



UNIVERSITÉ CADI AYYAD  
FACULTÉ DES SCIENCES SEMLALIA  
MARRAKECH



# Système d'exploitation LINUX - Introduction au Shell -

DR. MOURDI YOUSSEF  
YOUSSEF.MOURDI@CED.UCA.MA

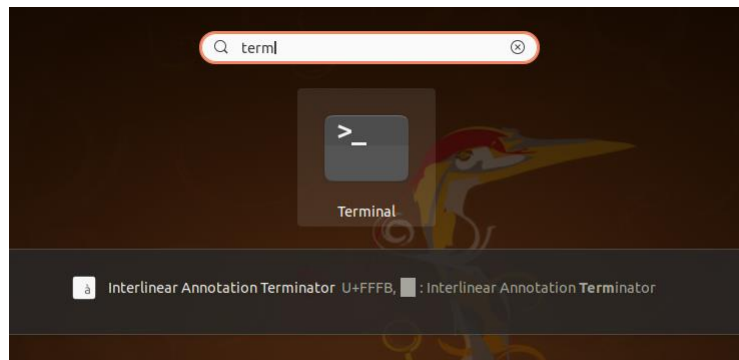
## Plan du chapitre

- Découvrir les fonctionnalités du Shell bash
- Utilisations des options des commandes
- Utilisations des alias et des méta caractères
- Enchaînement des commandes
- Utilisation de l'aide

## La ligne de commande Linux

- L'interface de ligne de commande (CLI) est une interface fournie par une application appelée terminal, elle sert d'interface entre l'homme et le système.
- Le terminal reçoit les commandes entrées par l'utilisateur et les fait passer au Shell
- Le Shell interprète les commandes entrées par l'utilisateur en instructions qui vont être exécutées par le noyau
- le résultat de la commande (s'il existe) est affiché sur le terminal.

## La ligne de commande Linux



Pour accéder au terminal : Applications > Recherche > terminal

## Le prompt (invite de commande)

- Le terminal affiche un prompt indiquant à l'utilisateur que le terminal est prêt à recevoir une commande
- Le prompt à souvent la forme suivante

```
youssef@namenode:~$
```

- Le prompt affiche :
  - Le nom de l'utilisateur : ex. youssef
  - Le nom de la machine: ex. namenode
  - L'emplacement courant : ex ~ indique le répertoire de travail de l'utilisateur
  - Et le privilège : \$ ou % pour un utilisateur normal # pour le root (Administrateur)

## Le Shell

- Le Shell est l'interpréteur qui traduit les commandes entrées par l'utilisateur en actions que le noyau doit exécuter
- Il existe plusieurs types de Shell mais le plus utilisé par les distributions Linux est bash (Bourne-again-shell)
- Le Shell bash offre plusieurs fonctionnalités
  - Historique des commandes
  - Scripting
  - Alias
  - Variables...

# Fonctionnalités du bash

- **Historique des commandes**
  - Permet d'accéder facilement aux commandes déjà exécutées et les réexécuter
- **Scripting**
  - Un script shell est fichier exécutable contenant plusieurs commandes
  - Le rôle d'un script est d'automatiser l'exécution de plusieurs commandes en une seule (le script)
- **Alias**
  - Permet de donner des noms réduits à des commandes dont le nom est long.
- **Variables**
  - Les variables permettent de stocker des informations utilisées par le Shell bash.

# Le format des commandes

- Le format d'une commande est le suivant :

**commande [options] [arguments]**

- Les options permettent de modifier le comportement de la commande.
- Les arguments spécifient sur quel élément la commande sera appliquée ( fichier, utilisateur , disque...)
- Chaque argument et chaque option doit être séparées des autres par des espaces
- Les options peuvent être combinés
- Certaines commandes n'exigent ni arguments ni options.
- Il faut noter que Linux fait la différence entre le majuscule et le minuscule pour les commandes , options et arguments

## Format des commandes (exemples)

- La commande **ls** sans arguments permet de lister les fichiers et les dossiers contenu dans le répertoire de travail courant

```
youssef@namenode:~$ ls
Bureau Documents Images Modèles Musique Public Téléchargements Vidéos
youssef@namenode:~$
```

- On peut aussi passer un ou plusieurs arguments à la commande **ls** pour spécifier le répertoire dont on souhaite afficher le contenu
- Dans cet exemple on a fournit comme argument à la commande **ls** le répertoire /etc/ppp

```
youssef@namenode:~$ ls /etc/ppp
chap-secrets ip-down.d ip-up.d ipv6-down.d ipv6-up.d options.pptp peers
ip-down      ip-up      ipv6-down  ipv6-up    options    pap-secrets
youssef@namenode:~$
```

## Format des commandes (exemples)

- Ici on a donné deux arguments à la commande ls : /etc/ppp et /etc/ssh

```
youssef@namenode:~$ ls /etc/ppp /etc/ssh
/etc/ppp:
chap-secrets ip-down.d ip-up.d ipv6-down.d ipv6-up.d options.pptp peers
ip-down      ip-up      ipv6-down  ipv6-up    options    pap-secrets

/etc/ssh:
moduli      ssh_config.d sshd_config.d  ssh_host_ecdsa_key.pub ssh_host_ed25519_key.pub ssh_host_rsa_key.pub
ssh_config  sshd_config  ssh_host_ecdsa_key ssh_host_ed25519_key  ssh_host_rsa_key  ssh_import_id
youssef@namenode:~$
```

## Utilisation des options

- Une option permet d'étendre ou de modifier le comportement d'une commande
- Une option est le plus souvent une seule lettre mais parfois elle peut-être un mot
- Les options en une seule lettre sont précédées par un tiret (-) alors que les options en mot entier sont précédée par double tiret (--)

## Utilisation des options (exemples)

- Dans cette exemple on a utilisé la commande **ls** avec l'option **-l** qui permet d'afficher les fichiers et dossiers contenus dans le répertoire courant ainsi que d'autres informations tel que : les permissions ,la taille du fichier etc...

```
youssef@namenode:~$ ls -l
total 32
drwxr-xr-x 2 youssef youssef 4096 nov. 7 22:06 Bureau
drwxr-xr-x 2 youssef youssef 4096 nov. 3 15:35 Documents
drwxr-xr-x 2 youssef youssef 4096 nov. 3 15:56 Images
drwxr-xr-x 2 youssef youssef 4096 nov. 3 15:56 Modèles
drwxr-xr-x 2 youssef youssef 4096 nov. 3 15:56 Musique
drwxr-xr-x 2 youssef youssef 4096 nov. 3 15:35 Public
drwxr-xr-x 2 youssef youssef 4096 nov. 7 00:01 Téléchargements
drwxr-xr-x 2 youssef youssef 4096 nov. 3 15:56 Vidéos
youssef@namenode:~$
```

## Utilisation des options (exemples)

- Dans cet exemple (slide suivant) on a combiné deux options de la commande **ls -l** et **-h** ainsi la commande **ls -l -h** ou **ls -lh** permet de lister le contenu d'un répertoire en fournissant des informations (l'option -l) mais aussi afficher la taille de fichiers en format lisible pour l'homme au lieu de l'afficher en octets
- Cet exemple montre comment on peut combiner les options **ls -l -h = ls -lh = ls -h l**. L'ordre n'est pas important
- L'option **-h** possède une autre forme en mot complet : **--human-readable**
- Cet exemple montre aussi qu'on peut utiliser des arguments et des options en même temps pour une commande

## Utilisation des options (exemples)

```
youssef@namenode:~$ ls -l /usr/bin/perl
-rwxr-xr-x 2 root root 3478464 oct. 19 11:56 /usr/bin/perl
youssef@namenode:~$ ls -lh /usr/bin/perl
-rwxr-xr-x 2 root root 3,4M oct. 19 11:56 /usr/bin/perl
youssef@namenode:~$
```

## L'historique des commandes

- Lorsqu'on exécute une commande sur un terminal celle-ci est enregistrée dans une « liste d'historique des commandes »
- Cela s'avère intéressant lorsqu'on veut exécuter la même commande une autre fois on aura pas à la ré-saisir
- Pour afficher la liste d'historique on utilise la commande **history**

```
youssef@namenode:~$ history
1  cd /home/hadoop/
2  ls
3  /bin/python3 /home/youssef/Bureau/test.py
4  sudo su
5  sudo su
6  ping www.google.com
7  sudo -hadoop
8  sudo hadoop
9  su - hadoop
10 sudo su
11 sudo hadoop
12 su hadoop
13 sudo su
14 su - hadoop
15 ls
16 ls /etc/ppp
17 ls /etc/ppp et /etc/ssh
18 ls /etc/ppp /etc/ssh
19 ls -l
20 ls -l /etc/ppp
21 ls -l /usr/bin/perl
22 ls -lh /usr/bin/perl
23 history
youssef@namenode:~$
```

## L'historique des commandes (exemples)

- On peut naviguer dans l'historique en utilisant les touches « up » et « down » du clavier
- Pour exécuter une commande qui figure dans l'historique on tape dans la ligne de commande un point d'exclamation suivi par son numéro d'ordre : ex **!11**

```
youssef@namenode:~$ !16
ls /etc/ppp
chap-secrets  ip-down.d  ip-up.d    ipv6-down.d  ipv6-up.d  options.pptp  peers
ip-down      ip-up      ipv6-down  ipv6-up      options    pap-secrets
youssef@namenode:~$
```



## Autres options de l'historique

Exemple	signification
history 5	Afficher les 5 dernières commandes dans l'historique
!!	Exécuter la dernière commande une autre fois
!-5	Exécuter la 5 <sup>ème</sup> commande partant de la fin de la liste
!!s	Exécuter la dernière commande ls

## Variables du shell bash

- Une variable permet au Shell de stocker des données qui peuvent être utilisées par le Shell ou par une commande.
- Une variable possède un nom et est stockée temporairement dans la mémoire une fois le terminal ferme toutes les variables créées seront perdues

Exemple : la variable HISTSIZE contient le nombre de commandes mémorisées dans l'historique.

- Pour afficher la valeur d'une variable on utilise la commande **echo**, il faut alors faire précéder le nom de la variable par un **\$**

Exemple: **echo \$HISTSIZE**

- Pour modifier la valeur d'une variable il ne faut pas utiliser le signe \$

Exemple : **HISTSIZE=500**

## Variables du shell bash

```
youssef@namenode:~$ echo $HISTSIZE
1000
youssef@namenode:~$
```

```
youssef@namenode:~$ HISTSIZE=500
youssef@namenode:~$ echo $HISTSIZE
500
youssef@namenode:~$
```

## La variable PATH

- Cette variable indique au Shell les emplacement ou chercher les commandes à exécuter
- Si on tape une commande et on reçoit l'erreur « command not found » c'est parce que le Shell n'arrive pas à trouver la commande saisie dans les chemins indiqués par la variable PATH

```
youssef@namenode:~$ echo $PATH
/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/games:/usr/local/games:/snap/bin
youssef@namenode:~$
```

```
youssef@namenode:~$ cmd
La commande « cmd » n'a pas été trouvée, mais il y en a 17 similaires.
youssef@namenode:~$
```

## La commande which

- La commande **which** permet d'afficher le chemin où est stocké une commande donnée
- La commande **which** cherche dans la variable PATH pour déterminer le chemin d'une commande

```
youssef@namenode:~$ which date
/usr/bin/date
youssef@namenode:~$ which cal
/usr/bin/cal
youssef@namenode:~$
```

## Alias

- Un alias est utilisé pour remplacer une longue commande par un pseudo plus petit
- De cette façon l'utilisateur n'aura pas à taper de longues commandes  
Exemple : la commande `ls -l` a souvent comme alias `ll`
- Pour visualiser les alias définis sur le système on utilise la commande **alias**
- Pour créer de nouveaux alias on utilise la commande

**alias nom=commande**

## Alias (exemple)

- Il faut noter que les alias créés de cette façon ne sont pas permanents et seront perdus une fois le Shell fermé.

The first terminal window shows the creation of several aliases:

```

sysadmin@localhost:~$ alias
[sysadmin@localhost ~]$ alias
alias l='ls -d .* --color=auto'
alias ll='ls -l --color=auto'
alias ls='ls --color=auto'
alias which='alias | /usr/bin/which --tty-only --read-alias --show-dot --show-ti
lde'
[sysadmin@localhost ~]$
  
```

The second terminal window shows the use of these aliases:

```

[sysadmin@localhost ~]$ alias lh='ls -shl'
[sysadmin@localhost ~]$ lh /etc/ppp
total 44K
-rwxr-xr-x. 1 root root 6.4K Apr 27 2012 ip-up.ipv6to4
drwxr-xr-x. 2 root root 4.0K Jun 22 2012 peers
-rwxr-xr-x. 1 root root 3.2K Apr 27 2012 ip-down.ipv6to4
-rwxr-xr-x. 1 root root 3.2K Apr 27 2012 ip-up
-rwxr-xr-x. 1 root root 1.7K Apr 27 2012 ip-down
-rwxr-xr-x. 1 root root 438 Apr 27 2012 ip-up
-rwxr-xr-x. 1 root root 386 Apr 27 2012 ip-down
-rw-r--r--. 1 root root 78 Aug 22 2010 chap-secrets
-rw-r--r--. 1 root root 77 Aug 22 2010 pap-secrets
-rw-r--r--. 1 root root 5 Aug 22 2010 options
[sysadmin@localhost ~]$
  
```

## Le globbing

- Le Globbing permet de référencer plusieurs objets à la fois (fichiers ,répertoires) en utilisant des caractères spéciaux appelés **métacaractères**
- Ces caractères ont une signification particulière pour le Shell et peuvent être utilisés avec n'importe quelle commande
- Le Globbing est un outil puissant permettant d'appliquer une commandes sur un ensemble de fichiers que de traiter fichier par fichier

## Globbering : métacaractère (\*)

- Le caractère **\*** représente 0 ou n'importe quel nombre de caractères dans un nom de fichier

Exemple 1 : Dans cet exemple on affiche tous les fichiers contenus dans /etc et qui commence par la lettre t

```
sysadmin@localhost:~$ echo /etc/t*
/etc/terminfo /etc/timezone
sysadmin@localhost:~$
```

## Globbering : métacaractère (\*)

Exemple 2 : Dans cet exemple on affiche tous les fichiers contenus dans /etc dont le nom se termine par « .d »

```
sysadmin@localhost:~$ echo /etc/*.d
/etc/apparmor.d /etc/bash_completion.d /etc/cron.d /etc/depmod.d /etc/fstab.d
/etc/init.d /etc/insserv.conf.d /etc/ld.so.conf.d /etc/logrotate.d /etc/modprobe
/etc/pam.d /etc/profile.d /etc/rc0.d /etc/rc1.d /etc/rc2.d /etc/rc3.d /etc/rc
d /etc/rc5.d /etc/rc6.d /etc/rcS.d /etc/rsyslog.d /etc/sudoers.d /etc/sysctl.d
/etc/update-motd.d
```

Exemple 3 : Dans cet exemple on affiche tous les fichiers contenus dans /etc et qui débutant par « r » et se termine par « .conf »

```
sysadmin@localhost:~$ echo /etc/r*.conf
/etc/resolv.conf /etc/rsyslog.conf
sysadmin@localhost:~$
```

## Globbering : métacaractère ( ? )

- Le « ? » représente un seul et unique caractère dans un nom de fichiers

Exemple 1 : Dans cet exemple on affiche les fichiers du répertoire /etc dont les noms contiennent exactement 8 caractères et commence par « t »

```
sysadmin@localhost:~$ echo /etc/t????????
/etc/terminfo /etc/timezone
sysadmin@localhost:~$
```

## Globbering : métacaractère ( ? )

Exemple 2 : Dans cet exemple on affiche tous les fichiers ayant dans leurs noms 20 au plus caractères

```
sysadmin@localhost:~$ echo /etc/*????????????????????????????
/etc/bindresvport.blacklist /etc/ca-certificates.conf
sysadmin@localhost:~$
```

Exemple 3 : On affiche tous les fichiers ayant dans leurs noms une extension de 3 caractères exactement

```
sysadmin@localhost:~$ echo /etc/*.???
/etc/blkid.tab /etc/issue.net
sysadmin@localhost:~$
```

## Globbering : les crochets « [ ] »

- Les crochets **[ ]** sont utilisés pour représenter un seul caractère défini dans une liste de caractères mises entre les crochets

Exemple 1 : Dans cet exemple on souhaite afficher tous les fichiers commençant par un « g » ou un « u » et se terminent par 0 ou plus de caractères.

```
sysadmin@localhost:~$ echo /etc/[gu]*
/etc/gai.conf /etc/groff /etc/group /etc/group- /etc/gshadow /etc/gshadow- /etc/ucf.conf /etc/udev /etc/ufw /etc/update-motd.d /etc/updatedb.conf
sysadmin@localhost:~$
```

## Globbering : les crochets « [ ] »

Exemple 2 : Dans cet exemple les crochets sont utilisés pour représenter un caractère dans un intervalle entre deux caractères, ici on veut afficher tous les fichiers qui commencent par un caractère compris entre « a » et « d » (a,b,c ou d) et se terminent par une chaîne de caractères de longueur quelconque

```
sysadmin@localhost:~$ echo /etc/[a-d]*
/etc/adduser.conf /etc/alternatives /etc/apparmor.d /etc/apt /etc/bash.bashrc /etc/bash_completion.d /etc/bind /etc/bindresvport.blacklist /etc/blkid.conf /etc/blkid.tab /etc/ca-certificates /etc/ca-certificates.conf /etc/calendar /etc/cole-setup /etc/cron.d /etc/cron.daily /etc/cron.hourly /etc/cron.monthly /etc/cron.weekly /etc/crontab /etc/dbus-1 /etc/debconf.conf /etc/debian_version /etc/default /etc/deluser.conf /etc/depmod.d /etc/dhcp /etc/dpkg
sysadmin@localhost:~$
```

## Globbering : les crochets « [ ] »

Exemple 3 : On souhaite afficher les fichiers dont le nom contient au moins un nombre

```
sysadmin@localhost:~$ echo /etc/*[0-9]*  
/etc/X11 /etc/dbus-1 /etc/iproute2 /etc/mke2fs.conf /etc/python3 /etc/python3.2  
/etc/rc0.d /etc/rc1.d /etc/rc2.d /etc/rc3.d /etc/rc4.d /etc/rc5.d /etc/rc6.d  
sysadmin@localhost:~$
```

Il faut noter que l'ordre des caractères entre les crochets est importants si on met `echo /etc/*[9-0]*` aucun résultat ne sera affiché

## Globbering : métacaractère « ! »

- Le point d'exclamation est utilisé avec les crochets pour faire la négation d'une liste de caractères
  - Exemple : `echo /etc/[!DP]*`
  - Permet d'afficher tous les fichiers qui ne commencent pas par un « D » ou un « P »



# Annuler l'interprétation des métacaractères par le shell (quoting)

- Pour annuler l'interprétation des métacaractères par le Shell on utilise trois quotes.
- Les guillemets « " " »
- L'apostrophe « ' ' »
- L'apostrophe inversée « ` ` »

## Les guillemets « " " »

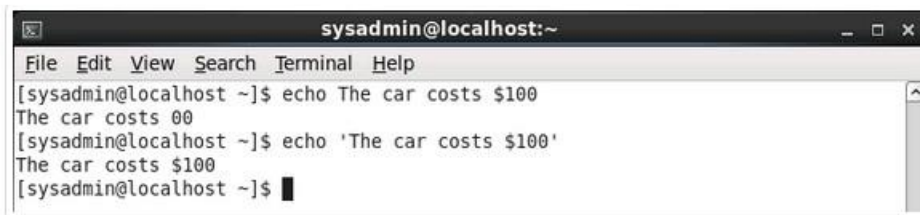
- Le Shell n'interprète pas les métacaractères se trouvant entre guillemets ainsi un « \* » ou un « ? » n'a pas d'effet s'il est placé entre " ".



```
sysadmin@localhost:~  
File Edit View Search Terminal Help  
[sysadmin@localhost ~]$ echo /etc/[DP]*  
/etc/DIR_COLORS /etc/DIR_COLORS.256color /etc/DIR_COLORS.lightbgcolor /etc/Packa  
geKit  
[sysadmin@localhost ~]$ echo "/etc/[DP]*"  
/etc/[DP]*  
[sysadmin@localhost ~]$
```

## L'apostrophe « ' ' »

- Les apostrophes empêchent le Shell d'interpréter les métacaractères , les commandes , les variables et d'autres caractères spéciaux qu'on a pas encore vu

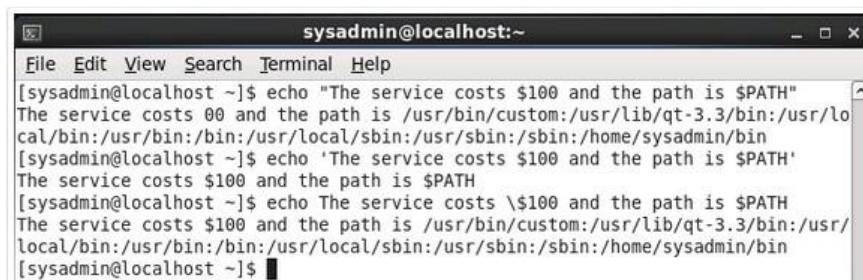


```

sysadmin@localhost:~
File Edit View Search Terminal Help
[sysadmin@localhost ~]$ echo The car costs $100
The car costs 00
[sysadmin@localhost ~]$ echo 'The car costs $100'
The car costs $100
[sysadmin@localhost ~]$
  
```

## L'anti slash « \ »

- L'anti slash permet d'annuler l'effet d'un seul caractère spécial , en plaçant l'anti slash devant celui-ci.



```

sysadmin@localhost:~
File Edit View Search Terminal Help
[sysadmin@localhost ~]$ echo "The service costs $100 and the path is $PATH"
The service costs 00 and the path is /usr/bin/custom:/usr/lib/qt-3.3/bin:/usr/local/bin:/usr/bin:/bin:/usr/local/sbin:/usr/sbin:/sbin:/home/sysadmin/bin
[sysadmin@localhost ~]$ echo 'The service costs $100 and the path is $PATH'
The service costs $100 and the path is $PATH
[sysadmin@localhost ~]$ echo The service costs \$100 and the path is $PATH
The service costs $100 and the path is /usr/bin/custom:/usr/lib/qt-3.3/bin:/usr/local/bin:/usr/bin:/bin:/usr/local/sbin:/usr/sbin:/sbin:/home/sysadmin/bin
[sysadmin@localhost ~]$
  
```

## L'apostrophe inversée « ` ` »

- Permet d'exécuter une commande et de faire passer son résultat à une autre commande, on appelle cela substitution de commande

Exemple : pour commencer exécutant la commande date



```

sysadmin@localhost:~
File Edit View Search Terminal Help
[sysadmin@localhost ~]$ date
Sun Sep 15 18:23:04 CDT 2013
[sysadmin@localhost ~]$
  
```

## L'apostrophe inversée « ` ` »

- À présent on exécute la commande echo today is date



```

sysadmin@localhost:~
File Edit View Search Terminal Help
[sysadmin@localhost ~]$ echo Today is date
Today is date
[sysadmin@localhost ~]$
  
```

- On remarque que le Shell traite le mot date comme un texte régulier et non pas comme une commande
- On souhaite à présent que date soit traité comme une commande il faut alors la placer entre ` `



```

sysadmin@localhost:~
File Edit View Search Terminal Help
[sysadmin@localhost ~]$ echo Today is `date`
Today is Sun Sep 15 18:28:11 CDT 2013
[sysadmin@localhost ~]$
  
```

## L'apostrophe inversée « ` ` »

- L'enchaînement des commandes permet d'exécuter des commandes multiples à la suite d'une commande mais cette exécution dépend du résultat de la première commande (échec ou réussite)
- Il existe trois types d'enchaînement utilisant les symboles suivants :
  - Le point virgule (;)
  - &&
  - ||

## Le point virgule (;)

- Permet d'exécuter plusieurs commandes l'une à la suite de l'autre d'une façon indépendante
- Le résultat d'une commande ne va pas influencer sur l'exécution de l'autre

```

sysadmin@localhost:~$ cal 1 2014; cal 2 2014; cal 3 2014
January 2014
Su Mo Tu We Th Fr Sa
                1  2  3  4
 5  6  7  8  9 10 11
12 13 14 15 16 17 18
19 20 21 22 23 24 25
26 27 28 29 30 31

February 2014
Su Mo Tu We Th Fr Sa
                1
 2  3  4  5  6  7  8
 9 10 11 12 13 14 15
16 17 18 19 20 21 22
23 24 25 26 27 28

March 2014
Su Mo Tu We Th Fr Sa
                1
 2  3  4  5  6  7  8
 9 10 11 12 13 14 15
16 17 18 19 20 21 22
23 24 25 26 27 28 29
30 31
sysadmin@localhost ~$
  
```

## Le (&&)

- Il agit comme un « AND » logique
- Si la première commande a réussi alors la deuxième sera exécutée si non il ne sera pas exécutée

```
sysadmin@localhost:~$ ls /etc/xml
catalog catalog.old xml-core.xml xml-core.xml.old
sysadmin@localhost:~$ ls /etc/junk
ls: cannot access /etc/junk: No such file or directory
sysadmin@localhost:~$
```

```
sysadmin@localhost:~$ ls /etc/xml && echo success
catalog catalog.old xml-core.xml xml-core.xml.old
success
sysadmin@localhost:~$ ls /etc/junk && echo success
ls: cannot access /etc/junk: No such file or directory
sysadmin@localhost:~$
```

## Le (||)

- Il agit comme un OU logique
- Si la première commande échoue la deuxième sera exécutée
- Si la première commande réussit la deuxième ne sera pas exécutée

```
sysadmin@localhost:~$ ls /etc/xml || echo failed
catalog catalog.old xml-core.xml xml-core.xml.old
sysadmin@localhost:~$ ls /etc/junk || echo failed
ls: cannot access /etc/junk: No such file or directory
failed
sysadmin@localhost:~$
```

## Utilisation des sources d'aide

---

- Le Shell Linux offre plusieurs sources d'aides permettant de comprendre le comportement des commandes et aussi de découvrir de nouvelles commandes.
- On va découvrir les sources d'aides suivantes :
  - Les pages **man**
  - L'option **--help**
  - Les commandes **whatis** et **whereis**

## Les pages man

---

- Le manuel(man) permet de donner une description détaillée d'une commande avec toutes ses options
- Pour afficher les pages man d'une commande il faut taper : man commande
- Exemple : **man cal** permet d'afficher les pages du manuel concernant la commande cal

## Les pages man

```

sysadmin@localhost:~
File Edit View Search Terminal Help
CAL(1) BSD General Commands Manual CAL(1)

NAME
  cal - displays a calendar

SYNOPSIS
  cal [-smjy13] [[[day] month] year]

DESCRIPTION
  cal displays a simple calendar. If arguments are not specified, the cur-
  rent month is displayed. The options are as follows:

  -1    Display single month output. (This is the default.)
  -3    Display prev/current/next month output.
  -s    Display Sunday as the first day of the week.
  -m    Display Monday as the first day of the week.
  -j    Display Julian dates (days one-based, numbered from January 1).
  -y    Display a calendar for the current year.

```

25/11/2020

DR. MOURDI YOUSSEF

45

## Les pages man

- Le tableau suivant indique les raccourcis clavier permettant de naviguer à travers une page du manuel (man)

Commande /touche clavier	rôle
entrée	Afficher la ligne suivante
espace	Afficher la page suivante
/terme	Chercher un terme
n	Trouver le prochain terme recherché
1G	Aller au début du manuel
G	Aller à la fin du manuel
h	Afficher l'aide
q	Quitter le manuel

25/11/2020

DR. MOURDI YOUSSEF

46

## Les sections d'une page man

- Les pages man sont subdiviser en section chaque section donne une description de la commande le tableau suivant résume les principales sections qu'on peut trouver dans une page man

Section	Informations fournies
NAME	Donne le nom de la commande et une très brève description
SYNOPSIS	Donne la syntaxe de la commande
DESCRIPTION	Donne plus de détaille sur la commande
OPTIONS	Donne la description de chacune des options de la commande, parfois ces informations figurent dans la section Description et non dans la section Options
FILES	Donne l'ensemble des fichiers associés à la commande et qui permettent de configurer la commande
AUTHOR	Le nom et parfois le contact de la personne qui a crée la page man
REPORTING BUGS	Comment reporter les bugs relatifs à la commande
COPYRIGHT	Information sur le copyright (droit d'auteur)
SEE ALSO	Où chercher d'autres informations ou quelles sont les commandes associées à la commande en question

## La section SYNOPSIS du man

- La section SYNOPSIS est très importante car elle donne une brève description sur comment utiliser la commande

Exemple : soit la section SYNOPSIS de la commande cal

```
SYNOPSIS
cal [-smjy13] [[[day] month] year]
```

- On peut utiliser la commande cal avec les options suivantes : -s, -m, -j, -y, -1 et -3
- Les crochets [ ] indiquent que ces options ne sont pas obligatoires pour exécuter la commande
- Il existe d'autres options pour la commande cal: day, month et year
- On peut utiliser l'option year seule mais si j'utilise l'option month il faut obligatoirement spécifier l'année(year)
- La même chose si j'utilise l'option day



## L'option --help

- Plusieurs commandes possèdent une option **--help** permettant d'afficher une aide sommaire sur la commande et ses options les informations affichées

```

sysadmin@localhost:~$ ps --help
***** simple selection *****
-A all processes
-N negate selection
-a all w/ tty except session leaders
-d all except session leaders
-e all processes
-T all processes on this terminal
-a all w/ tty, including other users
-g OBSOLETE -- DO NOT USE
-r only running processes
-x processes w/o controlling ttys
***** output format *****
-o,o user-defined -f full
-j,j job control -s signal
-O,O preloaded -o v virtual memory
-l,l long -u user-oriented
-F extra full -x registers
***** selection by list *****
-C by command name
-G by real group ID (supports names)
-U by real user ID (supports names)
-g by session OR by effective group name
-p by process ID
-s processes in the sessions given
-t by tty
-u by effective user ID (supports names)
-U processes for specified users
-t by tty
***** long options *****
--Group --User --pid --cols --ppid
--group --user --sid --rows --info
--cumulative --format --deselect
--sort --tty --forest --version
--heading --no-heading --context
***** misc options *****
-L list format codes -f ASCII art forest
-m,m,-L,-T,H threads -s children in sum -y change -l format
-M,Z security data -c true command name -c scheduling class
-w,w wide output -n numeric WCHAN,UID -H process hierarchy
[sysadmin@localhost ~]$
  
```

## La commande whatis

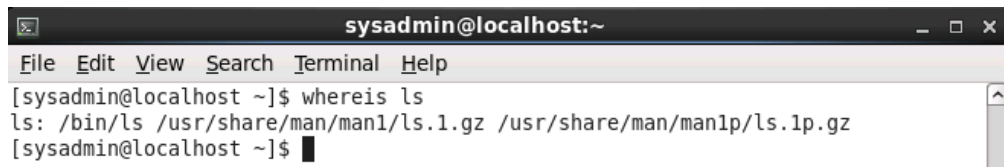
- La commande **whatis** permet de chercher et d'afficher les pages du manuel relatifs à un mot clé avec une description courte de chaque page

```

sysadmin@localhost:~$ whatis ls
ls          (1) - list directory contents
ls          (1p) - list directory contents
[sysadmin@localhost ~]$
  
```

## La commande whereis

- La commande **whereis** permet d'afficher l'emplacement des commandes et des pages man relatives à une commande



```
sysadmin@localhost:~  
File Edit View Search Terminal Help  
[sysadmin@localhost ~]$ whereis ls  
ls: /bin/ls /usr/share/man/man1/ls.1.gz /usr/share/man/man1p/ls.1p.gz  
[sysadmin@localhost ~]$
```