

FORMATION UNIX/LINUX RAPPEL COMMANDES DE BASE

Formateur : Anissa OMRANE

- Commandes de gestion d'arborescence de fichiers
- Gestion des droits d'accès
- Commandes et filtres usuels

3

Commandes de gestion d'arborescence de fichiers

Syntaxe d'une commande

4

- La syntaxe d'une commande

[Chemin/] nom_commande -[options ...] [arguments ...]

- Suite de **mots** {caractères non blancs} séparés par un ou plusieurs **séparateurs** {Espace – tabulation horizontale}
- Plusieurs caractères ont une signification particulière (méta-caractères)
- Une commande peut avoir des paramètres spécifiques [options]
introduits avec le signe '-' permettent de changer le comportement de base de la commande

Fichiers

5

- La base du système de fichiers est le fichier individuel (en anglais *file*).
- Le premier caractère de chaque ligne renvoyé par la commande `ls -l` correspond au type du fichier
- On distingue plusieurs types de fichiers
 - ▣ - les fichiers ordinaires (*ordinary files*) ;
 - ▣ d les répertoires (*directories*) ;
 - ▣ Les fichiers spéciaux (*special files* ou *devices*).
 - ▣ l lien symbolique
 - ▣ p pipe nommé
 - ▣ s socket
 - ▣ c périphérique en mode caractère (clavier)
 - ▣ b périphérique en mode bloc (disque)

Nomenclature fichier

6

- Dans Unix/Linux on fait la distinction entre minuscules et majuscules.
- De nombreux caractères servent en effet de caractères spéciaux pour le shell ou ont une signification particulière pour certains utilitaires.
- En règle générale, il est possible d'utiliser tous les caractères ASCII étendus à 8 bits.
- Le nom peut contenir le caractère de soulignement "_", le point ".", ou le signe "+". Le caractère "-" ne doit pas être utilisé au début d'un nom de fichier car de nombreuses commandes tenteraient d'interpréter ce nom comme une option, ce qui conduirait inévitablement à des erreurs.

- Exemples de noms de fichier valide

Budget2000

Budget2000

budget.2000

budget.avril.200

avril2000+juin2000

Exemple de noms litigieux

2*2

Adam(et_Eve)

Lorel&Hardie

-r

Méta-caractères d'expansion

7

Expansion des noms de fichiers

* remplace n'importe quelle chaîne de caractères

`micro*`

`microprocesseur micro-ordinateur microcosme`

? remplace n'importe quel caractère

`m?cro`

`micro macro`

[...] établit une coïncidence avec l'un des caractères mis entre crochets

`m[ai]*`

`mail macro micro mini`

[.-.] établit une coïncidence avec tous les caractères compris entre les 2 bornes

`ls [A-Z]*.[hc1-8]`

`Prog.c Include.h Data.1 Text.6`

.{...,...} remplace successivement les chaînes intérieures

`ls *.{C,cc}`

~ remplace par le \$HOME ou celui d'un autre login (csh, bash).

`cd ~username`

Les répertoires . et ..

8

- Tout répertoire contient deux répertoires particuliers qui ont pour noms . et ..
- . représente le répertoire de travail (répertoire courant),
- .. représente le répertoire parent du répertoire courant.
- Pour remonter dans le répertoire parent : cd ..
- ~ représente le répertoire d'accueil de l'utilisateur courant.

Commande de gestion arborescences

9

- ls
- cd/pwd
- mkdir/ rmdir
- touch
- rm
- mv
- cp

La commande `ls`

10

- **`ls`**: Lister le nom des fichiers du répertoire courant : `ls` (pour list)
- Exemple : `ls -l`, `ls -a` dans votre répertoire d'accueil.
 - ▣ `-l` : format détaillé
 - ▣ `-a` : liste aussi les noms de fichiers qui commencent par un "." (qui ne sont pas normalement listés)
 - ▣ `-A` : comme l'option `-a` mais les répertoires "." et ".." ne sont pas affichés
 - ▣ `-d` : si l'argument est un répertoire, liste seulement son nom (et pas les fichiers qui sont sous ce répertoire)
 - ▣ `-t` : trie par date de dernière modification (sinon la liste des fichiers est triée par défaut par ordre alphabétique), les plus récemment modifiés en premier
 - ▣ `-i` : affiche au début de la ligne le numéro de i-node des fichiers
 - ▣ `-R` : listage récursif des sous-répertoires
 - ▣ `-g` : affiche le groupe du fichier quand elle est associée à l'option `-l`

Informations associées aux fichiers avec ls

11

```
total 36
-rw-r--r-- 1 tichit tichit  9145 2009-04-03 11:44 Test.txt
-rw-r--r-- 1 tichit tichit 21702 2009-04-03 11:50 Titi.pl
```

- Les informations fournis sont dans l'ordre
 - ▣ Type du fichier et protection
 - ▣ Nombre de liens sur le fichier
 - ▣ Le nom du propriétaire
 - ▣ Le nom du groupe auquel appartient le propriétaire
 - ▣ Taille du fichier en octet
 - ▣ Date et heure de la dernière modification
 - ▣ Nom du fichier

Exemples avec ls

12

- ❑ `ls` donne la liste des fichiers du répertoire courant
- ❑ `ls nom_rep` donne la liste des fichiers du répertoire `nom_rep`
- ❑ `ls a*` donne la liste des fichiers commençant avec 'a'

Options utiles

- ❑ `ls -l` liste tous les attributs des fichiers (taille, dates, ...)
- ❑ `ls -t` liste par date de dernière modification
- ❑ `ls -a` liste tous les fichiers y compris les fichiers cachés
(commençant par .)
- ❑ `ls -R` affiche le contenu de tous les sous-répertoires récursivement

la commande **cd**/ la commande **pwd**

13

- Permet de Se déplacer dans les répertoires : **cd (change directory)**
 - ▣ **cd** sans arguments retourne dans le répertoire principal (identique à **cd ~**)
 - ▣ **cd rep** pour aller dans le sous répertoire rep
 - ▣ **cd . .** Pour remonter dans le répertoire père
 - ▣ Les répertoires peuvent êtres désignés à l'aide de leurs chemins d'accès (Path) absolu (commençant par la racine) ou relatif (à partir du répertoire courant)
- Savoir où on est : **pwd** : La commande **pwd (print working directory)** retourne le chemin absolu du répertoire courant.

La commande `mkdir`

14

Créer un répertoire : `mkdir`

- La commande `mkdir` (make directory) permet de créer un répertoire.
- `mkdir toto` : crée le répertoire `toto` dans le répertoire courant ;
- `mkdir ../toto` : idem dans le répertoire parent.
 - ▣ Le répertoire représenté par « `.` » désigne le répertoire courant
 - ▣ Le répertoire représenté par « `..` » désigne le répertoire père
- La commande `rmdir` (remove directory) permet de détruire un répertoire s'il est vide
- Pour supprimer un répertoire non vide on utilise `rm -R`

Création d'un fichier/consulter contenu

15

- Création par un éditeur/application ou commande touch
- **touch**
 - ▣ Modifie la date du dernier accès aux fichiers donnés en paramètre
 - ▣ Cré un fichier vide sinon
 - ▣ **touch file1 file2** : crée les 2 fichiers vides
- Pour consulter un fichier texte,
 - ▣ Commandes **cat** (texte court) ou **less** (texte long) ou more ...
 - ▣ Ou bien utiliser un éditeur exemple : **vi**, **emacs**, **gedit**, **vim** etc...
- **File [-f]** fichier: décrit le contenu d'un fichier

La commande **cat**

- Lit de l'entrée standard ou d'un fichier ou d'un pipe et affiche sur la sortie standard
- Concatène et affiche le contenu de 2 fichiers

```
nano file1.txt □ Bonjour
```

```
nano file2.txt □ Salut
```

```
$ cat
```

```
$ cat file1.txt
```

```
$ cat file2.txt
```

```
$ cat file1.txt file2.txt > newfile.txt
```

```
$ cat new.txt
```

more et less

17

- **less** visualiser un fichier texte page par page (sans le modifier). similaire à more, mais permet de revenir en arrière ou de rechercher une chaîne.
- Contrairement à vi (qui permet aussi de visualiser des fichiers), **less** n'a pas besoin de charger entièrement le fichier en mémoire et s'ouvre donc très rapidement même pour consulter de gros fichiers.
- **more** est une commande standard sous Unix utilisée pour voir (mais pas modifier) le contenu d'un fichier texte, page par page.

```
more +20 mypass //affiche a partir de la ligne 20
```

Copie de fichiers

18

- **cp** (copy, copier fichiers et répertoires)
 - ▣ **cp [option] source destination**
 - copie d'un seul fichier, en précisant le chemin et le nom du fichier destination
 - ▣ **cp [option] rep-source rep-cible**
 - copie l'ensemble des fichiers dans le rép. spécifié, en gardant les noms
- Principales options :
 - ▣ **-R** recopie récursive, permet de copier toute une arborescence
 - ▣ **-i** avertit l'utilisateur de l'existence d'un fichier du même nom et lui demande s'il veut le remplacer
 - ▣ **-p** effectue une copie en gardant le propriétaire et le groupe d'origine
 - ▣ **-v** affiche en clair le nom des fichiers copiés

Suppression de fichiers

19

- **rm** (remove, supprimer des fichiers)

- ▣ **rm [option] fichiers**

- ▣ L'option **-i** a été ajoutée d'office dans un alias pour demander confirmation à l'utilisateur, pour chaque fichier.

- **Options**

- ▣ **-r** : permet de supprimer un répertoire et ses sous répertoires (attention TRÈS dangereux)

- ▣ **-f** (force) : permet de supprimer les fichiers protégés en écriture **sans** demande de confirmation. Cela permet d'inhiber l'option **-i** et de gagner du temps .. sauf dégâts !

Attention, sous Unix, par défaut, un fichier détruit ne peut pas être récupéré

Déplacer un fichier

20

- **mv** (move, renommer ou déplacer)

- ▣ **mv [option] source destination**

- ▣ renomme le fichier source, ce qui est juste le changement de nom (de chemin)...

- ```
mv toto titi
```

- ▣ **mv [option] fichiers répertoire**

- déplace les fichiers sources dans le répertoire, en gardant les noms.

- ```
mv /home/jean/images/*.jpg /tmp/
```

- Principales options :

- ▣ **-b (b=backup)** : effectue une sauvegarde des fichiers avant de les déplacer. La copie porte le même nom suivi d'un tilde

- ▣ **-i (i=interactive)** : demande confirmation avant pour chaque fichier

- ▣ **-u (u=update)** : pour ne pas supprimer le fichier si sa date de modification est postérieure à celle du fichier remplaçant

21

Gestion des droits d'accès

Utilisateurs, Groupes, Permissions

Les utilisateurs et les groupes

22

Un usager possède :

- Un login
- Un numéro d'identification (UID, *User IDentification*)
- Un nom et prénom
- Un mot de passe
- Un nom de groupe principal
- Un numéro de groupe d'appartenance (GID, *Group IDentification*)
- Un répertoire d'accueil
- Un shell par défaut

De plus :

- Unix permet de regrouper des utilisateurs en groupes.
- Un groupe possède certains privilèges.
- Un utilisateur peut appartenir à plusieurs groupes.

Root

23

Il existe un compte spécial :

- login = root
- group = root
- HOME = /root
- UID = 0
- GID = 0

Ce compte à tous les droits. C'est le *super-utilisateur*. Les administrateurs de la machine ont accès à ce compte.

Les utilisateurs et les groupes

24

- Les informations relatives aux utilisateurs sont stockées dans le fichier **/etc/passwd**.

```
tichit@iml230:~$ cat /etc/passwd | grep tichit
```

```
tichit:x:1000:1000:Tichit Laurent,,,:/home/tichit:/bin/bash
```

- La commande **id** permet de connaître les informations relatives à un utilisateur :

```
tichit@iml230:~$ id
```

```
uid=1000(tichit) gid=1000(tichit) groupes=1000(tichit),  
4(adm),20(dialout),21(fax),24(cdrom),25(floppy),  
26(tape),29(audio),30(dip),44(video),46(plugdev),  
104(scanner),112(netdev),113(lpadmin),115(powerdev),  
117(admin),119(fuse)
```

Les utilisateurs et les groupes

25

- Les informations relatives aux groupes sont stockées dans le fichier **/etc/group** :
tichit@iml230:~\$ **cat /etc/group | grep tichit**

root

26

- La commande **su [login]** permet de devenir *root* (ou l'utilisateur *login*).
- Sur les distribution Linux récentes, le compte *root* est parfois désactivé.

Dans ce cas, on se servira de la commande **sudo [login] commande**, au cas pas cas, qui permet de devenir *temporairement root* (ou l'utilisateur *login*) pour exécuter la commande.

Droits d'accès

27

- Il existe 3 manières pour accéder à un fichier
 - ▣ Lire son contenu (**r**ead)
 - ▣ Modifier son contenu (**w**rite)
 - ▣ S'il s'agit d'un fichier ou d'un script, l'exécuter (**x**ecute)

- 3 catégories ou groupements d'accès à un fichier
 - ▣ Le propriétaire di fichier (**u**ser)
 - ▣ Les membres du groupe auquel appartient le fichier (**g**roup)
 - ▣ Tous les autres usagers (**o**thers)

Droits d'accès

28

- La commande `ls -l` permet de connaître les droits associé à chaque fichier la liste

```
tichit@iml230:~/Test$ ls -l
```

```
total 36
```

```
-rw-r--r-- 1 tichit tichit 9145 2009-04-03 11:44 Test.txt
```

```
-rw-r--r-- 1 tichit tichit 21702 2009-04-03 11:50 Titl.pl
```

- Les droits d'accès sont exprimé par 3 groupements de 3 caractères

rwX



User

rwX



Group

rwX



Other

Droits d'accès

29

- **r** : droit lire **w**: droit d'écrire **x**: droit d'exécution
 - : ce signe à la place de l'un de ces caractères exprime l'interdiction
- Seul le propriétaire du fichier peut modifier les droits d'accès à un fichier par la commande

chmod mode nom_de_fichier
- On peut modifier les permissions de 2 manières:
 - ▣ De manière absolue (indépendamment des droits actuels)
 - ▣ De manière relatives (en modifiant les permissions courantes)

Permissions absolues

30

Elles sont exprimées en octal. Si l'on désire uniquement que :

- Le fichier soit lisible
 - ▶ Par le propriétaire → 400
 - ▶ Par les membres du groupe → 040
 - ▶ Par les autres → 004
- Le fichier soit modifiable
 - ▶ Par le propriétaire → 200
 - ▶ Par les membres du groupe → 020
 - ▶ Par les autres → 002
- Le fichier soit exécutable
 - ▶ Par le propriétaire → 100
 - ▶ Par les membres du groupe → 010
 - ▶ Par les autres → 001

Permissions absolues

31

Bien entendu, on peut composer les permissions. Il suffit d'additionner les valeurs.

- **chmod 644 fichier.txt** : propriétaire peut lire et modifier, groupe et autre peuvent uniquement lire.
- **chmod 755 fichier.pl** : idem plus tous peuvent exécuter.
- **chmod 710 fichier.sh** : le propriétaire a tous les droits, les membres du groupe peuvent uniquement exécuter, les autres n'ont aucun accès.

Permissions relatives (Symbolique)

32

En symbolique le mode est de la forme :

chmod “qui” “quelle action” “quelle permission” fichier
“Qui”

- **u** : user
- **g** : group
- **o** : other
- **a** : all

“Quelle action”

- **+** : ajoute des permissions
- **-** : retire des permissions

Ex :

- **chmod o-rw fichier.txt** : enlève les permissions en lecture et écriture pour les autres.
- **chmod a+x fichier.pl** : donne à tous le droit d'exécuter.
- **chmod go-w fichier.sh** : enlève aux membres du groupe et aux autres le droit de modifier.

Commandes et filtres usuels

- Filtres: `cat`, `head`, `tail`, `join`, `paste`, `cut`, `grep`, `look`, `lpr`, `more`, `pg`, `sort`, `touch`, `tr`, `uniq`, `wc`
- `Grep` et expressions régulières

Commandes et Filtres usuels

34

- ❑ Commandes utiles pour des traitements sur des données de type texte organisées en tableau
- ❑ Certains fichiers sont organisés en enregistrements (records) de champs (Fields)
 - ❑ Chaque ligne décrit une entité : personne, chose...
 - ❑ Une entité est définie par différents champs.
 - ❑ Les champs sont séparés par un caractère particulier
- ❑ Exemples
 - ❑ des fichiers de configuration du système : `/etc/passwd` `/etc/crontab`
 - ❑ des fichiers CSV = **données structurées en lignes** :
 - 1 ligne = 1 entité (cf OMGL) les colonnes = les attributs de l'entité

Commandes et Filtres usuels

35

- Extraire des informations dans des textes
 - ▣ compter les occurrences d'une information
 - ▣ comparer différentes informations
 - ▣ faire des classements, des sélections ...

- Pour aller plus loin
 - ▣ Il existe des outils programmables : sed, awk, perl
 - ▣ Exécute une opération sur les données
 - ▣ Sortie (pipe): standard ou fichier

Commandes et Filtres usuels

36

Parmi les filtres usuels:

```
cat, head, tail, join, paste, cut, grep,  
look, lpr, more, pg, sort, touch, tr,  
uniq, wc
```

Catégories de filtres

37

- Sélection d'une partie du texte
 - ▣ le début, la fin
 - ▣ des colonnes, des champs
 - ▣ certaines lignes contenant des chaînes précises
- Comptage
 - ▣ des lignes, mots, caractères
 - ▣ des exemplaires des mêmes données
- Classement des lignes Divers
 - ▣ transformation des caractères, mise en page...

La commande **head**

38

- Permet d'afficher le début d'un fichier, ou de l'entrée standard en spécifiant le nombre de lignes [option `-n`] ou le nombre de caractères [option `-c`] souhaités
- Sans options la commande affiche un certain nombre de ligne par défaut
- **syntaxe**

```
$ head [options] [fichier...]
```

- Exemples:
- `$ head -c 1000 /etc/passwd` : affiche les 1000 premiers caractères du fichiers `/etc/passwd`
- `$ head -n 10 /etc/passwd` affiche les 10 premières lignes du fichier

La commande tail

39

- ❑ Permet d'afficher les dernières lignes d'un fichier, ou de l'entrée standard en spécifiant le nombre de lignes [option `-n`] ou le nombre de caractères [option `-c`] souhaités ou même le nombre de blocs de (512 octets) avec [option `-b`]
- ❑ Sans options la commande affiche un certains nombre de ligne par défaut
- ❑ Syntaxe `$ tail [options] [fichier...]`
- ❑ Exemples:
- ❑ `$ tail - c 1000 /etc/passwd`
affiche les 15 derniers caractères du fichiers `/etc/passwd`
- ❑ `$ tail -n 5 /etc/passwd`
affiche les 5 dernières lignes du fichier
- ❑ `$tail -n +5 /etc/passwd` affiche la fin de `/etc/passwd` à partir de la 5^{ème} lignes

Join/paste

40

- Fusionner les lignes de deux fichiers

```
$ join [options] fichier_1 fichier_2
```

- Regrouper les lignes de différents fichiers
- concatène les lignes de même numéro de 2 fichiers
- L'option `-d` peut être utilisée pour spécifier un délimiteur, par défaut c'est la tabulation

```
$ paste [options] [fichier...]
```

La commande cut

41

- La commande cut permet d'afficher des zones spécifiques d'un fichier.

Par exemple : `cut -c1 /etc/passwd`

affichera la première colonne du fichier `/etc/passwd`.

Il existe d'autres spécifications

`-c1-5` Permet de sélectionner les colonnes 1 à 5.

`-c14-` Permet de sélectionner de la colonne 14 à la dernière.

`-c1-3,14-18` Permet de spécifier plusieurs plages de colonnes

- On peut également spécifier un *séparateur de champs* avec l'option `-d`.

`cut -d: -f6 /etc/passwd`

affichera le 6ème champ du fichier `/etc/passwd`, dont le séparateur de champs est le caractère double point (`:`:``).

La commande wc

42

- `wc` compte le nombre d'octets(bytes), le nombre de caractères blancs séparant un mot ainsi que les nouvelles lignes dans chaque fichier ou à partir de l'entrée standard
- Syntaxe :
`wc [option] ... [file] ...`
- Exemples

La commande uniq

43

- Comptage des lignes identiques `uniq` Affiche le contenu du fichier mais Donne un seul exemplaire des lignes adjacentes qui sont identiques.
- Commande `uniq`
 - ▣ sans option : enlève les lignes successives identiques
 - ▣ option `-c` : compte les lignes successives identiques
- Exemple, fichier `fruits` :
 - ▣ `uniq < fruits`
 - ▣ `uniq -c < fruits`
 - ▣ `sort < fruits | uniq -c`

```
poire  
pomme  
poire  
banane  
banane  
pomme  
pomme
```

La commande sort

44

- Effectuer des tri sue les lignes d'un flux donné en paramètre
 - ▣ -n numérique
 - ▣ -d dictionnaire
 - ▣ -k colonne
 - ▣ -t tabulation
 - ▣ -r inversé
 - ▣ -o Option output to a new file,
- Exemples
 - ▣ `sort -n /etc/passwd`
 - ▣ `Sort -nt: -k3 < /etc/passwd`
 - ▣ `Sort -nrt: -k3 < /etc/passwd`

La commande sort exemple

45

```
anissa@anissa-VirtualBox:~$ cat file.txt
```

```
xxxxxx
```

```
gggggg
```

```
hhhhh
```

```
kkkkkk
```

```
llllll
```

```
aaaaaa
```

```
BBBBBB
```

```
bbbbbb
```

```
CCCCCC
```

```
CCCCCC
```

```
anissa@anissa-VirtualBox:~$ sort file.txt
```

```
aaaaaa
```

```
bbbbbb
```

```
BBBBBB
```

```
CCCCCC
```

```
CCCCCC
```

```
gggggg
```

```
hhhhh
```

```
kkkkkk
```

```
llllll
```

```
xxxxxx
```

La commande sort

46

- **Equivalentes**

- `$ sort inputfile.txt > filename.txt` (équivalente)

- `$sort -o filename.txt inputfile.txt`

- **-r tri inverse:** `$ sort -r inputfile.txt`

- **-n tri numérique:** `$ sort -n filename.txt`

- **-nr tri numérique inversé:** `$ sort -nr filename.txt`

- **-k :** k introduit le numéro de colonne de tri: exemple “-k 2” désigne la seconde colonne :

- `$ sort -k2n filename.txt`

- **-c option :** This option is used to check if the **file given is already sorted or**

- **-u option :** To **sort and remove duplicates**

- **-M Option :** To **sort by month** pass the -M option to sort.

La commande tr

47

Tr : effectue des remplacements de caractères
`tr [option]... set1 [set2]`

Substitution ou suppression de caractères sélectionnés

Exemple : `$cat file1.txt | tr "[A-Z]" "[a-z]" > file2.txt`

Substituer les majuscules du fichier file1.txt en minuscules et enregistrer le résultat dans file2.txt