Informatique MEE - Shell

Laurent Signac – CC-BY-SA – 16-07-20 2045 388d0e8b5893eb619edd

1 Shell ?

Le shell est une interface homme machine en ligne de commande pour accéder aux fonctionnalités du système. Par extension, on parle de shell aussi pour accéder aux fonctionnalité d'un langage de programmation (shell Python).

2 Shell Unix : bash

Nous parlons ici de shell Unix, interface privilégiée pour accéder à ce système. Il en existe plusieurs : sh, C-shell, Korn shell, Bourne again shell… Le plus courant sur les système GNU/Linux et la Raspberry Pi est le Bourne again Shell ou bash.

Quelques caractéristiques de la plupart des shells :

1. Lors de la saisie d'une commande, vous pouvez accéder à un historique, éditer la ligne de commande, utiliser la complétion automatique.
2. Une fois la commande entrée, certaines chaînes de caractères sont substituées (variables, wild cards…)
3. Le shell exécute la commande résultante

3 Système de fichiers

3.1 Arborescence

Les type de systèmes de fichiers ainsi que la manière dont ils apparaissent à l'utilisateur varient selon les systèmes d'exploitation. Sous Unix, les systèmes de fichiers se présentent sous la forme d'une arborescence unique (il n'y a pas d'unité C: ou D:…), contenant tous les fichiers, et dont la racine est notée `/`.

Un fichier peut donc être désigné par son chemin absolu dans l'arborescence.
Par exemple : `/etc/systemd/system.conf` désigne le fichier `system.conf` situé dans le répertoire `systemd` lui même situé dans `etc` lui même situé à la racine.

Lorsqu'on interagit avec le shell, on se *trouve* en permanence dans un répertoire appelé **répertoire courant** (la commande `pwd` l'affiche). On peut donc aussi désigner un fichier par son chemin *relatif* par rapport au répertoire courant. Par exemple, si le répertoire courant est `/etc/nginx` le fichier précédent peut être désigné par le chemin relatif : `../systemd/system.conf`. Si on se trouve déjà dans `/etc/systemd`, il peut être désigné simplement par `system.conf`.

Tous les chemins absolus commencent par `/`. Tous les chemins relatifs commencent par autre chose.

3.2 Droits

Les droits permettent d'autoriser ou interdire certaines opérations sur les fichiers et les répertoires. On différencie les droits en lecture, écriture et exécution. Pour un répertoire le droit en lecture permet d'en lister le contenu, le droit en écriture permet de modifier/ajouter/supprimer un fichier dans ce répertoire, et le droit en exécution permet de le traverser.

À chaque fichier et répertoire sont associés un utilisateur (le propriétaire) et un groupe. Les trois permissions précédentes sont précisées, indépendamment pour l'utilisateur associé, le groupe associé, et les autres.

Par exemple, un fichier peut être en lecture, écriture pour le propriétaire, lecture seulement pour le groupe et ne disposer d'aucune permission pour les autres.

Ces permissions sont souvent écrire sous la forme `rw-r-----` (permissions dans l'ordre utilisateur, groupe, autres). On accède aux permissions avec l'option `-l` de la commande `ls`. On change l'utilisateur associé ou le groupe avec les commandes `chown` et `chgrp`. On change les permissions avec la commande `chmod`.

De son côté chaque utilisateur peut appartenir à un ou plusieurs groupes (on obtient la liste avec la commande `groups`).

4 Redirections et tubes

De nombreuses commandes lisent leur entrée sur l'entrée standard (le clavier) et dirigent leur sortie vers la sortie standard (l'écran). Entrée et sortie peuvent être redirigées vers des fichiers ou vers une autre commande.

Le symbole de redirection `>` redirige la sortie vers un fichier. La commande suivante liste le contenu du répertoire courant et met cette liste dans le fichier `liste_fichiers.txt`

```
ls > liste_fichiers.txt
```

Le symbole de redirection `<` redirige l'entrée depuis un fichier. La commande suivante indique combien il y a de lignes dans le fichier `liste.txt` :

```
wc -l < liste.txt
```

La sortie d'une commande peut être redirigée vers l'entrée d'une autre. La commande suivants inscrit dans le fichier `compta.txt` le nombre de lignes affichés par la commande `ls /lib` :

```
ls /lib | wc -l  > compta.txt
```

Lorsqu'on utilise `>` le fichier destination est écrasé. Pour que la sortie soit *ajoutée* au fichier, on utilise `>>`.

5 Commandes utiles :

`ls`	liste des fichiers du répertoire courant
`ls -l`	liste des fichiers du répertoire courant (+ d'infos)
`cd <rep>`	change de répertoire
`cd ..`	remonte dans l'arborescence des répertoires
`mkdir <rep>`	crée un répertoire
`rmdir <rep>`	efface un répertoire
`mv <source> <dest>`	déplace une arborescence
`cp <source> <dest>`	déplace un fichier
`rm <fichier>`	efface un fichier
`chmod u+x <file>`	rend le fichier exécutable (chmod permet de rendre les fichiers accessibles en lecture ou écriture aussi)
`file <fichier>`	donne des informations sur le contenu du fichier
`cat <fichier>`	Envoie le contenu d'un fichier sur la sortie standard
`diff <fichier1> <fichier2>`	Affiche les différences entre 2 fichiers
`strings <fichier>`	Extrait les chaînes de caractères imprimables d'un fichier binaire
`less <fichier>`	Permet de consulter le contenu d'un fichier (on peut naviguer avec les flèches)
`touch <fichier>`	crée un fichier (vide)
`grep <motif>`	sélectionne les lignes de l'entrée standard contenant le motif
`wc`	compte les mots/lignes/caractères sur l'entrée standard
`dpkg -l`	liste des logiciels installés
`apt-get update`	met à jour la liste des paquets disponibles
`apt-get upgrade`	met à jour le système
`apt-get install <paquet>`	installe le paquet logiciel
`unzip <fichier.zip>`	Décompresse l'archive
`ip addr`	donne des infos sur les interfaces réseaux