Licence d'informatique
Module de C/Unix

Commandes Unix de base

Nouvelle révision, septembre 1998

Introduction

Ce document présente rapidement les commandes Unix de base nécessaire pour débuter dans l'environnement Unix. Des détails supplémentaires sont disponibles dans le manuel en ligne :

```
% man <nom de commande>
```

Table des matières

Id : basic-commandes.tex,v 1.6 2001/10/29 07 :44 :07 marquet Exp

1 Syntaxe

1.1 Notations utilisées dans ce document

`[x]`	x peut être omis
`x...`	1 ou plusieurs occurrences de x
`[x]...`	0, 1 ou plusieurs occurrences de x
`[x | y]`	x ou y
`dirname`	nom de chemin repérant un répertoire
`filename`	nom de chemin repérant un fichier

1.2 Syntaxe d'une commande Unix

Une commande Unix est formée sur la syntaxe suivante :

```
commande [options]... [arguments]...
```

Les options sont souvent de la forme `-char`; de manière générale différentes options peuvent être regroupées :

```
ls -l -a
```

est équivalent à

```
ls -la
```

2 Principales commandes par thèmes

2.1 Manipulations du système de fichiers

commande	*description*
`pwd`	print working directory
`cd`	change directory
`mkdir`	make directory
`rmdir`	remove directory
`ls`	list directory
`mv`	move file
`cp`	copy file
`file`	type of file
`ln`	link files
`touch`	touch date of file
`find`	find file in directories
`chmod`	change mode of file
`cat`	show file
`more`	browser
`head`	head of file
`tail`	tail of file

2.2 Filtres Unix

commande	*description*
`tr`	translate characters
`sed`	stream editor
`grep`	grasp lines
`cut`	select fields
`paste`	paste fields
`uniq`	uniq and repeated lines
`tee`	copy stream in file
`sort`	sort lines
`join`	join files
`comm`	lines in common
`diff`	differences between files
`cmp`	compare files
`look`	find lines in a sorted file

2.3 Commandes utiles

commande	*description*
`man`	unix manual on line
`expr`	computes an expression
`sh`	Bourne shell
`ps`	processes list
`kill`	kill process

2.4 Bourne shell

commande	*description*
`if`	if clause
`for`	for loop
`case`	switch
`while`	while loop
`test`	condition for if and while
`getopts`	get options
`shift`	shift arguments
`echo`	echo
`read`	read one line
`exit`	exit from shell
`eval`	eval and execute
`break`	exit from loop
`set`	set shell parameters

3 Description des commandes

Cette section décrit dans l'ordre alphabétique les commandes de base. Toutes les options des commandes ne sont pas toujours données. Se référer au manuel en ligne (commande `man`) pour plus de détails.

3.1 `cat` — liste des fichiers

```
cat [filename]...
```

Écrit en séquence le contenu de chacun des fichiers argument sur la sortie standard. Si `filename` est omis, l'entrée standard est lue.

3.2 `cc` et `gcc`— compilation

```
cc -c [-g] sourcefile...
gcc -c [-g] sourcefile...
cc -o outputfile objectfile...
gcc -o outputfile objectfile...
```

Deux compilateurs C sont disponibles. La commande `cc` invoque le compilateur SPARCompiler qui est un produit de SUN. La commande `gcc` invoque le compialteur C développé par GNU.

Les premières formes sont des appels aux compilateurs ANSI C et génèrent un fichier objet (fichier `.o`). L'option `-g` permettra l'utilisation ultérieure du debugger.

Les secondes forment correspondent à la phase d'édition de liens pour la production d'un programme exécutable à partir d'un ensemble de fichiers objets.

3.3 `cd` — changement de répertoire

```
cd [dirname]
```

Changement du catalogue de travail pour `dirname`. Par défaut, le répertoire personnel est utilisé. Ce n'est pas une commande Unix, mais une commande intégrée à l'interpréteur de commandes.

3.4 `chmod` — changement des droits

```
chmod mode name...
```

Modifie les droits sur les fichiers ou catalogues `name`. Le `mode`peut s'exprimer de 2 manières :

Forme absolue Un ou bit à bit est effectué entre les valeurs (octale) suivantes :

0400	droit r pour le propriétaire
0200	droit w pour le propriétaire
0100	droit x pour le propriétaire
0040	droit r pour le groupe
0020	droit w pour le groupe
0010	droit x pour le groupe
0004	droit r pour les autres utilisateurs
0002	droit w pour les autres utilisateurs
0001	droit x pour les autres utilisateurs

Forme symbolique Elle suit la syntaxe

```
[who] op perm [op perm]...
```

Les opérations peuvent être

+	ajout de droits
−	retrait de droits
=	affectation de droits

L'utilisateur est désigné par une concaténation des lettres

`u`	désigne le propriétaire
`g`	désigne le groupe
`o`	désigne les autres utilisateurs
`a`	désigne tout le monde (correspond à `ugo`)

si `who` est omis, `a` est utilisé.
Les droits sont désignés par `perm` qui est une combinaison des lettres

`r`	pour lecture
`w`	pour écriture
`x`	pour exécution

3.5 `cp` — copie de fichiers

```
cp [-ip] filename1 filename2
cp -[r|R] [-ip] dirname1 dirname2
cp [-iprR] filename... dirname
```

La forme 1 copie le contenu de `filename` dans `filename2`. Les droits et le propriétaire de `filename2` sont préservés s'il existe déjà sinon les droits de `filename1` sont utilisés.

La forme 2 copie récursivement `dirname1` avec son contenu et ses sous-répertoires dans `dirname2`. Si `dirname2` n'existe pas, `cp` le crée et y duplique les fichiers et sous-répertoires de `dirname1` ; sinon, `cp` copie `dirname1` dans `dirname2` comme sous-répertoire.

La forme 3 copie chaque fichier dans `dirname` qui doit déjà exister.

les options de `cp` sont

`-i` demande confirmation si la copie écrase un fichier existant ;

`-p` duplique également les droits et la date de modification ;

`-r` | `-R` copie récursive (attention à la copie récursive d'un répertoire dans lui même !).

3.6 `date` — la date

```
date
```

Fournit la date et l'heure.

3.7 `df` — espace disque disponible

```
df
```

Écrit sur la sortie standard l'espace disque libre des systèmes de fichiers.

3.8 `echo` — affichage

```
echo [-n] [argument]...
```

Écrit ses arguments sur la sortie standard comme des chaînes de caractères séparées par des espaces. L'option `-n` n'ajoute pas de fin de ligne à la fin de la sortie.

3.9 `expr` — évaluation d'expressions

```
expr expression
```

calcule la valeur de l'expression `expression` et l'affiche sur la sortie standard. Les expressions sont de la forme

`nombre`	une donnée entière
`( expression)`	parenthésage
`expression * expression`	produit
`expression / expression`	quotient
`expression + expression`	addition
`expression - expression`	soustraction
`expression % expression`	modulo

D'autres opérateurs de traitements de chaînes de caractères sont aussi disponibles. Se référer au manuel en ligne.

3.10 `grep` — recherche de motifs dans un fichier

```
grep [-options] motif [filename]...
```

Sélectionne dans les fichiers `filename` (l'entrée standard par défaut) les lignes contenant le motif `motif` décrit par une expression régulière du type de celles utilisées dans les éditeurs de texte (`sed`). Le motif peut être de la forme :

`chaine`	une chaîne de caractères
`motif*`	un nombre quelconque d'occurrences de `motif`
`motif1 motif2`	la concaténation de 2 motifs

Les différentes options sont :

`-c` écrit seulement le nombre de lignes dans lesquelles le motif a été trouvé ;

`-h` n'affiche pas les noms de fichiers ;

`-i` ne fait pas de différence entre majuscules et minuscules ;

`-l` écrit seulement les noms de fichiers, pas les lignes ;

`-n` chaque ligne est précédée de son numéro dans le fichier qui la contient ;

`-s` n'affiche rien (utile si l'on veut seulement tester le code de retour) ;

`-v` écrit sur la sortie standard les lignes ne contenant pas le motif ;

`-e motif` identique à `motif` ; utile si `motif` commence par "`-`".

Le code de retour est :

0	au moins une coïncidence
1	aucune coïncidence
2	erreur

3.11 `ln` — création de lien

```
ln [-s] filename [name]
```

Crée un lien de nom `name` (par défaut `filename`) pour le fichier `filename`. L'option `-s` indique que ce lien est un line symbolique ; sinon c'est un lien physique. Les liens physique ne peuvent pas être créé à travers les systèmes de fichiers.

3.12 `ls` — lister un répertoire

```
ls [-options] [filename]...
```

Pour chaque répertoire de la liste `filename...`, écrit sur la sortie standard la liste des fichiers et répertoires qu'il contient.

Pour chaque fichiers de cette liste, écrit sur la sortie standard le nom du fichier. Par défaut, la sortie est triée par ordre alphabétique. Si aucun argument n'est donné, la commande traite le catalogue de travail.

Les options sont :

`-a` liste toutes les entrées. En son absence, les noms commençant par "`.`" ne sont pas listés ;

`-A` identique à `-a` sauf que "`.`", "`..`" ne sont pas listés ;

`-c` ordonne la sortie par date de dernière modification ;

`-C` force une sortie multi-colonnes avec les entrées triées du haut vers le bas (option par défaut) ;

`-d` si l'argument est un répertoire, écrit seulement son nom et ne liste pas son contenu ;

`-f` force chaque argument à être interprété comme un nom de répertoire et liste les noms trouvés. Cette option invalide les options `-l`, `-t`, `-s` et `-r`, et valide l'option `-a` ;

`-F` place un "`/`" derrière les noms de répertoire, un "`@`" derrière les noms de liens symbolique et un "`*`" derrière les fichiers exécutables ;

`-g` montre le groupe du propriétaire du fichier ;

`-i` écrit les numéros de i-nœud pour chaque fichier listé ;

`-l` liste les droits, le nombre de liens, le nom du propriétaire, la taille en octets et la date de dernière modification

`-R` liste récursivement les sous répertoires rencontrés.

3.13 `mail` — lecture/émission de message

```
mail [user]...
```

Sans argument, `mail` permet de lire les message qui nous sont destinés.

Les arguments sont des noms d'utilisateurs de la forme

`user`	pour les utilisateurs locaux
`user@site`	pour les utilisateurs de site distant

On préférera peut-être utiliser une interface graphique telle `mailtool` à la commande `mail`.

3.14 `man` — manuel en ligne

```
man [section] nom
```

Affiche les pages du manuel de référence. `nom` peut être un nom de commande, de fonction de bibliothèque, d'appel système...

3.15 `mkdir` — création de répertoire

```
mkdir [-p] dirname...
```

Crée des répertoires. L'option `-p` permet aux répertoires pères d'être créés si besoin.

3.16 `mv` — déplacement de fichiers

```
mv [-] [-fi] filename1 filename2
mv [-] [-fi] dirname1 dirname2
mv [-] [-fi] filename... dirname
```

La première forme change le nom du fichier `filename1` en `filename2`. Si `filename2` existe déjà, il est effacé avant que `filename1` ne soit renommé. Si `filename2` est protégé en écriture, `mv` affiche le mode et demande confirmation pour ne pas respecter la protection.

La deuxième forme change le nom du répertoire `dirname1` en `dirname2` seulement si `dirname2` n'existe pas. Sinon applique la troisième forme.

Cette troisième forme déplace les fichiers `filename` dans le dernier nom de la liste qui doit correspondre à un répertoire.

Les options sont

- `-` interprète tous les arguments suivants comme étant des noms de fichiers, ce qui permet des noms de fichiers commençant par `-` ;
- `-f` supprime tous les messages à propos des protections ;
- `-i` mode interactif : demande confirmation si un `mv` remplace un fichier existant.

3.17 `passwd` — changement de mot de passe

```
passwd
nispasswd
yppasswd
```

Ces trois commandes sont équivalentes et changent le mot de passe de l'utilisateur courant pour l'ensemble des machines Unix du domaine.

3.18 `ps` — liste les processus

```
ps [-option]
```

Affiche des informations sur les processus. Sans option, donne des informations sur les processus dont l'utilisateur est le propriétaire. L'options `-a` permet d'inclure les informations sur les processus dont on est pas le propriétaire ; d'autres options sont disponibles.

3.19 `pwd` — répertoire courant

```
pwd
```

Écrit le nom de chemin absolu du répertoire de travail.

3.20 `rm` — destruction de fichiers

```
rm [-] [-fir] filename...
```

Efface un ou plusieurs fichiers. Si une entrée était le dernier lien sur le fichier, le contenu du fichier est perdu. `rm` demande confirmation s'il n'y a pas droit d'écriture sur un des fichiers de la liste `filename`, mais il faut avoir le droit d'écriture sur le répertoire qui contient le fichier. Les options sont :

- `-` interprète tous les arguments suivants comme étant des noms de fichiers, ce qui permet des noms de fichiers commençant par `-` ;
- `-f` pas de demande de confirmation même s'il n'y a pas droit d'écriture sur le fichier ;
- `-i` interactif ; demande de confirmation avant d'effacer chaque fichier et, avec `-r`, d'examiner chaque répertoire ;
- `-r` efface récursivement le contenu d'un répertoire, ses sous-répertoires et le répertoire lui même.

3.21 `rusers` — liste les utilisateurs locaux

```
rusers
```

Le résultat de `rusers` est similaire à celui de `who`. La commande affiche la liste des utilisateurs connectés, mais pour les machines distantes. Une demande est diffusée sur le réseau local et les réponses sont affichées dès leur arrivée.

3.22 `talk` — communiquer avec un autre utilisateur

```
talk username@host
```

`talk` est un programme de communication qui fait dialoguer deux utilisateurs connectés sur deux stations.

3.23 `test` — comparaison de chaînes et d'entiers, test du type de fichiers

```
test expression
```

La commande `test` évalue l'expression `expression` Si sa valeur est vraie, `test` retourne 0 (vrai), sinon retourne 1 (faux). En cas d'erreur une valeur supérieure à 1 est retournée. Les primitives suivantes sont utilisées pour construire des expressions (consulter le manuel en ligne pour une description complète des expressions) :

Existence et droit sur les fichiers

- **`-r filename`** : vrai si le fichier `filename` existe et si le droit en lecture est positionné
- **`-w filename`** : vrai si le fichier `filename` existe et si le droit en écriture est positionné
- **`-x filename`** : vrai si le fichier `filename` existe et si le droit en exécution est positionné
- **`-f filename`** : vrai si le fichier `filename` existe et n'est pas un répertoire
- **`-d filename`** : vrai si le fichier `filename` existe et est un répertoire
- **`-s filename`** : vrai si le fichier `filename` existe et a une taille non nulle

Test sur des chaînes de caractères

- **`-n string`** : vrai si la longueur de la chaîne de caractères `string` est non-nulle
- **`-z string`** : vrai si la longueur de la chaîne de caractères string est nulle
- **`string1 = string2`** : vrai si les chaînes de caractères `string1` et `string2` sont égales
- **`string1 != string2`** : vrai si les deux chaînes de caractères sont différentes

Test sur des des entiers

- **`arg1 op arg2`** : les arguments sont des valeurs entières. L'opérateur binaire `op` est l'un des suivants : `-eq` (egalité), `-ne` (non-égalité), `-lt` (*less than*, plus petit que), `-le` (*less than or equal*, plus petit ou égal), `-gt` (*greater than*, plus grand que), `-ge` (*greater than or equal*, plus grand ou égal).

Combinaison des expressions Les primitives précédentes peuvent être combinées avec les opérateurs suivants :

- **`!`** : opérateur unaire de négation
- **`-a`** : opérateur binaire et
- **`-o`** : opérateur binaire ou
- **`( expression )`** : parenthésage des expressions

`-a` est prioritaire sur `-o`. Il faut remarquer que les parenthèses ont un sens pour le shell et doivent donc être précédées du caractère `\verb`.

3.24 `who` — liste des utilisateurs

```
who
```

Écrit la liste des utilisateurs connectés au système.