Outils Numériques

Ievgen Redko

Université Jean Monnet, Saint-Etienne

Plan

Plan

Plan

- multi-tâches, multi-utilisateurs
- créé le 1/1/1970 à 0 h 0 mn 0 s
- Université de Berkeley et ATT
- 2 grandes familles : BSD (Berkeley) et System V (ATT)
- normes : POSIX (Portable Operating System Interface)
- libres : Linux, FreeBSD, Android
- propriétaires : des centaines...

Plan

En UNIX, des **utilisateurs** exécutent des **commandes** qui agissent sur des **fichiers**.

Entrées/sorties : fichiers spéciaux

Plan

Nature

- humain :
  - **root** et les autres
  - **root**, les **sudoers** et les autres
- . . . ou pas :
  - robots : messagerie, sauvegarde, . . .
  - démons : interface disques, matériel, . . .

Informations sur l'utilisateur

Connu par :

- son identifiant (ou *UID* : User Identification Number)
- son nom de connexion (ou *login*)
- son groupe (ou *GID* : Group Identification Number)
- son mot de passe (seul le mot de passe **crypté** est stocké)
- son répertoire d'accueil
- son interpréteur de commandes (sh, csh, tcsh, ksh, zsh, bash, ...)

Plan

Plan

Définitions

Fichier

Un fichier est une suite d'octets que le système peut lire ou écrire

Système de fichier

Un système de fichiers est une façon d'organiser et de stocker une arborescence sur un support (disque dur, CD, clé USB, réseau, . . .).

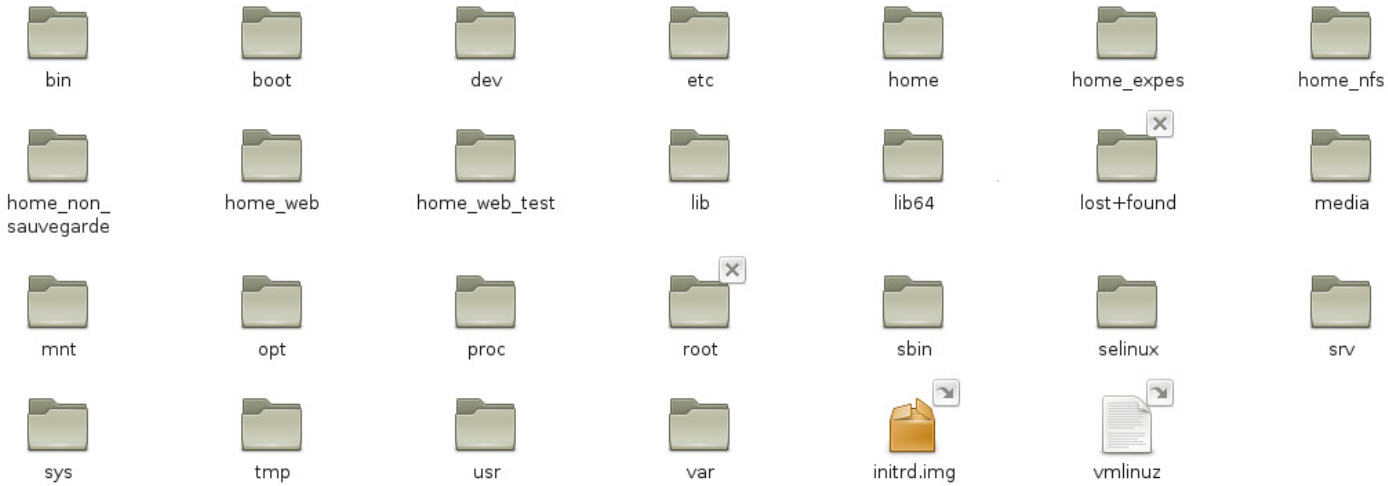
Exemples

- FAT32, FAT64, NTFS, ISO 9660 (Windows, LINUX, MacOs)
- NFS (Windows, LINUX, MacOS)
- HFS(+) (MacOS, LINUX)
- EXT3, EXT4, ReiserFS (LINUX)

Types de fichiers UNIX

- les fichiers réguliers : programmes, textes, images, sons, ...
- les répertoires
- les fichiers spéciaux liés aux périphériques : bloc ou caractère
- les liens symboliques
- les sockets : branchements réseau
- les tubes : communications inter-programmes

L'arborescence des fichiers sous Linux



Plus précisément . . .

/usr : applications (exécutables, documentations, librairies,. . .)

/home : utilisateurs

/var : variable : le mél non encore lu, les fichiers en attente d'impression, les fichiers de compte rendu (logs). . .

/etc : configuration du système : réseaux, mot de passe, imprimantes. . .

/dev : fichiers spéciaux qui permettent la communication avec les périphériques

/tmp : zone de stockage **temporaire**

/root : le répertoire d'accueil du super-utilisateur

Plan

Propriétaire d'un fichier

Tout fichier est possédé par un **propriétaire** (un utilisateur). Du point de vue du fichier :

- u : l'utilisateur normal, son propriétaire (souvent le créateur du fichier)
- g : le groupe du propriétaire
- o : tous les autres utilisateurs ou le reste du monde

Les droits d'accès

- r : accès en lecture
- w : accès en écriture
- x : accès en exécution

Remarque

Les droits ont des significations différentes en fonction du type de fichier

Affichage des droits

- 10 caractères
- premier : type du fichier (`-`, `d`, `b`, `c`, `l`, `s`, `p`)
- 3 suivants : droits `rwx` du propriétaire
- 3 suivants : droits `rwx` des membres du groupe du propriétaire
- 3 suivants : droits `rwx` du reste du monde (Internet. . .)

À retenir

$$\underbrace{d}_{\text{type}} \quad \underbrace{rwx}_{\text{utilisateur}} \quad \underbrace{rwx}_{\text{groupe}} \quad \underbrace{rwx}_{\text{autres}}$$

Les droits sur les fichiers réguliers

- r : afficher/lire le contenu du fichier (`r` ou -)
- w : modifier le contenu (`w` ou -)
- x : lancement du programme (`x` ou -)

Exemple

```
-rw-r-xr--
```

Les droits sur les répertoires

- r : lire le contenu du répertoire (liste des fichiers)
- w : modifier le contenu (créer ou supprimer des fichiers)
- x : accéder au répertoire et s'y déplacer (si on attribue *w*, il faut attribuer *x*)

Exemple

```
dr-x-wxr--
```

Plan

Le terminal et le shell

- En UNIX, interaction avec le système pas seulement graphique (*remember* WYSIWYG...)
- Possibilité de lancer un terminal muni d'un interpréteur de commandes (**shell**).
- Le shell interprète des commandes qui permettent d'interagir avec le système.

Le shell

```
user - machine> ls -l toto*
```

Prompt

`user - machine>` est appelée *invite* de commande (ou *prompt*). Elle est configurable et ne fait pas partie de la commande

Syntaxe des commandes

Les commandes suivent la syntaxe suivante :

Nom `ls`

Options `-l`

Arguments `toto*`

Le shell : quelques trucs et astuces...

- copier/coller : bouton gauche/bouton milieu de la souris
- flèches HAUT et BAS : navigation dans l'historique de commandes
- touche TAB : complétion automatique de noms
- pour connaître la nature du **contenu** d'un fichier : commande file
- pour savoir ce qu'on exécute **réellement** : commande type
- fichier de configuration du shell : .bashrc (fourni...)

Commandes de base : aide en ligne

```
man <commande>
```

```
apropos <mot-clé>
```

Options fréquentes :

- `--help, -h` affiche un bref descriptif de la commande.
- `--version` affiche la version du programme.

Plan

Commandes de base : désignation des fichiers

2 façons de désigner un fichier :

1. chemin **absolu** : chemin de la racine du SGF jusqu'au fichier. Commence par /
2. chemin **relatif** : chemin du répertoire courant jusqu'au fichier. Commence par autre chose que /

Exemples

- `/home/redko/Sujets/L1/OL/decembre2014.tex`
- `../../L1/OL/decembre2014.tex` (depuis le répertoire `/home/redko/Sujets/L3/Langages`)

Commandes de base : recopie de fichiers

Version simple

```
cp <fic1> <fic2>
```

Copie $< fic_1 >$ en $< fic_2 >$.

Version générale

```
cp <fic_ou_rép> [<fic_ou_rép>...] <rép>
```

Copie les arguments (fichiers ou répertoires) dans le répertoire cible final

Commandes de base : déplacement ou renommage

Version simple

```
mv <fic1> <fic2>
```

Renomme $< fic_1 >$ en $< fic_2 >$.

Version générale

```
mv <fic_ou_rép> [<fic_ou_rép>...] <rép>
```

Déplace les arguments (fichiers ou répertoires) dans le répertoire cible final

Remarque

C'est le *Drag & Drop* des gestionnaires graphiques...

Commandes de base : changement de mode

```
chmod [-R] <mode> <fic_ou_rép> [<fic_ou_rép>...]
```

Change les droits d'accès au fichier(s) ou au répertoire(s).

<mode> de la forme ugoa±rwx :

- ugoa : La première partie indique à qui s'applique le changement (u pour utilisateur, g pour le group, o (*others*) pour les autres, a est équivalent à ugo).
- ± : + ajoute le droit, − enlève le droit
- rwx : le(s) droit(s) concerné(s)

Option R : changement récursif (descend dans les répertoires)

Commandes de base : changement de répertoire courant

```
cd [<rép>]
```

(Change Directory)

Sans argument, le répertoire de travail de l'utilisateur est sélectionné.

Le répertoire ~login représente le répertoire de travail de l'utilisateur login.

Commandes de base : répertoires spéciaux

Dans chaque répertoire, il existe deux répertoires spéciaux nommés (“.” et “..”).

. représente le répertoire courant, .. le répertoire père.

Exemples

- `cd .` ne fait **rien**
- `cd ..` remonte d'un cran dans l'arborescence.

Commandes de base : liste des fichiers

```
ls [-alF] [<fic_ou_rép>...]
```

(LiSt)
Sans argument, le contenu du répertoire courant est listé.

Commandes de base : liste des fichiers, options

```
ls [-alF] [<fic_ou_rép> ...]
```

- -a afficher les fichiers «cachés» (dont le nom commence par «.»)
- -l donner des informations supplémentaires sur les fichiers (type, permission, propriétaire, taille, ...).
- -F accoler au nom un caractère rappelant le type du fichier(/ pour les répertoires, * pour les fichiers exécutables, ...).

Commandes de base

Exemple de listing avec `ls -l`

```
ls -l
```

```
total 0
drwx---@ 3 redko staff 96 Aug 29 14:31 Applications
drwx---+ 7 redko staff 224 Oct 7 17:57 Desktop
drwx---+ 8 redko staff 256 Oct 3 19:12 Documents
drwx---+ 4 redko staff 128 Oct 7 17:57 Downloads
drwx---@ 13 redko staff 416 Oct 2 15:37 Dropbox
drwx---@ 64 redko staff 2048 Sep 30 16:10 Library
drwx---+ 3 redko staff 96 Jan 1 2018 Movies
drwx---+ 4 redko staff 128 Sep 25 11:52 Music
drwx---+ 3 redko staff 96 Jan 1 2018 Pictures
drwxr-xr-x+ 4 redko staff 128 Jan 1 2018 Public
```

Commandes de base : création et destruction

```
mkdir <rép> [<rép>...]
```

Crée les répertoires passés en arguments. L'option `-p` peut s'avérer intéressante (voir le TP).

```
rmdir <rép> [<rép>...]
```

Supprime les répertoires passés en arguments (uniquement s'ils sont vides).

Commandes de base : suppression

```
rm [-fir] <fic_ou_rép> [<fic_ou_rép>...]
```

Supprime les fichiers passés en arguments. **Attention**, la suppression est définitive, aucun système de corbeille n'existe pour les fichiers ainsi supprimés.

Commandes de base : suppression, options

```
rm [-fir] <fic_ou_rép> [<fic_ou_rép>...]
```

Supprime les fichiers passés en arguments.

- -f effectuer la suppression sans interaction avec l'utilisateur (pas de confirmation, pas de message d'erreur). À utiliser avec **prudence**.
- -i demander confirmation pour chaque argument.
- -r supprimer les fichiers *récursivement i.e.* de supprimer un répertoire non vide. À utiliser avec **prudence**.

Commandes de base : affichage

```
more <fic>
```

```
less <fic>
```

Permet d'afficher un fichier page par page et de se déplacer à l'intérieur.

Commandes de base : recherche de motif

```
grep [-civ] <motif> [<fic>...]
```

n'affiche que les lignes correspondant au motif défini.

- -c n'affiche pas les lignes, les compte seulement.
- -i ignore les différences minuscules/majuscules.
- -v inverse le motif, et affiche uniquement les lignes **ne** correspondant **pas** au motif.

Exemples

1. `grep section monbeautexte.tex` affiche les lignes du fichier `monbeautexte.tex` comportant le motif `section`
2. `grep toto *`

Commandes de base : statistiques

```
wc [-cwl] [<fic> ...]
```

affiche le nombre de lignes, de mots et de caractères du fichier

- -c n'affiche que le nombre de caractères.
- -w n'affiche que le nombre de mots.
- -l n'affiche que le nombre de lignes.

Exemple

```
redko> wc OL_UNIX.tex
  701  1878 19921 OL_UNIX.tex
```

Commandes de base :

```
cat [<fic> ...]
```

affiche le contenu de tous les fichiers passés en paramètre.

Plan

Principe de réécriture

Exécution d'une commande shell en deux étapes :

1. réécriture de la commande avec remplacement des caractères spéciaux par ordre alphabétique en fonction des fichiers du répertoire courant (cf. tableau suivant)
2. exécution de la commande réécrite

Principe de réécriture

`?` : **un** caractère quelconque

`*` : un nombre quelconque de caractère(s) quelconque(s)

`[ab]` : soit `a` soit `b`

Principe de réécriture

Nous nous trouvons dans un répertoire contenant uniquement 4 fichiers : `toto1`, `toto2`, `toto3`, `toto24`.

```
ls toto?
```

se réécrit en

```
ls toto1 toto2 toto3
```

Principe de réécriture

Nous nous trouvons dans un répertoire contenant uniquement 4 fichiers : `toto1`, `toto2`, `toto3`, `toto24`.

```
ls tot*
```

se réécrit en

```
ls toto1 toto2 toto24 toto3
```

Principe de réécriture

Nous nous trouvons dans un répertoire contenant uniquement 4 fichiers : `toto1`, `toto2`, `toto3`, `toto24`.

```
ls toto[12]?
```

se réécrit en

```
ls toto24
```

Principe de réécriture

Un nom de fichier peut contenir un caractère spécial, dans ce cas pour y accéder il faut précéder ce dernier par un \.

```
ls toto*
```

affiche tous les fichiers commençant par toto

```
ls toto\*
```

affiche l'unique fichier toto$*$ s'il existe

Plan

Rédirection de sortie

```
commande > fichier
```

écrit le résultat de `commande` dans `fichier`

Exemples

1. `ls -l > toto`
2. `cat toto1 toto2 > toto`
3. `cat > toto4`

Rédirection d'entrée

```
commande < fichier
```

`commande` prend ses arguments dans `fichier`

Exemple

```
more < toto
```

Enchaînement : le tube (ou pipe)

```
commande1 | commande2
```

le résultat de `commande1` est passé comme argument à `commande2`

Exemples

1. `ls | more` affiche la liste des fichiers page par page
2. `cat toto1 toto2 | wc -l`