bashrc** permet de personnaliser la console. Il figure dans les répertoires personnels (/root ou /home/sio).
On peut également éditer le fichier **/etc/bash.bashrc** qui concerne tous les utilisateurs. Ce fichier bashrc global doit être édité par le super utilisateur root.
Le bashrc personnel est prioritaire par rapport au bashrc global.

1.2.1. Personnaliser l'invite de commandes

Cf. TP1 SI1 §10 page 30 pour modifier le prompt dans le fichier **.bashrc** du root :

```
 GNU nano 3.2                          /root/.bashrc                          Modifié

# ~/.bashrc: executed by bash(1) for non-login shells.

# Note: PS1 and umask are already set in /etc/profile. You should not
# need this unless you want different defaults for root.
# PS1='${debian_chroot:+($debian_chroot)}\h:\w\$ '
# umask 022

# You may uncomment the following lines if you want `ls' to be colorized:
# export LS_OPTIONS='--color=auto'
# eval "`dircolors`"
# alias ls='ls $LS_OPTIONS'
# alias ll='ls $LS_OPTIONS -l'
# alias l='ls $LS_OPTIONS -lA'
#
# Some more alias to avoid making mistakes:
# alias rm='rm -i'
# alias cp='cp -i'
# alias mv='mv -i'
PS1='\[\033[01;32m\]\u@\h\[\033[00m\]:\[\033[01;34m\] \w\$\[\033[00m\]'
```

1.2.2. Créer un alias

Un alias est une commande synonyme qui permet d'éviter de saisir l'ensemble des paramètres.
Le premier alias dans l'exemple ci-dessous permet, lors de la saisie de la commande **ls**, d'activer la coloration des résultats car **ls** est en réalité systématiquement et automatiquement transformé en **ls --color=auto** (cf. également les alias **ll** et **l** ci-dessous) :

```
# You may uncomment the following lines if you want `ls' to be colorized:
export LS_OPTIONS='--color=auto'
# eval "`dircolors`"
alias ls='ls $LS_OPTIONS'
alias ll='ls $LS_OPTIONS -l'
alias l='ls $LS_OPTIONS -lA'
#
```

1.3. Le fichier .profile

De même qu'il existe un **~/.bashrc** et un **/etc/bash.bashrc**, il existe un **~/.profile** spécifique à un utilisateur et un **/etc/profile** commun à l'ensemble des utilisateurs. Ce sont des scripts de démarrage.

```
root@DS1: ~#l
total 16
-rw------- 1 root root  129 déc.  17 12:46 .bash_history
-rw-r--r-- 1 root root  634 déc.  17 12:46 .bashrc
drwxr-xr-x 3 root root 4096 déc.  17 12:33 .local
-rw-r--r-- 1 root root  148 août  17  2015 .profile
root@DS1: ~#_
```

Le fichier **.profile** est lu lorsque l'on se logue dans le cas des consoles **tty1** à **tty6** (Ctrl+Alt+F1 à F6). Le fichier **.bashrc** est lu lorsque l'on ouvre une console dans laquelle on ne se logue pas (par exemple la console graphique **Terminal** d'une Debian avec environnement de bureau ou d'une Ubuntu Desktop).

En réalité, le fichier **.profile** fait appel au fichier **.bashrc**. On peut donc configurer sa console au travers de ce dernier puisque quel que soit le type de shell, il sera lu d'une manière ou d'une autre.

1.4. Informations sur les commandes.

- La commande **which** permet de savoir quel est le fichier exécuté lorsque l'on entre le nom d'une commande.

```
root@DEB10Server: ~#which ls
/usr/bin/ls
root@DEB10Server: ~#
```

- La commande **whatis** permet de savoir rapidement à quoi sert une commande.

```
root@DEB10Server: ~#whatis rmdir
rmdir (1)            - remove empty directories
rmdir (2)            - delete a directory
root@DEB10Server: ~#_
```

- Pour obtenir des informations plus complètes sur une commande, la commande **man** *nom_de_la_commande* permet de faire appel au manuel en ligne de commande.

man rm :

```
RM(1)                                User Commands                                RM(1)

NAME
       rm - remove files or directories

SYNOPSIS
       rm [OPTION]... [FILE]...

DESCRIPTION
       This manual page documents the GNU version of rm.  rm removes each specified file.  By de-
       fault, it does not remove directories.

       If the -I or --interactive=once option is given, and there are more than  three  files  or
       the -r, -R, or --recursive are given, then rm prompts the user for whether to proceed with
       the entire operation.  If the response is not affirmative, the entire command is aborted.

       Otherwise, if a file is unwritable, standard input is a terminal, and the  -f  or  --force
       option  is  not  given,  or the -i or --interactive=always option is given, rm prompts the
       user for whether to remove the file.  If the response is  not  affirmative,  the  file  is
       skipped.

OPTIONS
       Remove (unlink) the FILE(s).

       -f, --force
              ignore nonexistent files and arguments, never prompt

       -i     prompt before every removal

       -I     prompt  once  before  removing more than three files, or when removing recursively;
              less intrusive than -i, while still giving protection against most mistakes

       --interactive[=WHEN]
              prompt according to WHEN: never, once (-I), or always (-i);  without  WHEN,  prompt
              always

 Manual page rm(1) line 1 (press h for help or q to quit)
```

1.5. Les commandes de gestion de fichiers et de répertoires.

ls : liste les fichiers d'un répertoire, affiche les attributs d'un fichier.

touch : crée un fichier.

cp : copie de fichier.

rm : supprime un fichier.

mkdir : crée un répertoire.

rmdir : supprime un dossier vide ; **rm –r** : supprime un dossier et son contenu.

mv : déplace ou renomme ou déplace et renomme un fichier ou un dossier.

ln : crée un lien.

pwd : affiche le répertoire courant.

cd : change de répertoire.

find : recherche un fichier.

cat, **more**, **less** : affichent le contenu d'un fichier.

1.6. Utilitaires.

grep : recherche de chaînes dans un fichier (permet de filtrer des données) ;

tail : affiche la fin d'un fichier ;

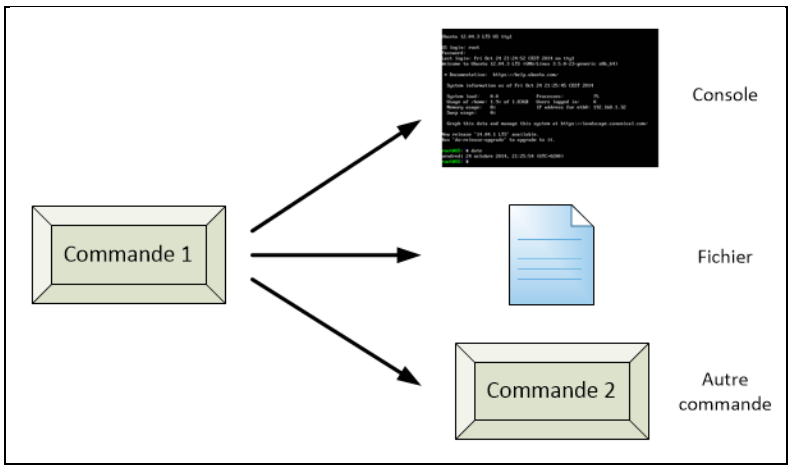
head : affiche le début d'un fichier ;

journalctl : visualiser les logs.

1.7. Les caractères spéciaux.

1.7.1. Les redirections de flux

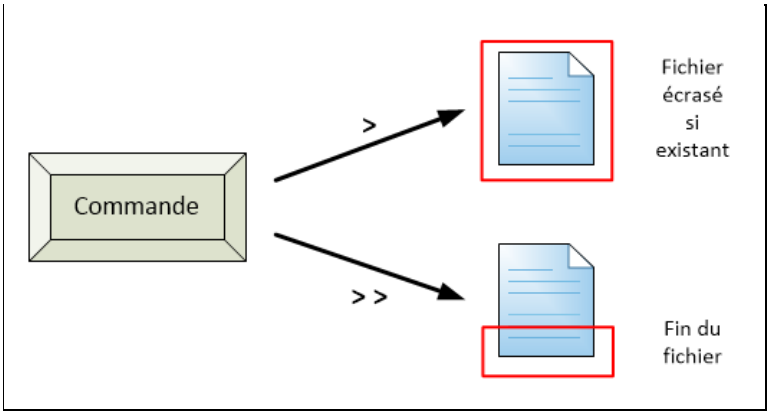
Les sorties standards comme les sorties d'erreurs sont affichées **par défaut** dans la **console**. Il est néanmoins possible de rediriger le résultat d'une commande ailleurs que dans la console :



→ Rediriger le résultat d'une commande dans un fichier

> ***fichier*** : redirige la sortie standard d'une commande dans un fichier (le fichier est créé s'il n'existe pas, il est écrasé s'il existe déjà).

>> ***fichier*** : l'écriture se fait en fin de fichier au lieu d'écraser le contenu du fichier.

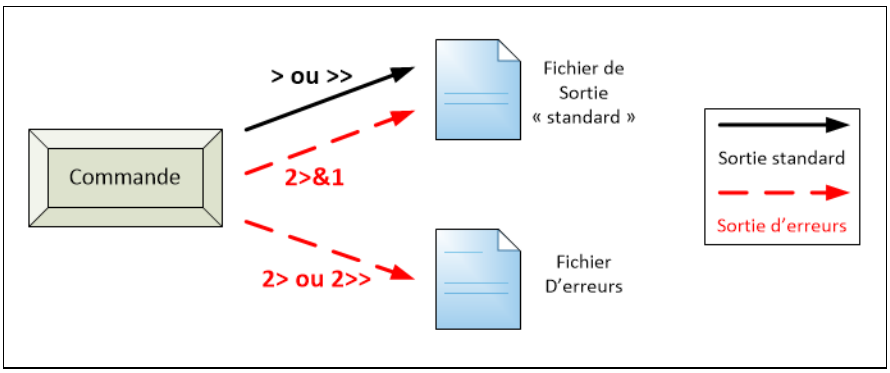


→ Rediriger les erreurs dans un fichier

2> ***fichier*** : redirige les messages d'erreur dans un fichier (s'il existe déjà, il est écrasé).

2>> ***fichier*** : redirige les sorties d'erreur à la fin d'un fichier (le fichier n'est pas écrasé s'il existe déjà).

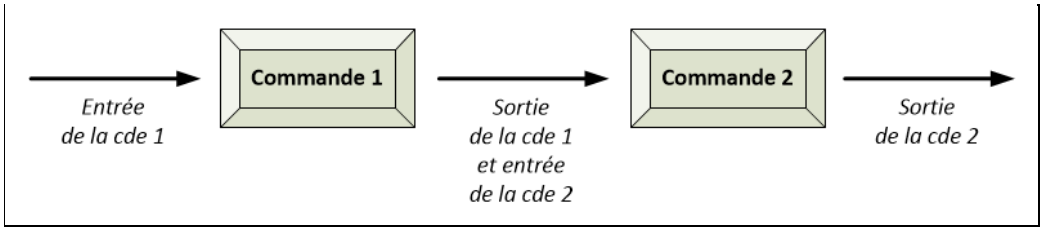
2>&1 : redirige les erreurs dans le même fichier que la sortie standard.



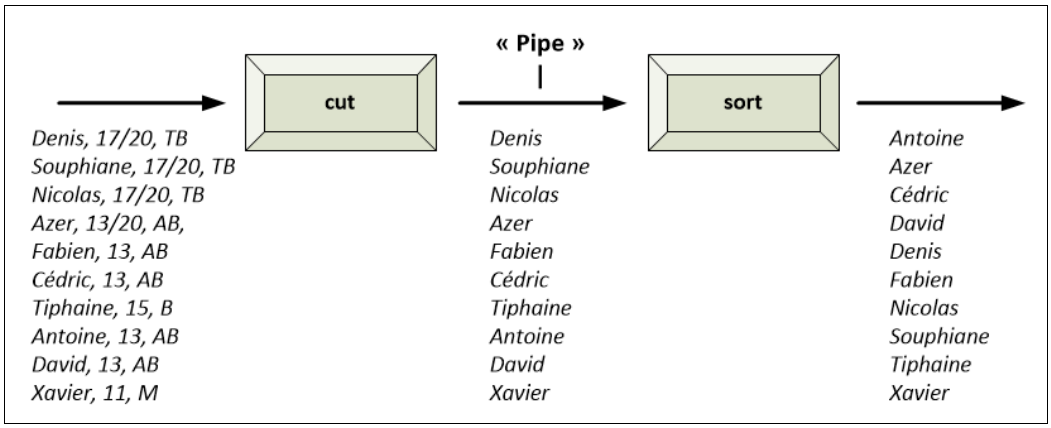
→ Chaîner les commandes avec le symbole |

Le symbole « **pipe** » (prononcer païpe) permet de chaîner des commandes, potentiellement de manière infinie, et décuple ainsi les possibilités offertes par la console.

cde1 **|** ***cde2*** : redirige la sortie standard d'une commande (cde1) en tant qu'entrée standard d'une autre commande (cde2).



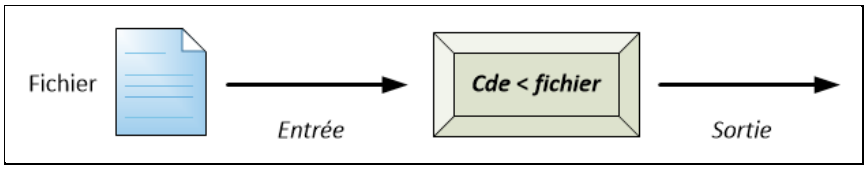
Exemple :



→ Lire depuis un fichier

On vient de voir que l'on pouvait rediriger la **sortie** d'une commande ailleurs que dans la console. L'**entrée** provient classiquement des paramètres de la commande mais on peut également faire en sorte qu'elle provienne d'un fichier.

cmde < fichier : redirige l'entrée standard d'une commande à partir d'un fichier.



1.7.2. Les jockers

***** : une suite quelconque de caractères dans un nom de fichier.

? : un caractère quelconque dans un nom de fichier.

[…] : un des caractères compris entre les crochets dans un nom de fichier.

1.7.3. Les caractères de protection

\ : annule la signification du caractère suivant.

'...' : annule la signification de l'ensemble des caractères compris entre les **quotes**.

"…" : idem mais les référencements de variables sont effectués ainsi que l'interprétation de commandes.

```
root@US1:~# n=123
root@US1:~# echo "La variable \$n vaut $n"
La variable $n vaut 123
root@US1:~# salut="bonjour à tous !"
root@US1:~# echo "Alors moi je dis : $salut"
Alors moi je dis : bonjour à tous !
root@US1:~# echo 'Alors moi je dis : $salut'
Alors moi je dis : $salut
root@US1:~# echo "Alors moi je dis : \"$salut\""
Alors moi je dis : "bonjour à tous !"
```

1.7.4. Les caractères de substitution

Les caractères de substitution (**anti-quotes**) permettent de remplacer une commande par l'affichage résultant de son exécution. Ce mécanisme est utilisé pour insérer dans une ligne de commande le résultat d'une autre commande :

```
root@US1:~# echo date
date
root@US1:~# echo `date`
lundi 7 novembre 2016, 02:52:28 (UTC+0100)
root@US1:~#
```

1.8. L'éditeur de texte Vi (ou VIM).

→ **Mode interactif** (mode par défaut au lancement de VIM)

x : suppression du caractère courant

dd : suppression de la ligne courante (elle est en fait coupée)

yy : copie de la ligne courante

p : coller

u : annule la dernière modification

→ La touche **i** bascule l'utilisateur dans le **mode insertion**. On sort de ce mode par la touche d'**échappement**.

→ La touche **:** permet de passer en **mode commande**.

:q! permet de quitter l'éditeur sans enregistrer les modifications.

:w permet d'enregistrer les modifications.

:wq permet de sauvegarder les modifications et de quitter l'éditeur.

1.9. La commande find.

La commande **find** recherche des fichiers dans une arborescence. Par défaut, elle n'affiche que le chemin des fichiers trouvés en fonction des critères de recherche donnés. Elle peut également exécuter des commandes sur chacun des fichiers trouvés.

1.9.1. Les principaux critères de sélection

-name *fichier* : on recherche les fichiers dont le nom est fichier. Il est possible d'utiliser des jockers mais ils doivent être entre quotes.

-type f ou **d** : on recherche les fichiers d'un type particulier (f pour fichier ordinaire et d pour répertoire).

-size *taille* : on recherche les fichiers ayant une certaine taille.

-group *g* : on recherche les fichiers appartenant au groupe *g*.

-user *u* : on recherche les fichiers appartenant à l'utilisateur u.

-perm *d* : on recherche les fichiers ayant les droits d.

-mtime *n* : on recherche les fichiers modifiés il y a n jours.

Remarque : quand un argument est numérique (par exemple celui de **-mtime**), il peut être précédé de + ou de – pour indiquer un seuil minimal ou au contraire une valeur maximale. Ainsi, **-mtime +5** signifie que l'on recherche les fichiers modifiés il y au moins 5 jours.

1.9.2. Les critères d'action

-print affiche le chemin des fichiers trouvés (par défaut).

-exec *cde* **{ }** \ **;** exécute la commande *cde* pour chaque fichier trouvé.

1.9.3. Les opérateurs

-a : ET (par défaut)

-o : OU

! : Négation