Système d'exploitation

Cours 2

Jean-Sébastien Coron

Université du Luxembourg

Plan du cours

- Syntaxe d'une commande et redirection entrées/sorties.
- Le système de fichier UNIX.

Syntaxe d'une commande

- Commande [option] [argument]
  - Une commande peut comprendre des options qui modifient son comportement.
  - Elle peut comprendre des arguments (par exemple nom de fichier).
  - Exemple: `ls -l fichier` donne des informations plus longues sur `fichier`.
- Format de [option]: -l [arg]
  - L'option peut aussi avoir un argument, qui est facultatif.

Redirection entrées/sorties

- Par défaut, une commande écrit dans la sortie standard (l'écran) et prend en entrée l'entrée standard (le clavier).
- On peut rediriger la sortie d'une commande dans un fichier.
  - Exemple: `ls -l > liste`. Stocke dans le fichier `liste` la liste des fichiers.
- On peut rediriger l'entrée d'une commande à partir d'un fichier:
  - `cat < input`: la commande lit le fichier `input` et l'imprime, au lieu de lire le clavier.

Le système de fichier UNIX

- Permet de stocker toutes les informations nécessaires au bon fonctionnement du système.
  - Fichiers ordinaires: contiennent du texte, des données ou des programmes.
  - Répertoires: contiennent des fichier ou d'autres répertoires.
  - Périphériques: elles sont gérées comme des fichiers.
  - Liens: un lien est un pointeur sur un autre fichier.
    - Lien dur: un lien dur sur un fichier est indistingable du fichier lui-même.
    - Lien symbolique: c'est un raccourci sur un autre fichier.

Structure des fichiers UNIX

- Organisé en une arborescence avec racine unique: le répertoire racine `/`.
- Spécifier une location:
  - Chemin absolu: `/home/enseign/cours/`
  - Chemin relatif à partir du répertoire courant.
    - "`.`" Répertoire courant.
    - "`..`" Répertoire parent.
    - Exemple: pour accéder à `/home/enseign/cours/premier` depuis `/home/enseign/exo`, on utilise `../cours/premier`.

Organisation typique

- `/`: le répertoire racine.
- `/bin`: utilitaires de bas-niveau
- `/usr/bin`: utilitaires plus évolués.
- `/sbin/`: répertoire de l'administrateur.
- `/lib`: librairies de programmes pour les utilitaires de bas-niveau.
- `/usr/lib`: librairies de programmes pour les utilitaires de haut-niveau.
- `/home`: répertoire pour les utilisateurs. Chaque sous-répertoire est nommé d'après le login.
- `/dev`: périphériques.

Manipulation de fichiers

- `pwd`, `cd`, `ls`, `ls -l`, `ls -a -l`.
- Informations sur un fichier:
  - `drwxrwxrwx 2 Administ None 4096 May 9 10:52 www`
  - `d`: type de fichier. d=directory, `-`=fichier, `l`=lien.
  - `rwxrwxrwx`: permissions.
  - `2`: liens sur ce fichier.
  - `Administ`: propriétaire.
  - `None`: groupe.
  - `4096`: taille du fichier.
  - `May 9 10:52`: date de dernière modification.
  - `www`: nom du fichier ou répertoire.

Autres commandes

- `mkdir`: création d'un répertoire.
- `rmdir`: suppression d'un répertoire.
- `cp source destination`: copie d'un fichier.
- `mv source destination`: renommer ou déplacer un fichier.
- `rm`: supprimer un fichier.
- `cat`: affiche le contenu d'un fichier.
- `more`, `less`: affiche le contenu d'un fichier avec des pauses.

Liens durs/symboliques

- `ln filename linkname`: crée un lien depuis le fichier ou répertoire `linkname` sur `filename`.
  - Les deux fichiers sont identiques. Si un des deux est modifié, l'autre aussi.
- `ln -s filename linkname`: crée un raccourci nommé `linkname` sur `filename`.
- Différence:
  - Un lien dur ne peut être crée que sur le même disque physique.
  - Pas de lien dur sur des répertoires.

Spécifier plusieurs fichiers

- On utilise des méta-caractères:
  - `?` : remplace un seul caractère.
  - `*` : remplace zéro, un ou plusieurs caractères.
  - `[ ]` : remplace un des caractères dans [].
- Exemple:
  - `*` : tous les fichiers.
  - `*.*` : tous les fichiers avec un ".".
  - `?are?` : tous les fichiers de 5 caractères avec `are` au milieu
  - `[m-z]*` : tous les fichiers commançant par une lettre entre m et z.

Exemple

- Afficher tous les fichiers du répertoire courant:
  - `$ ls *`
    `fich1.c fich2.c prog.c toto.txt`
- Afficher tous les fichiers qui se terminent par `.c`.
  - `$ ls *.c`
    `fich1.c fich2.c prog.c`
- Afficher tous les fichiers qui commencent par `fich`
  - `$ ls fich*`
    `fich1.c fich2.c`