Memento des commandes de base sous linux

Répertoires pwd

Affiche le chemin absolu du répertoire courant (*Print Working Directory*).

```
$ pwd
/home/nicolas
```

cd [répertoire]

Change de répertoire (*Change Directory*). Va dans `répertoire` ou dans le répertoire de l'utilisateur s'il n'y a pas d'argument.
Si "`-`" est indiqué en argument, déplace dans le répertoire précédent.

```
$ pwd
/home/nicolas
$ cd /var/tmp
$ pwd
/var/tmp
$ cd ..
$ pwd
/var
$ cd
$ pwd
/home/nicolas
$ cd -
/var
$ pwd
/var
```

ls [fichier ...]

Liste le contenu des répertoires ou le nom des fichiers passés en arguments (liste le répertoire courant si pas d'argument).
`-l` : affichage détaillé (long)
`-a` : affichage aussi des fichiers cachés dont le nom commence par un point (all)
`-i` : affichage des numéros d'inodes (inode)
`-d` : affichage du nom du répertoire et non de son contenu (directory)
`-t` : trie l'affichage suivant la date de modification des fichiers (time)
`-r` : inverse le tri d'affichage (reverse)
`-R` : affichage du contenu de tous les sous répertoires (recursive)

```
$ ls
fic1  fic2  rep1
$ ls -a
.  ..  .bash_history  .bash_profile  .bashrc  .viminfo  fic1  fic2
rep1
$ ls -l
total 8
-rw-r--r--    1 nicolas  users           0 Dec  6 11:48 fic1
-rw-r--r--    1 nicolas  users         868 Dec  6 11:48 fic2
drwxr-xr-x    3 nicolas  users        4096 Dec  6 11:48 rep1
$ ls -R
.:
```

```
fic1  fic2  rep1

./rep1:
ficA  repA

./rep1/repA:
$ ls -ltr
total 8
-rw-r--r--    1 nicolas  users           0 Dec  6 11:48 fic1
drwxr-xr-x    3 nicolas  users        4096 Dec  6 11:48 rep1
-rw-r--r--    1 nicolas  users         868 Dec  6 11:48 fic2
$ ls -l rep1
total 4
-rw-r--r--    1 nicolas  users           0 Dec  6 11:48 ficA
drwxr-xr-x    2 nicolas  users        4096 Dec  6 11:42 repA
$ ls -ld rep1
drwxr-xr-x    3 nicolas  users        4096 Dec  6 11:48 rep1
```

`mkdir <répertoire ...>`

Créé les répertoires (*MaKe DIRectory*) passés en arguments.

```
$ ls -l
total 0
$ mkdir rep1
$ ls -l
total 4
drwxr-xr-x    2 nicolas  users        4096 Dec  6 11:41 rep1
$ mkdir rep1/repA rep2
$ ls -lR
.:
total 8
drwxr-xr-x    3 nicolas  users        4096 Dec  6 11:42 rep1
drwxr-xr-x    2 nicolas  users        4096 Dec  6 11:42 rep2

./rep1:
total 4
drwxr-xr-x    2 nicolas  users        4096 Dec  6 11:42 repA

./rep1/repA:
total 0

./rep2:
total 0
```

`rmdir <répertoire ...>`

Supprime les répertoires (*ReMove DIRectory*) passés en arguments s'ils sont vides.

```
$ rmdir rep1 rep2
rmdir: `rep1': Directory not empty
$ ls -l
total 4
drwxr-xr-x    3 nicolas  users        4096 Dec  6 11:42 rep1
```

Fichiers `cp <source ...> <destination>`

Copie (*CoPy*) les fichiers `source` vers `destination`.
`-i` : demande confirmation avant écrasement (interactive)
`-f` : écrase sans demander confirmation (force)

`-R` ou `-r` : copie aussi les répertoires (recursive)

```
$ ls -l
total 8
-rw-r--r--    1 nicolas  users           0 Dec  6 11:48 fic1
-rw-r--r--    1 nicolas  users         868 Dec  6 11:48 fic2
drwxr-xr-x    3 nicolas  users        4096 Dec  6 11:48 rep1
$ cp fic1 fic3
$ cp fic2 fic3
$ cp -i fic2 fic3
cp: overwrite `fic3'? o
$ cp rep1 rep2
cp: omitting directory `rep1'
$ cp -r rep1 rep2
$ cp -dpr rep1 /tmp
$ ls -l
total 16
-rw-r--r--    1 nicolas  users           0 Dec  6 11:48 fic1
-rw-r--r--    1 nicolas  users         868 Dec  6 11:48 fic2
-rw-r--r--    1 nicolas  users         868 Dec  6 14:08 fic3
drwxr-xr-x    3 nicolas  users        4096 Dec  6 11:48 rep1
drwxr-xr-x    3 nicolas  users        4096 Dec  6 14:09 rep2
$ ls -ld /tmp/rep1
drwxr-xr-x    3 nicolas  users        4096 Dec  6 11:48 /tmp/rep1
```

Attention : sans l'option `-R` (ou `-r`), la commande `cp` ne pourra pas copier les répertoires ; il est nécessaire qu'elle travaille en "récursif" pour parcourir l'arborescence de fichiers sous le répertoire, et ainsi pouvoir copier les fichiers sous jacents.

`mv <source ...> <destination>`

Renomme/déplace (*MoVe*) les fichiers `source` vers `destination`.
`-i` : demande confirmation avant écrasement (interactive)
`-f` : écrase sans demander confirmation (force)

```
$ ls
fic1  fic2  fic3  rep1  rep2
$ ls rep2
ficA  repA
$ mv fic* rep2
$ ls
rep1  rep2
$ ls rep2
fic1  fic2  fic3  ficA  repA
$ mv rep1 rep2
$ ls
rep2
$ ls rep2
fic1  fic2  fic3  ficA  rep1  repA
```

`rm <fichier ...>`

Supprime (*ReMove*) les fichiers passés en arguments.
`-i` : demande confirmation avant suppression (interactive)
`-f` : supprime sans demander confirmation (force)
`-R` : supprime aussi les répertoires (recursive)

```
$ ls -R .
.:
```

```
rep2

./rep2:
fic1  fic2  fic3  ficA  rep1  repA

./rep2/rep1:
ficA  repA

./rep2/rep1/repA:

./rep2/repA:
$ rm rep2/fic1
$ ls rep2
fic2  fic3  ficA  rep1  repA
$ rm rep2/rep1
rm: `rep2/rep1' is a directory
$ rm -r rep2/rep1
$ ls rep2
fic2  fic3  ficA  repA
```

ln <source> <destination>

Créé le lien (*LiNk*) `destination` vers le fichier `source`.
`-s` : créé un lien "symbolique" (soft) à la place d'un lien "dur" (hard)

```
$ echo coucou > fichier1
$ ls -l
total 4
-rw-r--r--    1 nicolas  users           7 Dec  6 14:24 fichier1
$ ln fichier1 fichier2
$ ln -s fichier1 fichier3
$ ls -l
total 8
-rw-r--r--    2 nicolas  users           7 Dec  6 14:24 fichier1
-rw-r--r--    2 nicolas  users           7 Dec  6 14:24 fichier2
lrwxrwxrwx    1 nicolas  users           8 Dec  6 14:25 fichier3 ->
fichier1
$ rm fichier1
$ ls -l
total 4
-rw-r--r--    1 nicolas  users           7 Dec  6 14:24 fichier2
lrwxrwxrwx    1 nicolas  users           8 Dec  6 14:25 fichier3 ->
fichier1
$ cat fichier2
coucou
$ cat fichier3
cat: fichier3: No such file or directory
```

touch <fichier ...>

Met à jour l'horodatage des fichiers passés en arguments ou créé ceux-ci s'ils n'existent pas.

```
$ ls -l
total 4
-rw-r--r--    1 nicolas  users           7 Dec  6 14:24 fichier
$ date
Mon Dec  6 14:30:02 CET 2004
$ touch fichier nouvfichier
$ ls -l
```

```
total 4
-rw-r--r--    1 nicolas  users           7 Dec  6 14:30 fichier
-rw-r--r--    1 nicolas  users           0 Dec  6 14:30 nouvfichier
```

type <cmd ...>

Indique comment chaque commande cmd doit être interprétée si elle est invoquée en ligne de commandes.

```
$ type ls cd passwd
ls is aliased to `ls --color=tty'
cd is a shell builtin
passwd is /usr/bin/passwd
```

which <cmd ...>

Retourne les chemins des binaires de chaque commande cmd passés en argument.
Ne prend pas en compte les commandes intégrées du shell et les alias.

```
$ which ls cd passwd
/bin/ls
/usr/bin/passwd
```

whereis <cmd ...>

Retourne les chemins du binaire, des sources et de la page de manuel des commandes passées en argument.
Comme pour la commande which, whereis ne tient pas compte de l'interprétation du shell.

```
$ whereis ls cd passwd
ls: /bin/ls /usr/share/man/man1/ls.1.gz
cd:
passwd: /usr/bin/passwd /etc/passwd /usr/share/man/man1/passwd.1.gz
/usr/share/m
an/man5/passwd.5.gz
```

find <chemin(s)> <critère(s)> <action(s)>

Recherche de fichiers multi-critères :
- recherche récursive dans le(s) répertoire(s) indiqué(s) (chemin(s))
- les principaux critères (critère(s)) sont :

```
 -name '<motif>'
 -size <[+|-]taille>
 -mtime <[+|-]date>
 -user <nom|UID>
 -newer <fichier référence>
```

- les principales actions (action(s)) sont :

```
 -print
 -ls
 -exec <commande shell avec {} pour spécifier le fichier
trouvé> \;
 -ok <commande shell avec {} pour spécifier le fichier
trouvé> \;
```

```
$ find /home /usr -name 'ab*' -print 2> /dev/null
```

```
/usr/share/vim/vim61/ftplugin/abaqus.vim
/usr/share/vim/vim61/syntax/abaqus.vim
/usr/share/vim/vim61/syntax/abc.vim
/usr/share/vim/vim61/syntax/abel.vim
$ find /home /usr -name 'ab*' -exec basename {} \; 2> /dev/null
abaqus.vim
abaqus.vim
abc.vim
abel.vim
```

updatedb

Création et mise à jour de la base de données utilisée par la commande `locate`.

```
# updatedb
```

locate

Recherche de fichiers suivant leur nom ; cette commande est plus rapide que la commande `find` car elle utilise une base de données des fichiers présents sur le système (voir la commande `updatedb`). Cependant, si la base de données des fichiers n'est pas à jour, le résultat de la recherche n'affiche pas tous les fichiers existants.

```
$ locate service
/etc/services
/usr/share/man/man5/services.5.gz
```

Traitement de fichiers et filtres `cat <fichier ...>`

Affiche le contenu des fichiers texte passés en arguments.

```
$ cat fictexte
Je vis que la Terre ayant besoin de la lumière,
de la chaleur, et de l'influence de ce grand feu,
elle se tourne autour de lui pour recevoir
également en toutes ses parties cette vertu qui
la conserve.
                Savinien de CYRANO DE BERGERAC
$
```

tac <fichier ...>

Affiche le contenu inversé (de la dernière ligne à la première) des fichiers texte passés en arguments.

```
$ tac fictexte
                Savinien de CYRANO DE BERGERAC
la conserve.
également en toutes ses parties cette vertu qui
elle se tourne autour de lui pour recevoir
de la chaleur, et de l'influence de ce grand feu,
Je vis que la Terre ayant besoin de la lumière,
$
```

nl <fichier ...>

Affiche le contenu des fichiers texte passés en arguments en numérotant les lignes.

```
$ nl fictexte
```

```
     1  Je vis que la Terre ayant besoin de la lumière,
     2  de la chaleur, et de l'influence de ce grand feu,
     3  elle se tourne autour de lui pour recevoir
     4  également en toutes ses parties cette vertu qui
     5  la conserve.
     6                  Savinien de CYRANO DE BERGERAC
$
```

more <fichier ...>

Affiche page par page le contenu des fichiers texte passés en arguments.

Pour afficher le contenu d'un fichier :

```
$ more /etc/services
# /etc/services:
# $Id: services,v 1.32 2003/01/09 17:56:30 dwalsh Exp $
#
# Network services, Internet style
#
# Note that it is presently the policy of IANA to assign a single
well-known
# port number for both TCP and UDP; hence, most entries here have two
entries
# even if the protocol doesn't support UDP operations.
# Updated from RFC 1700, ``Assigned Numbers'' (October 1994).  Not all
ports
# are included, only the more common ones.
#
# The latest IANA port assignments can be gotten from
#       http://www.iana.org/assignments/port-numbers
# The Well Known Ports are those from 0 through 1023.
# The Registered Ports are those from 1024 through 49151
# The Dynamic and/or Private Ports are those from 49152 through 65535
#
# Each line describes one service, and is of the form:
#
# service-name  port/protocol  [aliases ...]   [# comment]

--Encore--(4%)
```

Dans un tube, pour visualiser le résultat d'une commande :

```
$ ps -ef | more
UID        PID  PPID  C STIME TTY          TIME CMD
root         1     0  0 Oct23 ?        00:00:19 init [3]
root         2     1  0 Oct23 ?        00:00:00 [keventd]
root         3     1  0 Oct23 ?        00:00:00 [ksoftirqd_CPU0]
root         4     1  0 Oct23 ?        00:00:00 [ksoftirqd_CPU1]
root         5     1  0 Oct23 ?        00:00:04 [kswapd]
root         6     1  0 Oct23 ?        00:00:00 [bdflush]
root         7     1  0 Oct23 ?        00:01:10 [kupdated]
root         8     1  0 Oct23 ?        00:00:00 [scsi_eh_0]
root         9     1  0 Oct23 ?        00:00:00 [khubd]
root        11     1  0 Oct23 ?        00:01:14 [kjournald]
root       113     1  0 Oct23 ?        00:00:00 [kjournald]
root       114     1  0 Oct23 ?        00:00:46 [kjournald]
root       115     1  0 Oct23 ?        00:18:06 [kjournald]
root       116     1  0 Oct23 ?        00:01:08 [kjournald]
root       484     1  0 Oct23 ?        00:00:03 /usr/sbin/sshd
```

```
root       493     1  0 Oct23 ?        00:16:07 syslogd -m 0
root       497     1  0 Oct23 ?        00:00:00 klogd -x
rpc        507     1  0 Oct23 ?        00:00:00 portmap
rpcuser    526     1  0 Oct23 ?        00:00:00 rpc.statd
ldap       576     1  0 Oct23 ?        04:21:33 /usr/sbin/slapd -u
ldap -h ldap:
--Encore--
```

less <fichier ...>

Affiche page par page le contenu des fichiers texte passés en arguments.

```
$ less /etc/services
# /etc/services:
# $Id: services,v 1.32 2003/01/09 17:56:30 dwalsh Exp $
#
# Network services, Internet style
#
# Note that it is presently the policy of IANA to assign a single
well-known
# port number for both TCP and UDP; hence, most entries here have two
entries
# even if the protocol doesn't support UDP operations.
# Updated from RFC 1700, ``Assigned Numbers'' (October 1994).  Not all
ports
# are included, only the more common ones.
#
# The latest IANA port assignments can be gotten from
#       http://www.iana.org/assignments/port-numbers
# The Well Known Ports are those from 0 through 1023.
# The Registered Ports are those from 1024 through 49151
# The Dynamic and/or Private Ports are those from 49152 through 65535
#
# Each line describes one service, and is of the form:
#
# service-name  port/protocol  [aliases ...]   [# comment]

tcpmux          1/tcp                           # TCP port service
multiplexer
/etc/services
```

wc <fichier ...>

Affiche le nombre de lignes, de mots et de caractères (*Word Count*) contenus dans les fichiers passés en arguments.

`-l` : affiche uniquement le nombre de lignes (line)

`-w` : affiche uniquement le nombre de mots (word)

`-c` : affiche uniquement le nombre de caractères (character)

```
$ wc /etc/services
    569    2805   19935 /etc/services
$ ls | wc -l
      5
```

od <fichier ...>

Affiche (en octal par défaut) le contenu des fichiers binaires passés en arguments (*Octal Dump*).

`-x` : affiche les données en hexadécimal

Affichage en octal du fichier `/bin/ls` :

```
$ od /bin/ls
0000000 042577 043114 000401 000001 000000 000000 000000 000000
0000020 000002 000003 000001 000000 116300 004004 000064 000000
0000040 057254 000001 000000 000000 000064 000040 000010 000050
0000060 000037 000036 000006 000000 000064 000000 100064 004004
0000100 100064 004004 000400 000000 000400 000000 000005 000000
0000120 000004 000000 000003 000000 000464 000000 100464 004004
0000140 100464 004004 000023 000000 000023 000000 000004 000000
0000160 000001 000000 000001 000000 000000 000000 100000 004004
0000200 100000 004004 045073 000001 045073 000001 000005 000000
0000220 010000 000000 000001 000000 050000 000001 150000 004005
...
```

Affichage en hexadécimal du fichier `/bin/ls` :

```
$ od -x /bin/ls
0000000 457f 464c 0101 0001 0000 0000 0000 0000
0000020 0002 0003 0001 0000 9cc0 0804 0034 0000
0000040 5eac 0001 0000 0000 0034 0020 0008 0028
0000060 001f 001e 0006 0000 0034 0000 8034 0804
0000100 8034 0804 0100 0000 0100 0000 0005 0000
0000120 0004 0000 0003 0000 0134 0000 8134 0804
0000140 8134 0804 0013 0000 0013 0000 0004 0000
0000160 0001 0000 0001 0000 0000 0000 8000 0804
0000200 8000 0804 4a3b 0001 4a3b 0001 0005 0000
0000220 1000 0000 0001 0000 5000 0001 d000 0805
...
```

La première colonne correspond au décalage par rapport au début du fichier ; les données sont représentés dans les colonnes suivantes.

`strings <fichier ...>`

Affiche le contenu texte des fichiers binaires passés en arguments.

```
$ strings /bin/ls
/lib/ld-linux.so.2
libtermcap.so.2
_DYNAMIC
_init
tgetent
_fini
_GLOBAL_OFFSET_TABLE_
_Jv_RegisterClasses
tgetstr
__gmon_start__
libc.so.6
strcpy
ioctl
stdout
...
```

`split -b <taille>k <src> <dst>`

Découpe le fichier `src` en plusieurs fichiers de la `taille` spécifiée (en Ko ici) nommés `dstaa`, `dstab`, `dstac`, ...

On peut ensuite reconstituer le fichier original à l'aide de la commande `cat`.

```
$ ls -l
total 1272
-rw-r--r--    1 nicolas  users     1296504 Dec  6 15:45 grosfic
$ split -b 500k grosfic archive_
$ ls -l
total 2552
-rw-r--r--    1 nicolas  users      512000 Dec  6 15:46 archive_aa
-rw-r--r--    1 nicolas  users      512000 Dec  6 15:46 archive_ab
-rw-r--r--    1 nicolas  users      272504 Dec  6 15:46 archive_ac
-rw-r--r--    1 nicolas  users     1296504 Dec  6 15:45 grosfic
$ cat archive_* > grosfic2
$ ls -l
total 3824
-rw-r--r--    1 nicolas  users      512000 Dec  6 15:46 archive_aa
-rw-r--r--    1 nicolas  users      512000 Dec  6 15:46 archive_ab
-rw-r--r--    1 nicolas  users      272504 Dec  6 15:46 archive_ac
-rw-r--r--    1 nicolas  users     1296504 Dec  6 15:45 grosfic
-rw-r--r--    1 nicolas  users     1296504 Dec  6 15:46 grosfic2
```

join <fichier1> <fichier2>

Effectue une jointure (dans le sens d'une base de données relationnelle) entre deux fichiers texte.

```
$ cat fic1
1       nicolas
5       franck
3       gerard
12      stef
75      willy
24      gerald
8       alain
84      abdel
9       soraya
$ cat fic2
5       patron
8       commercial
3       directeur
12      administrateur
1       formateur
24      technicien
84      commercial
9       secretaire
58      patisssier
$ join fic1 fic2
5 franck patron
8 alain commercial
84 abdel commercial
9 soraya secretaire
$ sort -n fic1 > fic1s
$ sort -n fic2 > fic2s
$ join fic1s fic2s
1 nicolas formateur
3 gerard directeur
5 franck patron
8 alain commercial
9 soraya secretaire
12 stef administrateur
24 gerald technicien
```

```
84 abdel commercial
```

paste <fichier1> <fichier2> <...>

Fusionne ligne par ligne les fichiers passés en argument.

```
$ cat fic1
1       nicolas
5       franck
3       gerard
12      stef
75      willy
24      gerald
8       alain
84      abdel
9       soraya
$ cat fic2
5       patron
8       commercial
3       directeur
12      administrateur
1       formateur
24      technicien
84      commercial
9       secretaire
58      patisssier
$ paste fic1 fic2
1       nicolas 5       patron
5       franck  8       commercial
3       gerard  3       directeur
12      stef    12      administrateur
75      willy   1       formateur
24      gerald  24      technicien
8       alain   84      commercial
84      abdel   9       secretaire
9       soraya  58      patisssier
```

gzip <fichier>

Compresse le fichier `fichier` au format GNU Zip ; par défaut, le fichier compressé est nommé `fichier.gz` et l'original est supprimé.
Voir la commande `gunzip` pour décompresser le fichier généré.

```
$ ls -l
total 1144
-rw-r--r--    1 nicolas  users     1166532 Dec  6 16:21 fichier
$ gzip fichier
$ ls -l
total 140
-rw-r--r--    1 nicolas  users      137644 Dec  6 16:21 fichier.gz
```

gunzip <fichier[.gz]>

Décompresse le fichier `fichier.gz` qui est au format GNU Zip ; par défaut, l'extension `.gz` est supprimée dans le nom du fichier non compressé et le fichier original compressé est supprimé.
Il n'est pas obligatoire de préciser l'extension `.gz` dans le nom du fichier passé en argument.
Voir la commande `gzip` pour compresser un fichier.

```
$ ls -l
total 140
-rw-r--r--    1 nicolas  users      137644 Dec  6 16:21 fichier.gz
$ gunzip fichier
$ ls -l
total 1144
-rw-r--r--    1 nicolas  users     1166532 Dec  6 16:21 fichier
```

`tr <liste1> <liste2>`
`tr -s <liste>`
`tr -d <liste>`

Substitue un à un les caratères de la `liste1` par ceux de la `liste2` dans le texte envoyé sur l'entrée standard de la commande.
`-d` : supprime les caractères de la `liste` dans le texte envoyé sur l'entrée standard de la commande.
`-s` : supprime les doublons des caractères de la `liste` dans le texte envoyé sur l'entrée standard de la commande.

```
$ echo "une chaine de caracteres" | tr "ace" "AkE"
unE khAinE dE kArAktErEs
$ ls -l
total 1148
-rw-r--r--    1 nicolas  users     1166532 Dec  6 16:21 fichier
drwxr-xr-x    2 nicolas  users        4096 Dec  6 16:52 rep
$ ls -l | tr -d " "
total1148
-rw-r--r--1nicolasusers1166532Dec616:21fichier
drwxr-xr-x2nicolasusers4096Dec616:52rep
$ ls -l | tr -s " "
total 1148
-rw-r--r-- 1 nicolas users 1166532 Dec 6 16:21 fichier
drwxr-xr-x 2 nicolas users 4096 Dec 6 16:52 rep
```

`grep <regexp> [fichier ...]`

Affiche uniquement les lignes, des fichiers passés en argument, correspondantes à l'expression rationnelle (ou expression régulière) `regexp`.
`-v` : inverse le résultat de la commande (affiche seulement les lignes ne correspondant pas à `regexp`)
`-c` : retourne le nombre de correspondances
`-n` : affiche les numéros des lignes correspondantes
`-l` : affiche les noms des fichiers contenant des lignes correpondant à `regexp`
`-i` : ne tient pas compte de la casse des caractères

`sed <instr ...> [fichier ...]`

Applique les instructions sed sur les fichiers passés en arguments.
`-f instr.sed` : utilise les instructions sed contenues dans le fichier `instr.sed`
Voir des exemples sur cette page.

`awk <instr ...> [fichier ...]`

Applique les instructions awk sur les fichiers passés en arguments.
`-f instr.awk` : utilise les instructions awk contenues dans le fichier `instr.awk`
Voir des exemples sur cette page.

`cut -d<délimiteur> -f<champ(s)> [fichier]`
`cut -c<colonne(s)> [fichier]`

Affiche les champs spécifiés avec l'option `-f` et séparés par le délimiteur indiqué après l'option `-d`, ou affiche les colonnes de caractères indiquées après l'option `-c`.

Pour afficher les 3ème et 6ème colonnes du fichier `/etc/passwd` :

```
$ cat /etc/passwd
root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/bin/sh
bin:x:2:2:bin:/bin:/bin/sh
sys:x:3:3:sys:/dev:/bin/sh
nobody:x:65534:65534:nobody:/home:/bin/sh
sshd:x:100:65534::/var/run/sshd:/bin/false
nicolas:x:1000:100::/home/nicolas:/bin/bash
$ cut -d":" -f3,6 /etc/passwd
0:/root
1:/usr/sbin
2:/bin
3:/dev
65534:/home
100:/var/run/sshd
1000:/home/nicolas
```

Dans un tube, pour filtrer le résultat d'une commande :

```
$ ls -l
total 1148
-rw-r--r--    1 nicolas  users     1166532 Dec  6 16:21 fichier
drwxr-xr-x    2 nicolas  users        4096 Dec  6 16:52 rep
$ ls -l | cut -c-10,16-23,56-
total 1148
-rw-r--r-- nicolas fichier
drwxr-xr-x nicolas rep
```

`head -<n> [fichier]`

Affiche les `n` premières lignes (ou les 10 premières si `n` n'est pas spécifié).

```
$ cat numeros
ligne 1
ligne 2
ligne 3
ligne 4
ligne 5
ligne 6
ligne 7
ligne 8
ligne 9
ligne 10
ligne 11
$ head -3 numeros
ligne 1
ligne 2
ligne 3
```

`tail -<n> [fichier]`
`tail +<n> [fichier]`

Avec `-`, affiche les `n` dernières lignes (ou les 10 dernières si `n` n'est pas spécifié).
Avec `+`, affiche à partir de la `n`ième ligne.

```
$ cat numeros
ligne 1
ligne 2
ligne 3
ligne 4
ligne 5
ligne 6
ligne 7
ligne 8
ligne 9
ligne 10
ligne 11
$ tail -3 numeros
ligne 9
ligne 10
ligne 11
$ tail +7 numeros
ligne 7
ligne 8
ligne 9
ligne 10
ligne 11
```

`sort`

`file`

Retourne le type des fichiers passés en arguments.

```
$ file /home /bin/ls /etc/passwd /usr/X11R6/bin/startx
/home:                  symbolic link to `/ramdisk/home'
/bin/ls:                ELF 32-bit LSB executable, Intel 80386, version
1 (SYSV),
                        for GNU/Linux 2.2.0, dynamically linked (uses
shared libs), stripped
/etc/passwd:            ASCII text
/usr/X11R6/bin/startx: Bourne shell script text executable
```

Documentation `man [section] <argument>`

Affiche la page de manuel électronique dont le nom est `argument` (se trouvant dans la section du manuel éventuellement spécifié).
`-k` : retourne le nom des pages de manuel contenant `argument`

Par exemple :

```
$ man 7 signal
```

Qui affiche :

```
SIGNAL(7)               Manuel de l'administrateur Linux
SIGNAL(7)

NOM
       signal - Liste des signaux disponibles.
```

```
DESCRIPTION
       Linux  supporte  supporte à la fois les signaux POSIX
classiques ("sig-
       naux standards") et les signaux POSIX temps-réel.

   Signaux standards
       Linux supporte les signaux  standards  indiqués  ci-dessous.
Plusieurs
       d'entre-eux  dépendent  de  l'architecture,  comme  on  le voit
dans la
       colonne "Valeur". Lorsque trois valeurs  sont  indiquées,  la
première
       correspond  normalement  aux  architectures  Alpha et Sparc, la
seconde
       pour les ix86, PPC et la dernière pour les Mips.  Un - dénote
un signal
       absent pour l'architecture correspondante.

       Les symboles de la colonne "Action" ont la signification
suivante :

       Term   Par défaut, terminer le processus.

       Ign    Par défaut, ignorer le signal.

:
```

La navigation dans une page de manuel électronique se fait de la même manière que dans l'éditeur de texte Vi.

apropos <argument>

Idem que `man -k <argument>`

```
$ apropos signal
alarm                (2)  - Programmer un réveil (timer) pour
l'émission d'un signal
alarm                (2)  - set an alarm clock for delivery of a
signal
bosskill             (8)  - Envoyer un signal éventuellement mortel à
votre chef
gsignal              (3)  - software signal facility
kill                 (1)  - Envoyer un signal à un processus
kill                 (2)  - Envoyer un signal à un processus
kill                 (2)  - send signal to a process
killall              (1)  - Envoyer un signal à des processus indiqués
par leurs noms
...
```

Les chiffres entre parenthèses indiquent la section du manuel électronique qui contient la page indiquée.

Droits `chmod <mode> <fichier ...>`

Modifie les droits d'accès (*CHange MODe*) aux fichiers passés en arguments suivant le `mode` (notation symbolique ou octale).
`-R` : applique les modifications à toute l'arborescence (recursive).

```
$ ls -l
```

```
total 1148
-rw-r--r--    1 nicolas  users     1166532 Dec  6 16:21 fichier
drwxr-xr-x    2 nicolas  users        4096 Dec  6 16:52 rep
$ chmod ug+x fichier
$ chmod 754 rep
$ ls -l
total 1148
-rwxr-xr--    1 nicolas  users     1166532 Dec  6 16:21 fichier
drwxr-xr--    2 nicolas  users        4096 Dec  6 16:52 rep
```

`umask [mode]`

Affiche le masque binaire déterminant les droits par défaut sur les nouveaux fichiers créés (sans argument) ou modifie ce masque (argument en notation octale).

```
$ umask
0022
$ touch fic1
$ mkdir rep1
$ ls -l
total 4
-rw-r--r--    1 nicolas  users           0 Mar  2 12:31 fic1
drwxr-xr-x    2 nicolas  users        4096 Mar  2 12:31 rep1
$ umask 0027
$ touch fic2
$ mkdir rep2
$ ls -l
total 8
-rw-r--r--    1 nicolas  users           0 Mar  2 12:31 fic1
-rw-r-----    1 nicolas  users           0 Mar  2 12:32 fic2
drwxr-xr-x    2 nicolas  users        4096 Mar  2 12:31 rep1
drwxr-x---    2 nicolas  users        4096 Mar  2 12:32 rep2
```

Gestion des comptes utilisateur `passwd [utilisateur]`

Change le mot de passe du compte `utilisateur` (mot de passe de l'utilisateur connecté si pas d'argument).
`-l` : vérrouille le compte passé en argument (lock)
`-u` : débloque le compte passé en argument (unlock)
`-S` : indique l'état du compte passé en argument (status)

Processus `ps`

Liste les processus (programmes en cours d'exécution).
`-ef` : affiche tous les processus avec des statistiques supplémentaires
`aux` : affiche tous les processus avec des statistiques supplémentaires

```
$ ps -ef
UID        PID  PPID  C STIME TTY          TIME CMD
root         1     0  0 04:42 ?        00:00:05 init [2]
root         2     1  0 04:42 ?        00:00:00 [keventd]
root         3     0  0 04:42 ?        00:00:00 [ksoftirqd_CPU0]
root         4     0  0 04:42 ?        00:00:00 [kswapd]
root         5     0  0 04:42 ?        00:00:00 [bdflush]
root         6     0  0 04:42 ?        00:00:08 [kupdated]
root         7     1  0 04:42 ?        00:00:00 [i2oevtd]
root         8     1  0 04:42 ?        00:00:08 [kjournald]
root        67     1  0 04:42 ?        00:00:03 [kjournald]
root        68     1  0 04:42 ?        00:00:02 [kjournald]
```

```
root       136     1  0 04:42 ?        00:00:01 /sbin/syslogd
root       139     1  0 04:42 ?        00:00:00 /sbin/klogd
root       144     1  0 04:42 ?        00:00:00 /usr/sbin/inetd
root       151     1  0 04:42 ?        00:00:00 /usr/sbin/sshd
root       154     1  0 04:42 ?        00:00:00 /usr/sbin/cron
root       158     1  0 04:42 tty2     00:00:00 /sbin/getty 38400 tty2
root       159     1  0 04:42 tty3     00:00:00 /sbin/getty 38400 tty3
root       160     1  0 04:42 tty4     00:00:00 /sbin/getty 38400 tty4
root       161     1  0 04:42 tty5     00:00:00 /sbin/getty 38400 tty5
root       162     1  0 04:42 tty6     00:00:00 /sbin/getty 38400 tty6
root       465     1  0 07:49 tty1     00:00:00 /sbin/getty 38400 tty1
root      2503   151  0 11:46 ?        00:00:00 /usr/sbin/sshd
nicolas   2505  2503  0 11:47 ?        00:00:02 /usr/sbin/sshd
nicolas   2506  2505  0 11:47 pts/0    00:00:00 -bash
nicolas   2517  2506  0 12:09 pts/0    00:00:00 ps -ef

$ ps aux
USER       PID %CPU %MEM   VSZ  RSS TTY      STAT START   TIME COMMAND
root         1  0.0  0.3  1272  484 ?        S    04:42   0:05 init
[2]
root         2  0.0  0.0     0    0 ?        SW   04:42   0:00
[keventd]
root         3  0.0  0.0     0    0 ?        SWN  04:42   0:00
[ksoftirqd_CPU0]
root         4  0.0  0.0     0    0 ?        SW   04:42   0:00
[kswapd]
root         5  0.0  0.0     0    0 ?        SW   04:42   0:00
[bdflush]
root         6  0.0  0.0     0    0 ?        SW   04:42   0:08
[kupdated]
root         7  0.0  0.0     0    0 ?        SW   04:42   0:00
[i2oevtd]
root         8  0.0  0.0     0    0 ?        SW   04:42   0:08
[kjournald]
root        67  0.0  0.0     0    0 ?        SW   04:42   0:03
[kjournald]
root        68  0.0  0.0     0    0 ?        SW   04:42   0:02
[kjournald]
root       136  0.0  0.4  1344  596 ?        S    04:42   0:01
/sbin/syslogd
root       139  0.0  0.9  1940 1152 ?        S    04:42   0:00
/sbin/klogd
root       144  0.0  0.3  1248  428 ?        S    04:42   0:00
/usr/sbin/inetd
root       151  0.0  0.9  2784 1208 ?        S    04:42   0:00
/usr/sbin/sshd
root       154  0.0  0.5  1652  684 ?        S    04:42   0:00
/usr/sbin/cron
root       158  0.0  0.3  1252  468 tty2     S    04:42   0:00
/sbin/getty 38400
root       159  0.0  0.3  1252  468 tty3     S    04:42   0:00
/sbin/getty 38400
root       160  0.0  0.3  1252  468 tty4     S    04:42   0:00
/sbin/getty 38400
root       161  0.0  0.3  1252  468 tty5     S    04:42   0:00
/sbin/getty 38400
root       162  0.0  0.3  1252  468 tty6     S    04:42   0:00
/sbin/getty 38400
root       465  0.0  0.3  1252  468 tty1     S    07:49   0:00
/sbin/getty 38400
root      2503  0.0  1.3  5704 1676 ?        S    11:46   0:00
```

```
/usr/sbin/sshd
nicolas   2505  0.1  1.4  5740 1792 ?        S    11:46   0:02
/usr/sbin/sshd
nicolas   2506  0.0  1.0  2224 1260 pts/0    S    11:46   0:00 -bash
nicolas   2518  0.0  1.2  3564 1584 pts/0    R    12:09   0:00 ps aux
```

pstree

Liste les processus de façon arborescente.
`-p` : affiche aussi les PID des processus

```
$ pstree
init-+-cron
     |-6*[getty]
     |-i2oevtd
     |-inetd
     |-keventd
     |-3*[kjournald]
     |-klogd
     |-sshd---sshd---sshd---bash---pstree
     `-syslogd

$ pstree -p
init(1)-+-cron(154)
        |-getty(158)
        |-getty(159)
        |-getty(160)
        |-getty(161)
        |-getty(162)
        |-getty(465)
        |-i2oevtd(7)
        |-inetd(144)
        |-keventd(2)
        |-kjournald(8)
        |-kjournald(67)
        |-kjournald(68)
        |-klogd(139)
        |-sshd(151)---sshd(2503)---sshd(2505)---bash(2506)---
pstree(2520)
        `-syslogd(136)
```

jobs

Liste les "jobs" (processus lancés par le shell courant).

```
$ sleep 10m &
[1] 2523
$ sleep 20m &
[2] 2524
$ sleep 30m &
[3] 2525
$ jobs
[1]   Running                 sleep 10m &
[2]-  Running                 sleep 20m &
[3]+  Running                 sleep 30m &
```

bg %<job>

Relance le "job" `job` en arrière plan.

```
$ sleep 40m
         [Ctrl + z pour suspendre le processus lancé en avant plan]
[4]+  Stopped                 sleep 40m
$ bg %4
[4]+ sleep 40m &
$ jobs
[1]   Running                 sleep 10m &
[2]   Running                 sleep 20m &
[3]-  Running                 sleep 30m &
[4]+  Running                 sleep 40m &
```

`fg %<job>`

Relance le "job" `job` en avant plan.

```
$ jobs
[1]   Running                 sleep 10m &
[2]   Running                 sleep 20m &
[3]-  Running                 sleep 30m &
[4]+  Running                 sleep 40m &
$ fg %4
sleep 40m
```

`kill <signal> %<job>`

Envoi le signal `signal` au "job" `job`.

`kill <signal> <pid>`

Envoi le signal `signal` au processus dont le PID est `pid`.

`nohup <cmd> &`

Exécute la commande `cmd` en tâche de fond qui survivra à la déconnexion de l'utilisateur (no hang up).

`nice [gentillesse] <cmd> &`

Exécute la commande `cmd` avec la priorité définie par la `gentillesse` (nice) spécifiée en argument.

`renice <gentillesse> <PID>`

Redéfini la priorité du processus identifié par son `PID` suivant la `gentillesse` (nice) spécifiée.

`top`

Affiche une liste - rafraîchie régulièrement - des processus en cours d'exécution et permet de les contrôler (équivalent de `kill` et `renice`) ; affiche aussi des statistiques d'utilisation de la mémoire.

```
 11:47:56 up  7:05,  2 users,  load average: 0.03, 0.01, 0.00
28 processes: 27 sleeping, 1 running, 0 zombie, 0 stopped
CPU states:   0.1% user,   0.3% system,   0.0% nice,  99.6% idle
Mem:    125896K total,    64268K used,    61628K free,    15068K
buffers
Swap:   498004K total,        0K used,   498004K free,    32608K
```

```
cached

  PID USER     PRI  NI  SIZE  RSS SHARE STAT %CPU %MEM   TIME COMMAND
 2510 nicolas   17   0   928  928   748 R     2.8  0.7   0:00 top
    1 root       8   0   484  484   424 S     0.0  0.3   0:05 init
    2 root       9   0     0    0     0 SW    0.0  0.0   0:00 keventd
    3 root      19  19     0    0     0 SWN   0.0  0.0   0:00
ksoftirqd_CPU0
    4 root       9   0     0    0     0 SW    0.0  0.0   0:00 kswapd
    5 root       9   0     0    0     0 SW    0.0  0.0   0:00 bdflush
    6 root      11   0     0    0     0 SW    0.0  0.0   0:08 kupdated
    7 root       9   0     0    0     0 SW    0.0  0.0   0:00 i2oevtd
    8 root       9   0     0    0     0 SW    0.0  0.0   0:08
kjournald
   67 root       9   0     0    0     0 SW    0.0  0.0   0:03
kjournald
   68 root       9   0     0    0     0 SW    0.0  0.0   0:02
kjournald
  136 root       9   0   596  596   488 S     0.0  0.4   0:01 syslogd
  139 root       9   0  1152 1152   408 S     0.0  0.9   0:00 klogd
  144 root       9   0   428  428   376 S     0.0  0.3   0:00 inetd
  151 root       9   0  1208 1208  1080 S     0.0  0.9   0:00 sshd
  154 root       8   0   684  684   564 S     0.0  0.5   0:00 cron
  158 root       9   0   468  468   408 S     0.0  0.3   0:00 getty
  159 root       9   0   468  468   408 S     0.0  0.3   0:00 getty
  160 root       9   0   468  468   408 S     0.0  0.3   0:00 getty
  161 root       9   0   468  468   408 S     0.0  0.3   0:00 getty
  162 root       9   0   468  468   408 S     0.0  0.3   0:00 getty
  465 root       9   0   468  468   408 S     0.0  0.3   0:00 getty
 2152 root       9   0  1676 1676  1516 S     0.0  1.3   0:00 sshd
 2154 nicolas    9   0  1792 1792  1604 S     0.0  1.4   0:00 sshd
 2155 nicolas    9   0  1256 1256  1024 S     0.0  0.9   0:00 bash
 2503 root       9   0  1676 1676  1516 S     0.0  1.3   0:00 sshd
 2505 nicolas   10   0  1792 1792  1604 S     0.0  1.4   0:00 sshd
 2506 nicolas   12   0  1260 1260  1024 S     0.0  1.0   0:00 bash
```

Temps `date`

Affiche ou modifie la date système.
`+"FORMAT"` : affiche la date suivant un format définit par une chaîne de caractères composée de séquences représentant des éléments de la date.
`-s "mm/jj/aaaa hh:mm:ss"` : définit la date système (droits administrateur obligatoires).

```
$ date
Mon Dec  6 17:06:33 CET 2004
$ date +"nous sommes le %d/%m/%Y"
nous sommes le 06/12/2004

# date
Mon Dec  6 17:07:07 CET 2004
# date -s "03/02/2005 10:30:00"
Wed Mar  2 10:30:00 CET 2005
# date
Wed Mar  2 10:30:02 CET 2005
```

`sleep <durée>`

Attend sans rien faire autant de temps que spécifié en argument. L'unité par défaut est la seconde.

```
$ sleep 3  # attend 3 secondes
$ sleep 3s # idem que la ligne précédente
$ sleep 3m # attend 3 minutes
$ sleep 3h # attend 3 heures
$ sleep 3d # attend 3 jours
```

cal [[mois] année]

Affiche le calendrier du mois en cours (pas d'argument), de l'année en cours (un argument) ou du mois et de l'année spécifiés (deux arguments).

Le mois en cours (en date du 17/05/2006) :

```
$ cal
      Mai 2006
di lu ma me je ve 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
```

Le mois d'avril 2006 (attention à l'année 06 signifie l'an 6 après JC !) :

```
$ cal 04 06
      Avril 6
di lu ma me je ve 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

$ cal 04 2006
     Avril 2006
di lu ma me je ve 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
```

L'année 2006 complète :

```
$ cal 2006
                              2006

      Janvier               Février                 Mars
di lu ma me je ve sa  di lu ma me je ve sa  di lu ma me je ve sa
 1  2  3  4  5  6  7            1  2  3  4            1  2  3  4
 8  9 10 11 12 13 14   5  6  7  8  9 10 11   5  6  7  8  9 10 11
15 16 17 18 19 20 21  12 13 14 15 16 17 18  12 13 14 15 16 17 18
22 23 24 25 26 27 28  19 20 21 22 23 24 25  19 20 21 22 23 24 25
29 30 31              26 27 28              26 27 28 29 30 31

       Avril                  Mai                   Juin
di lu ma me je ve sa  di lu ma me je ve sa  di lu ma me je ve sa
                   1      1  2  3  4  5  6              1  2  3
```

```
 2  3  4  5  6  7  8   7  8  9 10 11 12 13   4  5  6  7  8  9 10
 9 10 11 12 13 14 15  14 15 16 17 18 19 20  11 12 13 14 15 16 17
16 17 18 19 20 21 22  21 22 23 24 25 26 27  18 19 20 21 22 23 24
23 24 25 26 27 28 29  28 29 30 31           25 26 27 28 29 30

...
```

Surveillance `who`

Liste les utilisateurs connectés au système.

`who am i`

Liste l'utilisateur actuellement connecté au système.

`whoami`

Retourne le login de l'utilisateur actuellement connecté au système.

`finger [utilisateur ...]`

Liste les utilisateurs connectés au système (pas d'argument) ou détaille les comptes des utilisateurs passés en arguments.

Communication `write [utilisateur[@hôte]]`

Affiche un message sur le terminal de l'utilisateur passé en argument.

`wall`

Affiche un message sur le terminal des utilisateurs connectés.

`mail [utilisateur[@hôte]]`

Lit la boîte aux lettres de l'utilisateur connecté (pas d'argument) ou envoi un message électronique dans la boîte aux lettres de l'utilisateur passé en argument.

`talk`

Établi une session de messagerie instantanée (chat) avec l'utilisateur passé en argument.

`mesg`

Affiche si l'utilisateur actuellement connecté accepte ou non les messages provenant d'autres utilisateurs.

`mesg y`

Autorise les messages provenant d'autres utilisateurs.

`mesg n`

Refuse les messages provenant d'autres utilisateurs.

Divers `clear`

Efface l'écran du terminal.

`echo [argument ...]`

Affiche sur la sortie standard les chaînes de caractères passées en arguments, séparées par un espace.

`exit [code_retour]`

Quitte le shell en cours avec le code retour passé en argument ; par défaut `exit` retourne le code retour `0`.

`logout`

Déconnecte l'utilisateur.

`alias [nom='cmd']`

Affiche les alias définis dans l'environnement shell actuel (pas d'argument) ou en défini un nouveau.

`unalias <nom>`

Supprime l'alias de l'environnement shell actuel.