Les principales commandes linux

SOMMAIRE :

LISTE DES COMMANDES :

I. AIDE EN LIGNE

1 man

a. Syntaxe :

```
man [section] commande
man -k mot_clé
```

b. Description :

man permet de rechercher une aide sur une commande ou un mot-clé. Il utilise la variable `MANPATH` pour effectuer la recherche des pages et la variable `PAGER` pour connaître le programme chargé de l'affichage.

Les pages man sont organisées en 8 sections standard comme suit :

1 = Commandes utilisateur 1M = Commandes administrateur
2 = Appels systèmes C.
3 = Fonctions C.
4 = Format des fichiers système.
5 = Divers.
6 = Jeux.
7 = Fichiers spéciaux.
8 = Procédures de maintenance système.

c. Options courantes:

`section` Numéro de la section qui contient la page.
`-k` précise que la recherche s'effectue sur un mot-clé et non une commande.

d. Exemples :

```
$ man passwd
…
$ man 4 passwd
…
```

```
FoxSIN:~# man -k passwd
chgpasswd (8)         - update group passwords in batch mode
chpasswd (8)          - update passwords in batch mode
gpasswd (1)           - administer the /etc/group and /etc/gshadow files
pam_localuser (8)     - require users to be listed in /etc/passwd
passwd (1)            - change user password
passwd (5)            - the password file
update-passwd (8)     - safely update /etc/passwd, /etc/shadow and
/etc/group
```

e. Remarques :

Les pages man sont toujours articulées autour des paragraphes suivants :

NOM SYNTAXE
DESCRIPTION OPTIONS
FICHIERS UTILISES VOIR
AUSSI
Sous Linux, la description des fichiers se trouve dans la section 5

II. GESTION ET DÉPLACEMENT DE RÉPERTOIRES

- **ls** liste le contenu d'un répertoire.
- **cd** changement de répertoire courant.
- **pwd** affichage du répertoire courant.
- **mkdir** création de répertoire.
- **rmdir** destruction d'un répertoire.

2 ls

a. Syntaxe :

```
ls [options] [noms]
```

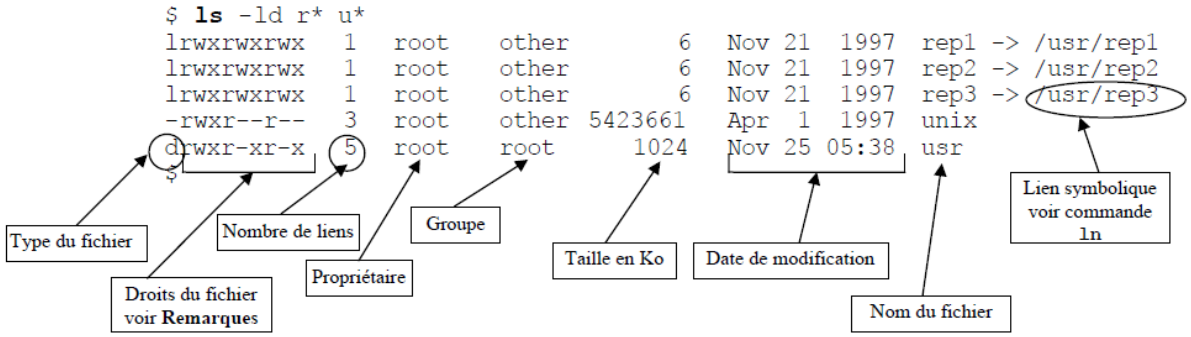
b. Description :

ls liste les répertoires et les fichiers précisés dans noms. Par défaut, la sortie est envoyée à l'écran par ordre alphabétique. Les options déterminent les informations à afficher et la présentation de l'affichage. Sans options, **ls** n'envoie que le nom des fichiers. Si noms n'est pas précisé, c'est le répertoire courant qui est listé.

c. Options courantes:

- **-R** Traitement récursif
- **-a** Tous les fichiers (y compris ceux qui commencent par un point) **-d** Affiche le nom des répertoires sans leur contenu
- **-l** Format long (avec beaucoup de détails)
- nom est le nom d'un fichier ou d'un répertoire (avec ou sans métacaractères)

d. Exemple :



e. Remarques :

Au niveau du propriétaire, du groupe, et des autres, il est possible de déterminer un certain nombre de droits :

- **La lecture (R)** : pour un fichier, ce droit permet la lecture du fichier; alors que pour un répertoire, il autorise l'utilisateur à lister son contenu.
- **L'écriture (W)** : pour un fichier, il permet sa modification; alors que pour un répertoire, il permet la création et la suppression des fichiers du répertoire (**ATTENTION** : cette permission est valable quels que soient les droits des fichiers).
- **L'exécution (X)** : pour un fichier, il autorise son exécution; et pour un répertoire, il permet de se positionner dessous

3 cd

f. Syntaxe :

```
cd [répertoire]
```

g. Description :

La commande **cd** permet de changer le répertoire de travail. Si `répertoire` n'est pas précisé, alors le nouveau répertoire de travail sera le répertoire de connexion ($HOME).

h. Option :

`répertoire` représente le futur répertoire de travail.

i. Exemples :

```
$ cd /usr/lib/news/bin
$ cd $HOME/rep1
```

j. Remarque :

La commande **cd**, comme toutes les commandes utilisant des répertoires, permet de spécifier deux types de chemins :

- les chemins relatifs : ils sont relatifs au répertoire de travail, et utilisent notamment le répertoire '`..`' (répertoire père).

```
Exemples : cd ../rep1
```

- les chemins absolus : ils faut spécifier toute l'arborescence depuis la racine.

```
Exemples : cd /home/rep1
```

4 pwd

k. Syntaxe :

```
pwd
```

l. Description :

La commande **pwd** permet d'afficher le répertoire de travail.

m. Exemples :

```
$ pwd
/usr/lib/news/bin
$ cd ..; pwd
/usr/lib/news
$ cd bin
$ pwd
/usr/lib/news/bin
$ cd /bin
$ pwd
/bin
```

5 mkdir

n. Syntaxe :

```
mkdir [-p]  nouveau_répertoire
```

o. Description :

La commande **mkdir** créé le répertoire spécifié sur la ligne de commande (`nouveau_répertoire`). Si l'un des répertoires intermédiaires n'existe pas, la commande retourne un code d'erreur (*exit status*) sans créer le répertoire (sauf si l'option `-p` est spécifiée).

p. Options courantes

- -p permet de créer tous les répertoires intermédiaires qui n'existeraient pas.
- répertoire représente le nom du répertoire à créer. C'est un argument obligatoire

q. Exemples :

```
$       mkdir /tmp/rep1
$       cd /tmp/rep1
$       mkdir rep11/rep111
mkdir:  Cannot  create directory " rep11/rep111": No such file or directory

$ mkdir -p rep11/rep111
```

r. Remarque :

Pour pouvoir créer un répertoire, le répertoire d'origine doit avoir les droits en écriture positionnés.

6 rmdir

s. Syntaxe :

```
rmdir [-p] [-s] répertoire
```

t. Description :

La commande **rmdir** supprime le répertoire spécifié sur la ligne de commande (répertoire). Si il existe des fichiers ou des sous répertoires, la commande retournera un code d'erreur (*exit status*).

u. Options courantes

- -p permet de détruire tous les sous-répertoires vides.
- -s mode silencieux (aucun affichage).
- répertoire représente le nom du répertoire à détruire. C'est un argument obligatoire.

v. Exemples :

```
$ rmdir /tmp/rep1/rep11/rep111 $ cd /tmp
$ rmdir -p rep1/rep11
rmdir: rep1/rep11: Whole path removed. $ cd rep1
rep1:  does not  exist
```

w. Remarque :

Pour pouvoir supprimer un répertoire, le répertoire père doit avoir les droits en écriture positionnés.

Voir aussi la commande **rm** -r pour supprimer des répertoires contenants des fichiers.

III. GESTION ET MANIPULATION DE FICHIERS

- cp copie de fichiers.
- mv déplacement de fichiers.
- rm destruction de fichiers.
- cat visualisation et/ou concaténation de fichiers.
- Pg | less visualisation d'un fichier texte page par page.
- chmod change les droits d'un fichier/répertoire.
- chown change le propriétaire d'un fichier/répertoire.
- chgrp change le groupe propriétaire du fichier/répertoire.
- find recherche de fichiers ou répertoires.
- grep recherche d'une chaîne de caractères dans un fichier.
- head/tail affiche le début/la fin d'un fichier
- ln crée un lien avec un fichier existant
- sort trie les lignes d'un fichier
- umask choix des permissions par défaut.
- wc compte le nombre de mots/lignes/caractères d'un fichier.

7 cp

a. Syntaxe :

```
cp [-i] [-p] fichier1 fichier2
```

```
cp [-i] [-p] [-r] source1 [source2...] répertoire
```

b. Description :

La commande **cp** copie le contenu de `fichier1` dans `fichier2` ; ou bien elle copie `source1` et `source2` (etc...) dans `répertoire`.

c. Options courantes

- `-i` mode interactif, demande la confirmation avant écrasement.
- `-p` conserve les dates du fichier source.
- `-r` copie récursive de répertoires.
- `source` *X* représente le nom des fichiers ou répertoires à copier.

d. Exemples :

```
$ cp /tmp/rep1/fic1
$ cp /tmp $HOME
$ cp -r rep1 rep2
```

e. Remarque :

Pour pouvoir copier un fichier/répertoire, vous devez avoir les droits suivants :

- droits de lecture du fichier à copier ;
- droits d'exécution sur le répertoire contenant le fichier à copier ;
- droits d'écriture sur le répertoire de destination.

8 mv

f. Syntaxe :

```
mv [-f] [-i] source1 [source2...] destination
```

g. Description :

La commande **mv** déplace les fichiers `fichier1, fichier2 etc...` dans `destination` .
Si `destination` est un fichier, alors **mv** a pour action de renommer `fichier1` en `destination` ; si `destination` est un répertoire, alors **mv** déplace `fichier1` dans ce répertoire.

h. Options courantes

- `-i` mode interactif, demande la confirmation avant écrasement
- `-f` force la commande.
- `sourceX` représente le nom des fichiers ou répertoires a déplacer.
- `destination` représente le nom des fichiers ou répertoires de destination

i. Exemples :

```
$ mv fic1 fic2
```

j. Remarque :

Pour pouvoir copier un fichier/répertoire, le répertoire cible doit avoir les droits en écriture positionnés, les droits en lecture sur le fichier source, et les droits d'accès dans le répertoire source.

9 rm

k. Syntaxe :

```
rm [-f] [-i] [-r] fichier1 [fichier2...]
```

l. Description :

La commande **rm** supprime les fichiers spécifiés sur la ligne de commande. Si vous n'avez pas les droits d'écriture sur `fichier1` , alors **rm** vous demandera de confirmer votre action ; la réponse oui (y) détruira quand même le fichier (sous réserve d'avoir les droits d'écriture sur le répertoire).

m. Options courantes

- -i mode interactif, demande la confirmation avant chaque
- -f suppression force la commande (aucune confirmation).
- -r récursif (détruit tous les sous répertoires. ATTENTION)

n. Exemples :

```
$ rm fic1
$ rm -i fic*
rm: File fic1. Remove ( yes/no)? y rm: File fic2. Remove ( yes/no)?
N
```

o. Remarque :

Pour pouvoir supprimer un fichier, le répertoire où se trouve le fichier doit avoir ses droits en écriture positionnés ; sauf dans le cas où le répertoire aurait les droit suivants (sticky bit):

```
drwxrwxrwt       1    user     user        6    Nov 21   1997   rep1
```

10 cat

p. Syntaxe :

```
cat [fichier...]
```

q. Description :

La commande **cat** visualise et/ou concatène les fichiers spécifiés sur la ligne de commande.
Par défaut, **cat** lit sur l'entrée standard et affiche le résultat sur la sortie standard.

r. Exemples :

```
$ cat villes
Sarrebourg
Douai
Couvron
Phalsbourg
$ cat villes pays
Sarrebourg
Douai
Couvron
Phalsbourg
France
Belgique
Italie
```

11 pg

s. Syntaxe :

```
pg [+numlig] [+/chaîne/] [fichier...]
```

t. Description :

La commande **pg** affiche à l'écran les fichiers spécifiés sur la ligne de commande.
Par défaut, **pg** lit sur l'entrée standard ce qui permet de l'associer à un pipe.

u. Options courantes

- `+numlig` spécifie le numéro de ligne où doit commencer
- `+/chaîne/` l'affichage commence l'affichage à la première ligne du fichier contenant `chaine`

v. Exemples :

```
$ pg villes
Sarrebourg
Douai
Couvron
(EOF):
$ pg +2 villes
Douai
Couvron
(EOF):
$ cat villes pays| pg +/Couvron/ pays
Couvron
Phalsbourg
France
…
(EOF)
```

12 chmod

w. Syntaxe :

```
chmod [-R] mode nom [...]
```

```
chmod [-R] [ ugoa]{+|-|=}[ rwx] nom [...]
```

x. Description :

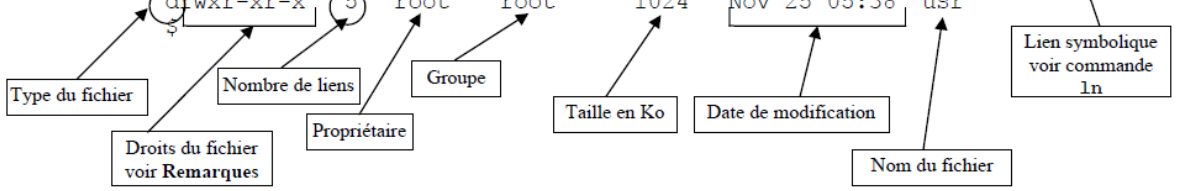
La commande **chmod** change les droits d'accès aux fichiers spécifiés sur la ligne de commande. Deux manières d'attribuer les droits sont possibles. La première stipule les droits de manière numérique par un calcul des différentes valeurs associées aux droits; la seconde permet de spécifier ces mêmes droits de manière plus symbolique.

y. Options courantes

- `-R` récursif sur tous les fichiers et sous-répertoires contenus si `nom` est un répertoire
- `mode` c'est la combinaison des droits numériques (voir **Remarques**)
- `ugoa` caractère spécifiant le champ d'application des modifications :
  - `u` représente l'utilisateur,
  - `g` le groupe,
  - `o` les autres,
  - `a` regroupe tous ces derniers.
- `+|-|=` indique l'action à accomplir respectivement l'ajout du droit, son retrait, ou bien son affectation.
- `rwx` indique le droit proprement dit (`r` lecture, `w` écriture, `x` exécution).

z. Exemple

```
$ ls -ld r* u*
lrwxrwxrwx   1   root    other        6   Nov 21   1997  rep1 -> /usr/rep1
lrwxrwxrwx   1   root    other        6   Nov 21   1997  rep2 -> /usr/rep2
lrwxrwxrwx   1   root    other        6   Nov 21   1997  rep3 -> /usr/rep3
-rwxr--r--   3   root    other  5423661   Apr  1   1997  unix
drwxr-xr-x   5   root    root      1024   Nov 25 05:38  usr
```



13 chown chgrp

aa. Syntaxe :

```
chown [-R] [-h] utilisateur nom [...]
chgrp [-R] [-h] groupe nom [...]
```

bb. Description :

La commande **chown** change le propriétaire des fichiers spécifiés sur la ligne de commande.
La commande **chgrp** change le groupe des fichiers spécifiés sur la ligne de commande.

cc. Options courantes

`-R`	récursif sur tous les fichiers et sous-répertoires contenus
`-h`	si `nom` est un répertoire

	traitement sur les liens symboliques
`nom`	exprime le nom d'un fichier ou d'un répertoire
`utilisateur`	représente soit le nom de l'utilisateur, soit son UID (User IDentification)
`groupe`	représente soit le nom du groupe, soit son GID (Group IDentification)

dd. Exemple :

```
$ chown root villes $ chown 102 villes $ chgrp news villes
```

ee. Remarque :

Seul le propriétaire des fichiers traités ou **root** ont le droit d'utiliser `chown` et `chgrp` .

14 find

ff. Syntaxe :

```
find répertoire option1 [option2...]
```

gg. Description :

La commande **`find`** permet de rechercher un fichier dans l'arborescence à partir du point spécifié.

hh. Options courantes:

`-name` `fich`	recherche sur le nom `fich`
`-size` `n`	recherche sur la taille en blocs
`-ctime` `n`	recherche sur la date de création
`-exec` `cmd {} \;`	exécute la commande `cmd` sur les fichiers trouvés
`-print`	affiche le résultat de la recherche
`!`	négation du critère de recherche

ii. Exemples :

```
$ find $HOME - name "vil*" -print
$ find . -print| cpio -ocvB / dev/streamer
$ find / - name "profile*" - exec pg {} \;
```

Représente le nom des fichiers trouvés.

jj. Remarque :

Il est nécessaire de faire suivre l'option **`-exec`** par `{} \;`

Métacaractère	Utilisation
?	indétermination d'un caractère
*	indétermination de **0** à **n** caractères
[xyz123...]	indétermination d'un caractère pris

	dans la liste
[b-t]	indétermination d'un caractère pris dans l'intervalle
[!xyz123...]	indétermination d'un caractère pris à l'extérieur de la liste
[!b-t]	indétermination d'un caractère pris à l'extérieur de l'intervalle

15 grep

kk. Syntaxe :

```
grep [-ilsfv] expression [fichier...]
```

ll. Description :

La commande **grep** permet de rechercher `expression` dans `fichier`. Elle affiche les noms de fichiers ainsi que les lignes contenant `expression` .

mm. Options courantes

`-i`	ne tient pas compte des minuscules et des MAJUSCULES
`-l`	n'affiche que le nom des fichiers (pas les lignes)
`-s`	pas de message d'erreur sur les fichiers inaccessibles
`-f fich`	spécifie un fichier contenant les expressions à rechercher
`-v`	affiche toutes les lignes, sauf celles qui contiennent l'expression
`expression`	chaîne de caractères (ou expression régulière non abordée dans ce cours)
`fichier`	nom des fichiers à traiter

nn. Exemples :

```
$ grep Sarrebourg *
villes : Sarrebourg
$ grep -l Sarrebourg *
villes
```

16 Head et tail

oo. Syntaxe :

```
head [-n] [fichier...]
```

```
tail [-n|+n] [-f] [fichier]
```

pp. Description :

La commande **head** affiche les `n` premières lignes d'un fichier, alors que **tail** affiche les dernières lignes d'un fichier.
Si `n` n'est pas précisé, il prend la valeur `10`.

qq. Options courantes

`-n`	nombre de lignes à afficher depuis le début/la fin de `fichier`
`+n`	affichage à partir de la ligne numéro `n`
`-f`	attente de nouvelles lignes (sortie par *`Ctrl-c`*)
`fichier`	nom des fichiers à traiter

rr. Exemples :

```
$ head -2 villes Sarrebourg
Douai
$ tail -2 villes Couvron Phalsbourg
$ tail +2 villes Douai

Couvron Phalsbourg
```

17 ln

ss. Syntaxe :

```
ln [-s] fichier1 fichier2
ln [-s] fichier1 [fichier2...] répertoire
```

tt. Description :

La commande **ln** permet de créer des entrées multiples dans l'arborescence d'un système de fichiers pour un même fichier physique.
Ce qui revient à dire que si l'on modifie un fichier, ses liens le sont aussi. **ln** permet aussi de faire des liens dans des systèmes de fichiers différents par la méthode des liens symboliques (un peu comme les raccourcis de chez MS).
Si le dernier argument de la ligne de commande est un répertoire, **ln** crée des liens dans ce répertoire pour tous les fichiers pré-cités (`fichier1, fichier2, ...`).

uu. Options

`-s` permet de faire un lien symbolique

vv. Exemples :

```
$       ln villes villes5
$       ls villes*
```

```
-rwxr--r--  2 root   other  5423661Apr    1      1997   villes
-rwxr--r--  2 root   other  5423661Apr    1      1997   villes5
$ ln -s villes villes10
$      ls villes*
-rwxr--r--  2 root   other  5423661Apr    1      1997   villes
-rwxr--r--  2 root   other  5423661Apr    1      1997   villes5
lrwxr--r--  1 root   other  5423661Apr    1      1997   villes10  ->
villes
```

ww. Remarque :

Les liens peuvent aussi concerner des répertoires (dans ce cas, ce seront toujours des liens symboliques).

18 sort

xx. Syntaxe :

```
sort [-ufnr] [-o fic] [fichier...]
```

yy. Description :

La commande **`sort`** trie les lignes des fichiers en arguments et affiche le résultat à l'écran. Le clavier est lu si `fichier` est omis.
Par défaut **`sort`** effectue un tri par ordre alphabétique; mais les options suivantes en modifient les critères.

zz. Options courantes

`-u`	permet de n'afficher qu'une seule fois les lignes multiples
`-f`	ne différencie pas les minuscules et MAJUSCULES
`-n` `-r`	effectue un tri numérique ordre décroissant
`-o fic`	enregistre la sortie dans `fic`

aaa. Exemples :

```
$ sort villes Couvron
Douai Phalsbourg Sarrebourg
$ sort -r villes Sarrebourg Phalsbourg
Douai
Couvron
```

19 umask

bbb. Syntaxe :

```
umask [???]
```

ccc. Description :

La commande umask permet de définir les droits affectés par défaut aux fichiers lors de leur création. Si le masque ??? est omis, alors umask affiche le masque en cours.

ddd. Options courantes

??? : chaque ? représente une valeur entre 0 et 7 qui est le complément à 7 des droits à affecter aux fichiers. Si l'on veut avoir des fichiers avec 751 (rwxr-x--x) comme droits, il faudra définir comme masque 026 (on remarque que 7+0 = 5+ 2 = 1+ 6 = 7).

eee. Exemples :

```
$ umask 022
$ umask
022
```

Vos fichiers seront donc créés avec les droits 755

fff. Remarque :

Il faut noter que les droits affectés à la création d'un fichier dépendent aussi de l'utilitaire qui les a créés; si vous avez un masque 000 et que vous créez un fichier avec vi, les droits effectifs de votre fichier sont 666 (rw-rw-rw-) car vi est un éditeur de texte et non de programmes shells.

À l'inverse, quel que soit le masque utilisé, le compilateur c cc (programme permettant de créer des fichiers programmes) positionnera toujours les droits d'exécution sur les fichiers qu'il crée.

20 wc

ggg. Syntaxe :

```
wc [-lwc] [fichier...]
```

hhh. Description :

La commande **wc** compte le nombre de lignes, mots, ou caractères d'un fichier texte
Si aucun fichier n'est passé en paramètre, c'est l'entrée standard qui sera lue. Si aucune option (-

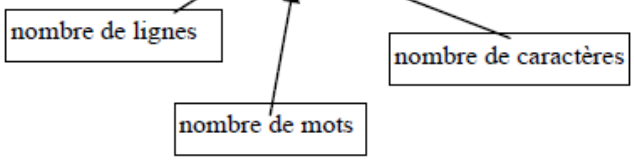
lwc) n'est précisée, alors **wc** compte le nombre de lignes, mots, **et** caractères du fichier.

iii. Options courantes

`-l` : précise que c'est le nombre de lignes qui doit être compté
`-c` : précise que c'est le nombre de mots qui doit être compté
-w : précise que c'est le nombre de caractères qui doit être compté

jjj. Exemple

```
$ wc -l villes
4 villes
$ wc villes
 4  4 36 villes
```



IV. COMMANDES ORIENTÉES SHELL

⇒ **echo**	affichage de texte sur la sortie standard	
⇒ **expr**	évaluation	d'expressions numériques
⇒ **test**	évaluation	d'expressions diverses
⇒ **clear**	efface l'écran	

21 echo

a. Syntaxe :

```
echo [-n] message
```

b. Description :

La commande **echo** affiche sur la sortie standard les messages passés en paramètres (après leur interprétation par le shell).

c. Options courantes

`-n` : n'affiche pas de saut de ligne final

d. Exemples :

```
FoxSIN:~# echo ceci est un petit message
ceci est un petit message
FoxSIN:~# echo $PATH
/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin
```

```
FoxSIN:~#
```

22 expr

e. Syntaxe :

```
expr exp1 {  + | - | \* | / | % } exp2
```

f. Description :

La commande **expr** évalue l'expression de la ligne de commande et retourne le résultat sur la sortie standard.

g. Options courantes

exp*X*	constante numérique ou variable du même type
+ | - | * | / | %	opérateur logique représentant respectivement l'addition, la soustraction, la multiplication (précédées d'un \ pour que le caractère * ne soit pas interprété par le shell), la division, le modulo (reste d'une division entière)

h. Exemples :

```
$ a=a+1
$ echo $a a+1
$ a=0

$ a=`expr a + 1` $ echo $a
1
$ echo `expr 23 % 6`
5
```

i. Remarque :

Il est indispensable de mettre un espace entre exp1 et l'opérateur, et entre l'opérateur et exp2 .

23 test

j. Syntaxe :

```
test [expression] [ expression ]
```

k. Description :

La commande **test** évalue expression et si sa valeur est vraie, retourne un code de sortie zéro

l. Options courantes

m. Options courantes

ch1 = ch2	chaines de caractères ch1 et ch2 identiques

`ch1 != ch2`	chaines de caractères `ch1` et `ch2` différentes
`nb1 - eq nb2`	nombres `nb1` et `nb2` égaux
`nb1 -ne nb2`	nombres `nb1` et `nb2` différents
`nb1 -gt nb2`	nombre `nb1` supérieur à `nb2`
`nb1 - ge nb2`	nombre `nb1` supérieur ou egal à `nb2`
`nb1 - lt nb2`	nombre `nb1` inférieur à `nb2`
`nb1 -le nb2`	nombre `nb1` inférieur ou egal à `nb2`
`-r fic`	vrai si `fic` existe et est lisible
	vrai si `fic` existe et est accessible en écriture
`-w fic`	vrai si `fic` existe et est exécutable
`-x fic`	vrai si `fic` existe et est un fichier ordinaire
`-f fic`	vrai si `fic` existe et est un fichier répertoire
`-d fic`	opérateur logique ET o
`-a -o`	pérateur logique OU

n. Exemples :

```
$ test $USERNAME = rs1
$ echo $?
1
$ [ -d / tmp ]
$ echo $?
0
```

Affichage du code de sortie (*exit status*).
1= FAUX
0=VRAI

o. Remarque :

cette commande est le plus souvent associée à des instructions de controle (if, while, ...)dans les scripts shell.

24 clear

p. Syntaxe :

```
clear
```

q. Description :

La commande **clear** efface l'écran du terminal actif.

r. Options courantes

clear n'accepte pas d'option

s. Exemples :

```
$ clear
```

V. UTILITAIRES RÉSEAU

⇒ **ping** vérification d'une connexion réseau.
⇒ **telnet** connexion au travers du réseau.

25 ping

a. Syntaxe :

```
ping correspondant [délai]
```

b. Description :

La commande **ping** envoie sur le réseau des paquets de réflexion. C'est à dire que le destinataire renvoie les paquets à l'émetteur.
Cette commande permet donc de vérifier une connexion réseau entre deux correspondants.

c. Option courantes

`correspondant délai` : nom du correspondant ou adresse IP
délai :d'attente entre l'émission d'un paquet et sa réponse

d. Exemples :

```
$ ping www.linux-france.com www.linux-france.com is alive $ ping
www.zindows-france.com
UX:ping: INFO: no  answer  from www.zindows-france.com
```

e. Remarque :

26 telnet

f. Syntaxe :

```
telnet [correspondant]
```

g. Description :

La commande **`telnet`** permet d'ouvrir une session sur une machine distante.
Si le `correspondant` n'est pas précisé sur la ligne de commande, `telnet` fonctionnera en mode interactif (prompt : `telnet>`) et le résumé des commandes s'obtient avec `?` .

h. Options courantes

`correspondant` nom du correspondant ou adresse IP

i. Exemples :

```
$ telnet www.linux-france.com Trying 192.124.13.42...

Connected to www.linux-france.com. Escape character is '^]'.

Red Hat Linux release 5.1 (Manhattan) Kernel 2.0.35 on an i586
login:
```

VI. COMMANDES D'ADMINISTRATION

⇒ **`id`** identification d'utilisateur et de groupe
⇒ **`ps`** liste et état des processus
⇒ **`passwd`** changement d'un mot de passe
⇒ **`kill`** émission de signal aux processus
⇒ **`who`** état des connexions au système
⇒ **`df`** état d'occupation des systèmes de fichiers
⇒ **`su`** changement d'identité
⇒ **`which`** localisation d'une commande ou alias .

27 id

a. Syntaxe :

```
id [utilisateur]
```

b. Description :

La commande **id** affiche des informations concernant le numéro d'utilisateur (UID) ainsi que sur les groupes d'appartenance (GID).
Si `utilisateur` est omis, `id` affiche les informations concernant l'utilisateur courant.

c. Options courantes

`utilisateur`	nom d'un utilisateur connu du système

d. Exemples :

```
$       id
uid=102( rsl) gid=100( other) groups=101(    ftp)
$       id root
uid=0( root)  gid=3( sys)  groups=0( root),    1( other),  2( bin),  3( sys),
4(adm), 5( uucp), 6(mail), 7( tty), 8(audit), 10( nuucp), 12(
daemon), 23(cron), 25( dtadmin), 47( priv),9( lp)
```

28 ps

e. Syntaxe :

```
ps [-ef] [-t liste] [-u liste]
```

f. Description :

La commande **ps** affiche l'état des processus; si aucune option n'est donnée, ce sont les processus de la session active qui sont affichés.

g. Options courantes

`-e -f`	affiche tous les processus du système affiche les information au format long
`-t liste`	affiche les processus liés aux terminaux de la `liste`
`-u liste`	affiche les processus liés aux utilisateurs de la `liste`

h. Exemples :

```
$ ps
PID    CLS    PRI    TTY    TIME   COMD
6665   TS     70     pts/3  0:00   ksh
```

```
9280    TS      59      pts/3   0:00    ps
$ ps -ef | head -5
UID     PID     PPID    CLS     PRI C   STIME   TTY TIME COMD
root    0       0       SYS     79      0       Nov 28 ?        0:14 sysproc
root    1       0       TS      70      0       Nov 28 ?        0:03
/sbin/init
root    1019    1       TS      85      0       Nov 28 ?        0:00
/usr/lib/saf/sac -t 300
root    88      1       TS      88      0       Nov 28 ?        0:00
/usr/lib/mousemgr
$ ps -u rsl
PID     CLS     PRI     TTY             TIME    COMD
964     TS      80      ?               0:00    srv_tab
958     TS      80      ?               0:00    xlemsup
961     TS      80      ?               0:00    srv_imp
6665    TS      70      pts/3           0:00    ksh
9323    TS      59      pts/3   0:00    ps
```

i. Remarque :

Sous Linux , il faut utiliser l'option **x** pour voir tout les processus.

29 passwd

j. Syntaxe :

```
passwd [utilisateur]
```

k. Description :

La commande **passwd** permet à l'utilisateur de modifier son mot de passe.
Si vous êtes `root` , il vous est alors possible de modifier le mot de passe des autres utilisateurs.

l. Options courantes

`utilisateur`	nom d'un utilisateur du système

m. Exemples :

```
$ passwd
UX:passwd: INFO: Changing password for rsl Old password:
New password:
Re-enter new  password:
$
```

30 kill

n. Syntaxe :

```
kill [-sig] num_process
```

o. Description :

La commande **kill** envoie au processus portant le numéro num_process un signal (sig).
Par défaut, c'est le signal 15 (TERM) qui est envoyé.

p. Options courantes

-sig signal valide à transmettre.
Les plus courants sont :
15 (TERM) demande au processus de se terminer (proprement!!!)
9 (KILL) demande au processus de se terminer (inconditionnel)
num_process numéro d'un processus (PID)

q. Exemples :

```
$       ps
        PID     CLS     PRI     TTY     TIME    COMD
        6665    TS      70      pts/3   0:00    ksh
        9280    TS      59      pts/3   0:00    ps
$       kill -9 6665
```

r. Remarque :

Le numéro du processus (PID) peut être déterminé avec la commande **ps**
La commande kill ne vous permettant pas de tuer les tâches des autres utilisateurs (seul le compte root peut le faire).

31 who

s. Syntaxe :

```
who am i who
```

t. Description :

La commande **who** affiche les utilisateurs connectés au système. Elle permet aussi de vous informer sur votre connexion.

u. Options courantes

am i :'qui suis-je' en français; information sur votre connexion

v. Exemples :

```
$ who
root    tty0    Nov 30 10:41
rsl     ttyp0   Nov 30 15:28 (192.152.230.14)
rsl     pts/3   Nov 30 09:07 (192.152.230.15)
$ who am i
```

```
rsl     pts/3   Nov 30 09:07 (192.152.230.15)
```

w. Remarque :

ATTENTION, si vous avez utilisé la commande **su**, **who** ne vous donnera que les informations de votre connexion initiale.

32 df

x. Syntaxe :

```
df -k
```

y. Description :

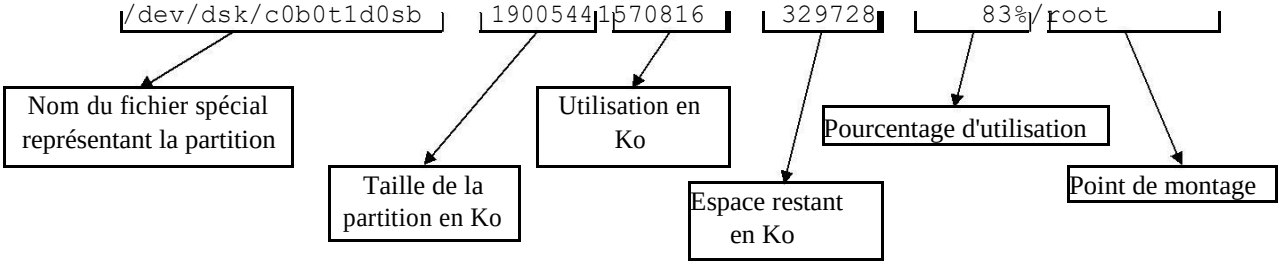
La commande **df** vous donne des informations sur l'état d'occupation des systèmes de fichiers.
Par défaut, df donne ces indications en blocs

z. Options courantes

`-k` donne les indications en Ko

aa. Exemples :

```
$ df -k
filesystem              kbytes  used     avail    capacitymounted on
/dev/root               500736  344072    156664        69%/
/proc                       00                0         0%/proc
/dev/dsk/c0b0t0d0sa      16384    4298     12086        26%/stand
/dev/fd                     00                0         0%/dev/fd
/dev/dsk/c0b0t1d0s3   20070449800        150904        25%/var
/dev/dsk/c0b0t0d0s3   30003295896        204136        32%/home
/dev/dsk/c0b0t0d0s4    1210368  251472   958896        21%/home
/dev/dsk/c0b0t0d0sb   19005441700720     199824        89%/rsl
/dev/dsk/c0b0t1d0s1    1935360  745184  1190176        39%/ux
/dev/dsk/c0b0t1d0sb    1900544 1570816   329728        83%/root
```



33 su

bb. Syntaxe :

```
su - [utilisateur]
```

cc. Description :

La commande **su** permet de changer en cours de session l'utilisateur courant.
Par défaut, si utilisateur n'est pas précisé, **su** essaie de vous connecter `root`

dd. Options courantes

`-utilisateur` : charge le profil de l'utilisateur (variables, ...) nom d'un utilisateur connu du système

ee. Exemples :

```
$ su
Password:
$ su - rsl Password:
```

34 which

ff. Syntaxe :

```
which [commande]
```

gg. Description :

La commande **which** vous indique la commande qui sera exécutée si vous tapez `commande` .
which effectue une recherche dans le `PATH`.

hh. Options courantes

`commande`	commande telle que vous la tapez au clavier

ii. Exemples :

```
$ which ls
/usr/bin/ls

$ which startx
/usr/bin/X11/startx
```

VII. ARCHIVAGE ET RESTAURATION DE DONNÉES

- cpio.
- tar.

35 cpio

a. Syntaxe :

```
cpio -i[cvBdmut] [-E fic] < fichier_archive cpio -o[cvB] >
fichier_archive
```

b. Description :

La commande **cpio** permet d'archiver les fichiers dont les noms sont reçus sur l'entrée standard et de restaurer les fichiers d'une archive.
Les archives peuvent être soit des fichiers normaux, soit des fichiers spéciaux de type blocs, ce qui permet de mettre les archives directement sur un support physique (streamer, DAT, disquette, ...).
Par sa manière de créer une archive, cpio est entièrement portable entre différents systèmes (UNIXWARE esim2, SINIX esim1, ...). Il doit par conséquent être votre outil privilégié pour l'archivage.
ATTENTION, si une archive a été créée avec des chemins absolus, il n'est pas possible de les restaurer ailleurs qu'à leur emplacement d'origine.

c. Options courantes

`-i`	restauration ou listage d'une archive existante
`-o`	création d'archive cpio
`-c`	ajout d'un en-tête ASCII (portabilité)
`-v`	mode bavard
`-B`	écrit des blocs de 5 Ko
`-d`	restauration de l'arborescence
`-m`	restauration des dates de modification des fichiers
`-u`	restauration inconditionnelle
`-t`	listage des fichiers de l'archive
`-E fic`	spécifie le fichier `fic` a restaurer

d. Exemples :

```
$ find . -print | cpio -ocvBdmu >/ dev/streamer fic1
fic2
rep1
rep1/fic11
rep2/fic12
$ cpio -icvBdmu < /dev/fd0 fic1
rep1
rep1/fic11
$ cpio -itv < / dev/dat
-rw-rw-rw-    1      root   other 192     Nov 30 10:36 1998,    fic1
drw-rw-rw-    1      root   other 331     Nov    30     10:36  1998,
       rep1
-rwxr-xr-x    1      root   other 138     Apr    25     09:29  1997,
       rep1/fic11
```

36 tar

e. Syntaxe :

```
tar c[vf] [ fic_sortie] [fichier...] tar x[vf] [ fic_entree]
[fichier...] tar t[vf] [ fic_entree] [fichier...]
```

f. Description :

La commande **tar** archive et restaure les fichiers entrés sur la ligne de commande.
ATTENTION, si une archive a été créée avec des chemins absolus, il n'est pas possible de la restaurer ailleurs qu'à son emplacement d'origine.

g. Options courantes

c	création d'un fichier d'archive
x	extraction de fichiers d'une archive
t	listage du contenu d'une archive
v	mode bavard
f	précise le fichier d'archive à utiliser

h. Exemples :

```
$ tar cvf /dev/fd0 *
a fic1 1 tape block
a rep1   1 tape block
a rep1/fic11 1tape block
$ tar tvf archive
-rw-r--r--102/100      52      Jul     8       11:26 1998      fic1
drw-r--r--102/100      34      Jul     8       11:26 1998      rep1
-rw-rw-rw-102/100      36      Nov 30  09:07 1998      rep1/fic11
$ tar xvf / dev/streamer
x fic1, 52 bytes, 1 tape block x rep1, 34 bytes, 1 tape block
x rep1/fic11, 36 bytes, 1 tape block
```