Les commandes *Unix* de base

Avertissement : Toutes les commandes données dans ce cours sont issues d'une `SuSe` **Linux 7.1** avec `bash` comme shell.

Caractéristiques du système *Unix*

➤ multi-utilisateur
plusieurs utilisateurs peuvent travailler en même temps.

➤ multi-tâche
exécution et gestion des programmes en parallèle en temps partagé sur le processeur.

➤ sécurisé
contrôle d'accès aux matériels et aux fichiers à travers des systèmes de groupes et de droits. Par défaut,

- ◆ seul l'administrateur (le super-utilisateur) a la possibilité d'accéder aux parties sensibles du système et d'agir dessus.
- ◆ chaque utilisateur dispose d'une zone réservée dans laquelle il a tous les droits, et dont il contrôle les accès.
- ◆ des zones complètes du système peuvent être réservées à des groupes d'utilisateurs, et interdites à d'autres.

➤ basé sur le langage `C`.
le noyau du système ainsi que la majorité des fonctionnalités du système sont implémentés en `C`.

➤ conçu pour fonctionner en réseau
la majorité des protocoles standards ont été créés sous *Unix* et inclus au noyau. Ils y sont naturellement adaptés.

L'ensemble de ces caractéristiques fait d'*Unix* un système d'exploitation fiable et performant.

Notion d'utilisateur et de groupe

utilisateur : personne autorisée à se connecter sur le système.

- ❏ son accès est autorisé après identification :
  - ❏ par son nom d'utilisateur (`username` ou `login`).
  - ❏ par son mot de passe associé (`password`).
- ❏ il dispose d'une zone privée sur le disque, généralement située dans `/home/`*username* ou dans `/users/`*username* sur laquelle il a tous les droits.
- ❏ la taille de cette zone privée (*i.e.* le nombre d'octets qu'il est possible d'y stocker) peut être restreinte à un quota.
- ❏ il est identifié par un numéro unique ou `UID` (User IDentification number).
- ❏ il appartient à un ou plusieurs groupes (mais un groupe dans lequel il est par défaut).

groupe : ensemble de personnes ou de groupes.

- ❏ les groupes sont utilisés pour contrôler les accès au sein du système.
- ❏ le changement de groupe est (éventuellement) contrôlé par un mot de passe.
- ❏ il est géré par un administrateur spécifique au groupe (ou à défaut le super-utilisateur).
- ❏ il est identifié par un numéro unique ou `GID` (Group IDentification number).

Sous *Unix* , tout fichier, répertoire, service, ...
appartient à un utilisateur **et** à un groupe.

Fichiers de description des utilisateurs et des groupes

`/etc/passwd` pour les utilisateurs

`/etc/group` pour les groupes

Session

Une session est une utilisation d'une machine *Unix* par une personne autorisée.

- ➤ Elle commence par son identification (logging).
- ➤ Elle se termine par une déconnection (delogging).

❑ Ouverture d'une session

```
Welcome to SuSE Linux 6.2 (i386) - Kernel 2.2.10 (tty2).
mathinfo162 login: pascal
Password:
Last login: Mon Jul 31 11:53:19 on tty1
Another day to die ...
%
```

❑ shell

Programme lancé automatiquement immédiatement après l'ouverture de la session permettant d'exécuter des commandes *Unix* ou des programmes en ligne de commande (souvent **sh**, **csh** ou **bash**).

Le shell utilisé dans ce cours est **bash**.

❑ Fermeture d'une session

Par la commande `logout`, `exit` ou avec un `ctrl-D`.

Attention, toujours se délogger. Sinon :

- ➤ Toutes les informations contenues dans votre compte sont accessibles, modifiables et effaçables.
- ➤ Votre identité électronique peut être utilisée. Vous êtes responsables des actions effectuées depuis votre compte.

Commandes pour utilisateurs et groupes

id : affiche les informations d'identification de l'utilisateur.

whoami : affiche le nom de l'utilisateur.

users : affiche les noms de tous les utilisateurs connectés sur le système.

who : comme **users** mais avec plus d'informations.

passwd : change le mot de passe actuel.

quota : affiche les informations de quota (si disponibles). (`/tmp`)

groups : affiche les groupes auquels l'utilisateur appartient.

newgrp `groupname` : changement de groupe.

su `username` : changement d'identité de l'utilisateur.

lastlog : date de la dernière connection.

Exemple :

```
% id
uid=500(pascal) gid=100(users) groups=100(users)
% whoami
pascal
% users
pascal pascal root
% who
pascal    tty1     Sep 13 11:23
pascal    pts/0    Sep 13 11:23 (:0.0)
root      tty2     Sep 13 11:45
% passwd
Password:
New password:
New password (again):
Password changed
% groups
users
% lastlog -u pascal
Username     Port  From     Latest
pascal       tty1           Mon Sep  4 08:17:00 +0200 2000
% su
Password:
$ whoami
root
$ groups
root bin uucp shadow dialout nogroup
$ exit
%
```

Fichiers et répertoires

❑ Règles sur les fichiers et répertoires sous *Unix* .

- ❖ lors de l'écriture d'un chemin, les noms des répertoires sont séparés par des / (exemple : `/home/pascal`).
- ❖ les noms de fichiers ou répertoires commençant par un point ne sont pas affichés par défaut (i.e. pour "cacher" un fichier, le faire commencer par un point).
- ❖ tous les caractères et tous les noms de fichiers sont possibles. Eviter les noms de fichiers commençant par - ou contenant des caractères de contrôle, et comme nom d'exécutable des noms de commande *Unix* ou shell (par exemple `test`).

❑ **ls** : contenu du répertoire courant (`list`).

options : **-l** : format long (type, permission, nb.lien, owner, group, size, mod.time, nom)

-a : affiche les fichiers cachés.

-R : affichage récursif.

❑ **commandes communes**

cp : copie (`copy`).

rm : effacement (`remove`).

mv : déplacement ou renommage (`move`).

options : **-v** : verbose (affichage des opérations effectuées).

-i : interactif (confirmation de chaque commande).

-R : récursif (s'applique à l'arborescence).

-f : force l'action (fichiers protégés).

❑ **commandes spécifiques aux répertoires**

pwd : nom du répertoire courant (`print working directory`).

cd : change de répertoire (`change directory`).

mkdir : crée un répertoire vide (`make directory`).

rmdir : efface un répertoire vide (`remove directory`).

pushd popd dirs : gestion de l'historique des répertoires avec une pile (ajoute à la pile, dépile, contenu de la pile).

Fichiers et répertoires : exemples

```
% ls -a
.     ..     .bashrc   .profile tata    titi      toto
% ls -l
total 23
drwxr-xr-x   2 pascal   users    1024 Sep 13 20:29 tata
-rw-r--r--   1 pascal   users    4630 Sep 13 17:04 titi
-rw-r--r--   1 pascal   users   15834 Sep 13 17:04 toto
% ls -R
tata  titi  toto
tata:
tutu
% cp -v titi tutu
titi -> tutu
% rm -i t*
rm: tata: is a directory
rm: remove ‘titi’? n
rm: remove ‘toto’? n
rm: remove ‘tutu’? y
% cp tata lulu
cp: tata: omitting directory
% cp -R tata lulu
% rm -Ri lulu
rm: descend directory ‘lulu’? y
rm: remove ‘lulu/tutu’? y
rm: remove directory ‘lulu’? n
% mv lulu lolo
% ls -F
 lolo/  tata/  toto
% mv tata/ toto lolo/
% pwd
/home/pascal
% cd lolo
% mkdir titi
% ls -F
tata/  titi/  toto
% rmdir tata
rmdir: tata: Directory not empty
% rmdir titi
% pushd /tmp
/tmp /home/pascal/lolo
% pwd
/tmp
% popd
/home/pascal/lolo
% pwd
/home/pascal/lolo
```

Expressions régulières pour les noms de fichiers (rappel)

Lorsqu'une commande contient une expression régulière, le shell interprète l'expression régulière, la remplace dans la commande par une liste de fichiers ou de répertoire, puis il exécute la nouvelle commande ainsi construite.

Requètes sur des fichiers existants

Pour les expressions régulières suivantes, l'expression régulière n'est étendue **que sur les fichiers existants** correspondants à l'expression régulière.

Définition

`*`	toute chaîne de caractères.
`?`	tout caractère.
`[aeyuio]`	toute lettre parmi la liste $\{a, e, y, u, i, o\}$.
`[!aeyuio]`	tout caractère autre que $\{a, e, y, u, i, o\}$.
`[a-f]`	les lettres de a à f.

Exemples

`ab*`	tout fichier commençant par ab.
`*ab`	tout fichier finissant par ab.
`a*b`	tout fichier commençant par a et finissant par b.
`a??`	tout fichier de **3** caractères commençant par a.
`[a-z]*.[cho]`	tout fichier commençant par une lettre minuscule et dont l'extension est .c, .h ou .o.

Requètes générale

$\{expr_1, \ldots, expr_n\}$	$expr_1$ ou $expr_2$ ou ... ou $expr_n$

Exemples

a{1,2}	les fichiers `a1` et `a2`.
a{1,b{A,B}}	les fichiers `a1`, `abA`, `abB`.

Différence entre ces deux écritures

`ls -l a[1-3]`	liste les fichiers existants parmi `a1`, `a2`, `a3`
`ls -l a{1,2,3}`	liste les fichiers `a1`, `a2` et `a3`. Une erreur est signalée si l'un de ces fichiers n'existe pas.
`mkdir a[1-3]`	création du répertoire `a[1-3]`.
`mkdir a{1,2,3}`	création des répertoires `a1`, `a2` et `a3`.

Fichiers et répertoires (2)

❑ lien

ln : copie par lien sur l'original (`link`).

On préfère généralement les liens symboliques (option **-s**) aux liens classiques (ou durs) car ils sont plus généraux et permettent de lier des répertoires.

Exemple :

```
% ls -F
tata/  toto
% ln toto titi
% ln tata tutu
ln: tata: hard link not allowed for directory
% ls -li
total 11
 76130 drwxr-xr-x  2 local  users   1024 Sep 13 21:55 tata
 76133 -rw-r--r--  2 local  users   4630 Sep 13 21:56 titi
 76133 -rw-r--r--  2 local  users   4630 Sep 13 21:56 toto
% rm toto
% ls
tata  titi
% ln -s titi toto
% ln -s tata tutu
% ls -l
total 6
drwxr-xr-x   2 local  users   1024 Sep 13 21:55 tata
-rw-r--r--   1 local  users   4630 Sep 13 21:56 titi
lrwxrwxrwx   1 local  users      4 Sep 13 23:10 toto -> titi
lrwxrwxrwx   1 local  users      4 Sep 13 23:10 tutu -> tata
% rm -R tata
% cd tutu
bash: cd: tutu: No such file or directory
```

❑ place occupée sur le disque

du : calcul de la place occupée par des fichiers ou des répertoires.

df : place libre sur le disque (avec point de montage).

Exemple :

```
% ls -F
lolo/  tutu
% ls -F lolo
tata/  titi   toto@  tutu@
% du .
1       ./lolo/tata
19      ./lolo
25      .
% df .
Filesystem  1024-blocks  Used Available Capacity Mounted on
/dev/hda3      987220  831145   105071     89%   /
```

Droits sur les fichiers et répertoires

❑ **Principe** : sous *Unix* , la sécurité se gère :

sur trois niveaux

① utilisateur (`User`)

② groupe (`Group`)

③ autres (`Other`)

et avec trois types d'accès :

❶ lecture (`Read`)

❷ écriture (`Write`)

❸ exécution ou traversée (`eXecute`)

❑ **Exemple** : sortie de la commande `ls -l`

```
-rw-r-----   1 pascal   users        1819 Sep  7 08:32 toto
```

La lecture des 10 premiers caractères est la suivante :

`-` type (- :fichier, d :répertoire, l :lien). `toto` est un fichier.

`rw-` droits d'accès pour l'utilisateur : lecture et écriture.

`r--` droits d'accès pour le groupe : lecture seule.

`---` droits d'accès pour les autres : aucun.

❑ **Gestion des droits**

chmod : changement des droits d'accès.

Exemple :

```
chmod u-w g+r toto
chmod g=rx o-rxw /home/pascal
chmod a=rx /home/pascal/.messages
chmod 700 toto
chmod 644 toto
```

Pour les deux derniers exemples, chaque chiffre code un niveau d'accès (dans l'ordre utilisateur-groupe-autres) dont le droit d'accès est codé en octal avec : 4=lecture, 2=écriture, 1=exécution, 0=rien.

umask : fixe les droits d'accès par défaut lors de la création d'un fichier en donnant le masque octal des modes à ne pas activer.

Exemple :

```
umask 077   ⇒ masque par défaut : rwx --- ---
umask 033                         rwx r-- r--
umask 227                         r-x r-x ---
```

Note : par défaut, seuls les fichiers exécutables sont créés avec le droit `x`.

Notion de processus

- **Processus** : tout programme en cours d'exécution sur la demande de l'utilisateur et le système.
- **Propriétés des processus sous** *Unix* La gestion d'un processus utilise les propriétés suivantes :
  - un numéro unique affecté à sa création (`PID` ou process ID).
  - le numéro du processus parent qui l'a lancé (`PPID`).
  - l'identité de son propriétaire.
  - ses caractéristiques temporelles (date de début, temps CPU utilisé).
  - ses caractéristiques mémoires (mémoire vive et virtuelle utilisées).
  - sa priorité (-20=priorité minimale, 0=priorité normale, 20=priorité maximale).
  - son état (`R`=exécution, `S`=endormi, `T`=stoppé, `Z`=zombie, `D`=sommeil définitif). Deux états supplémentaires sont signalés : `W`=n'utilise plus de mémoire, `N`=processus prioritaire.
- **Règle de fonctionnement des processus**
  - dans un shell, un processus peut être exécuté soit en premier plan (on doit attendre la fin de l´exécution pour pouvoir entrer une nuvelle commande), soit en tâche de fond (on récupère la main tout de suite, le processes tournant en parallèle avecc le shell).
  - un processus peut être stoppé, puis relancé plus tard sans l'affecter, ou bien même tué. Ceci est effectué par l'envoi de signaux au processus.
  - seul le propriétaire d'un processus peut le contrôler (exception faite du super-utilisateur).
  - un processus père ne peut pas mourir avant la mort de tous ses processus fils.
  - la fin d'un processus père entraîne la fin de tous ces processus fils.
- **Principaux signaux**

2	`SIGINT`	interruption
3	`SIGQUIT`	interruption avec core
9	`SIGKILL`	forcer terminaison
15	`SIGTERM`	terminer (défaut)
17	`SIGSTOP`	stopper (ou pause)

Gestion des processus

❑ Information sur les processus

ps : affiche le statut des processus (par défaut, seulement ceux de l'utilisateur).

top : affiche les processus utilisant le plus de temps de processeur (q pour quitter).

pstree : affiche l'arbre de parenté des processus.

time `commande` : exécute la commande puis affiche son temps d'exécution (`real`=écoulé, `user`=en mode user, `sys`=en mode noyau).

❑ Contrôle des processus

kill `-num pid` : envoie le signal `num` au processus `pid`. Si `-num` n'est pas spécifié, le signal par défaut est 15.

nohup `commande` : indique à la commande qu'elle ne doit pas se terminer si son père meurt.

wait : rend la main lorsque tous les processus en tâche de fond sont terminé.

nice renice : changement de la priorité d'un processus (*root*).

at batch : lancement différé de processus.

❑ Shell et processus

Un processus lancé depuis un shell s'appelle un "*job*". En plus de leurs numéros de `pid`, ces *job*s sont numérotés[1] dans l'ordre de leurs lancements. Ces numéros s'appellent les *job id* (ou `jid`).

Commandes

jobs	donne la liste des jobs en cours d'exécution.
commande **&**	lance *commande* en tâche de fond.
bg `%jid`	lance le job stoppé `jid` en tâche de fond.
fg `%jid`	lance le job stoppé `jid` en premier plan.

Séquences claviers

ctrl-C	tue le processus du premier plan.
ctrl-Z	suspend le processus du premier plan.

Références aux processus (utilisable à la place d'un `pid`)

`%jid`	le processus dont le numéro de job est `jid`.
`%`*mot*	le processus dont la commande commence par *mot*.
`%?`*mot*	le processus dont la commande contient *mot*.
`%+`	(ou `%%`) le dernier processus lancé en tâche de fond.
`%-`	l'avant-dernier processus lancé en tâche de fond.

[1] La numérotation est réinitialisée à 1 lorsqu'aucun *job* n'est exécuté.

Gestion des processus : exemples

```
% top
 11:48pm  up 12:26,  4 users,  load average: 1.09, 1.06, 1.01
49 processes: 48 sleeping, 1 running, 0 zombie, 0 stopped
CPU states:  0.5% user,  0.9% system,  0.0% nice, 98.6% idle
Mem:   30484K av,  28576K used,   1908K free,  24060K shrd,    764K buff
Swap:  68540K av,   2340K used,  66200K free                  8204K cached

  PID USER     PRI  NI  SIZE  RSS SHARE STAT  LIB %CPU %MEM   TIME COMMAND
 2438 pascal    19   0   784  784   612 R       0  1.3  2.5   0:00 top
 2376 local      5   0  2620 2620  1440 S       0  0.1  8.5   0:00 xterm
    1 root       0   0   196  196   168 S       0  0.0  0.6   0:03 init
    2 root       0   0     0    0     0 SW      0  0.0  0.0   0:00 kflushd
% time ls
lolo  tutu

real    0m0.016s
user    0m0.000s
sys     0m0.000s
%  find / -name "toto"
^Z
[1]+  Stopped                 find / -name "toto"
% ps
  PID TTY STAT TIME COMMAND
 2462   1 S    0:00 bash
 2585   1 T    0:00 find / -name toto
 2586   1 R    0:00 ps
% bg %1
[1]+ find / -name "toto" &
% ps
  PID TTY STAT TIME COMMAND
 2462   1 S    0:00 bash
 2585   1 D    0:01 find / -name toto
 2587   1 R    0:00 ps
% kill -9 2585
[1]+  Killed                  find / -name "toto"
% find / -name "toto"
^C
% ps
  PID TTY STAT TIME COMMAND
 2462  ?  S    0:00 bash
 2724  ?  R    0:00 ps
% nohup find / -name "toto" &
[1] 2750
nohup: appending output to ‘nohup.out’
%
^D
```

Shell : aide à la saisie des commandes

❑ Contrôle du défilement

ctrl-S : arrêt du défilement.

ctrl-Q : reprise du défilement.

❑ Déplacement sur la ligne de commande

ctrl-A : début de ligne.

ctrl-E : fin de ligne.

***alt-*←** **/** ***alt-*→** : mot précédent/suivant.

❑ Edition de la ligne de commande

ctrl-K : efface jusqu'à la fin de la ligne.

ctrl-W : efface le mot précédent.

ctrl-U : efface jusqu'au début de la ligne.

❑ Gestion de l'historique des commandes

history : historique de l'ensemble des commandes tapées.

!`histnum` : la commande numéro `histnum` de l'historique.

!! : la commande précédente.

ctrl-P **ou** ↑ : commande précédente.

ctrl-N **ou** ↓ : commande suivante.

ctrl-R : recherche dans l'historique.

❑ Complétion automatique des commandes

tab :

(une fois) complète la commande ou le nom de fichier.

(deux fois) affiche la liste des complétions possibles.

Personnalisation de l'environnement

❑ **Fichiers de démarrage et de sortie**

Les fichiers systèmes suivants contiennent les configurations par défaut :

`/etc/profile` : executés au login (obligatoire).

`~/.profile` : executés au login (U).

`~/.bashrc` : executés à chaque lancement de shell (U).

`~/.bash_logout` : executés à chaque sortie de shell (U).

❑ **Variables système**

Les commandes de manipulation des variables sont :

printenv : affiche l'ensemble des variables systèmes.

printenv `VAR` : affiche la valeur de la variable système `VAR`.

export `VAR=value` : affecte `value` à la variable système `VAR`.

Les principales variables système sont :

HOME : contient le répertoire racine de l'utilisateur.

PATH : contient l'ensemble des chemins dans lesquels les exécutables sont recherchés.

PS1 : contient la définition du prompt.

❑ **Alias** : les alias sont des raccourcis permettant de donner des noms simples à des commandes complexes :

Exemple :

```
% ll toto
bash: ll: command not found
% alias ll='ls -l'
% ll toto
-rw-r--r--   1 pascal   users       15834 Sep 13 17:04 toto
% alias l=less
% alias ls="ls -$LS_OPTIONS"
```

La commande `unalias` sert à supprimer un alias. En cas d'alias sur une commande classique, penser à utiliser son chemin complet (par exemple, `/bin/rm`).

❑ **Personnalisation du terminal**

stty : fixe ou affiche les caractéristiques texte du terminal.

xset : fixe ou affiche les caractéristiques du terminal X.

autres sources : `/usr/doc`

howto : `config-HOWTO`, `Bash-Prompt-HOWTO`

mini-howto : `colour-ls`, `Xterm-title`

Impression d'un fichier

Les impressions se gèrent avec une queue d'impression (les documents sont imprimés dans l'ordre où ils arrivent, et se voient attribuer un numéro de job `jid`).

lpr `fichier` : impression d'un fichier.

lpq : affichage de la file d'attente.

lpc : configuration et statut de la file d'attente.

lprm `jid` : suppression d'un travail (identifié par son numéro) dans la file d'attente.

Exemple :

```
% lpr toto
% lpq
lp is ready and printing
Rank   Owner      Job  Files                 Total Size
active pascal     1    toto                  150 bytes
% /usr/sbin/lpc
lpc> help
abort   enable  disable help    restart status  topq    ?
clean   exit    down    quit    start   stop    up
lpc> status
lp:
        queuing is enabled
        printing is enabled
        no entries
        lp is ready and printing
lpc> quit
% lprm 1
dfA001mathinfo162 dequeued
cfA001mathinfo162 dequeued
% lpq
no entries
```

Toutes ces commandes ont une option commune dans le cas où plusieurs imprimantes sont disponibles : **-P**`nomqueue` où `nomqueue` identifie l'imprimante à utiliser.

Archivage

La commande **tar** sert pour tous les travaux d'archivage. L'archivage peut aussi bien se faire sur un disque, une bande ou un fichier. La syntaxe est la suivante :

tar `options UnitéStockage fichiers`

où les options contiennent un seul mode d'accès à l'archive, et d'éventuelles options supplémentaires.

Modes :

c : création.

x : extraction.

t : contenu.

r : ajout.

Options supplémentaires :

v : mode verbose (décrit les opérations faites).

f : la sortie est un fichier et non pas une unité de stockage.

z : archive en mode compressé.

M : mode multivolume (*i.e.* sur plusieurs média).

Exemple :

```
% ls -l
total 22
-rw-r--r--   1 pascal   users   4630 Sep 13 17:04 titi
-rw-r--r--   1 pascal   users  15834 Sep 13 17:04 toto
% tar cvf sauve.tar t*
titi
toto
% ls -l sauve.tar
-rw-r--r--   1 pascal   users  30720 Sep 13 17:08 sauve.tar
% tar tvf sauve.tar
-rw-r--r-- pascal/users   4630 2000-09-13 17:04 titi
-rw-r--r-- pascal/users  15834 2000-09-13 17:04 toto
% tar czf sauve.tgz t*
% ls -l sauve.tgz
-rw-r--r--   1 pascal   users   6407 Sep 13 17:18 sauve.tgz
% tar cv /dev/fd0 t*
```

Compression de fichiers

.

compress / uncompress : compression/décompression (Lempel-Zip adaptatif). Extension du fichier compressé : `.Z`

gzip / gunzip : compression/décompression (LZ77). Extension du fichier compressé : `.gz`

Help ! Mais comment ça marche ?

❑ Aide sur les commandes

❑ L'aide intégrée

On passe en paramètre à la commande l'option **--help** (**-help**, **-h**, **-?**).

Exemple :
```
% ls --help
```

❑ L'aide en ligne

Grâce à la commande `man`.

Exemple :
```
% man ls
```

Les manuels sont subdivisés en 10 sections :

1	commandes
2	fonctions noyau
3	fonctions libraries
4	fichiers speciaux `/dev`
5	format de fichiers

6	jeux
7	codes et protocoles
8	commandes système
9	fonctions machine
n	autres applications

Le numéro de la section dans laquelle il faut rechercher peut être précisé :

Exemple :
```
% man exit
% man 2 exit
% man 3 exit
```

❑ L'aide en ligne (texinfo)

Sur les systèmes `Linux`, de l'aide peut également être obtenue en tapant avec la commande **info**.

❑ Aide sur le fonctionnement et la configuration

- ❑ lire la documentation.
- ❑ les livres.
- ❑ les `HOWTO` (où `mini-HOWTO`) du répertoire `/usr/doc`.