TP N°1
Systèmes de gestion de fichiers (Point de vue utilisateur) Manipulation du manuel

Le système de gestion de fichiers

Le système de fichiers racine (root file system), soit le système de fichiers primaire est associé au répertoire le plus haut **/** :
/bin commandes binaires utilisateur essentielles (pour tous les utilisateurs)
/boot fichiers statiques du chargeur de lancement
/dev fichiers de périphériques
/etc configuration système spécifique à la machine
/home répertoires personnels des utilisateurs (optionnel)
/lib bibliothèques partagées essentielles et modules du noyau
/mnt point de montage pour les systèmes de fichiers montés temporairement
/proc système de fichiers virtuel d'information du noyau et des processus
/root répertoire personnel de root (optionnel)
/sbin binaires système (binaires auparavant mis dans /etc)
/sys état des périphériques (model device) et sous-systèmes (subsystems)
/tmp fichiers temporaires

Partie I : Commandes de bases Linux

Le but de cette partie est la prise en main des commandes de base de l'environnement Linux.

1. Introduction

- Qu'est-ce que le **shell** ?

C'est l'interpréteur de commandes (l'interface) entre l'utilisateur et le système d'exploitation, d'où son nom anglais «shell», qui signifie «coquille».

Le shell est ainsi chargé de faire l'intermédiaire entre le système d'exploitation et l'utilisateur grâce aux lignes de commandes saisies par ce dernier. Son rôle consiste ainsi à lire la ligne de commande, interpréter sa signification, exécuter la commande, puis retourner le résultat sur les sorties.

Il existe plusieurs types de shells, les plus connus depuis Unix ayant une version améliorée sous Linux. Le fichier /etc/shells contient une liste de tous les shells disponibles :

/bin/**ash**
/bin/**bash**
/bin/**bash1**
/bin/**csh**
/bin/false
/bin/passwd
/bin/**sh**
/bin/**tcsh**
/usr/bin/**csh**
/usr/bin/**ksh**
/usr/bin/**tcsh**
/usr/bin/**zsh**

Les plus connus sont bash (version améliorée du shell Bourne sous Unix), ksh (version améliorée du shell Korn sous Unix) et tcsh (version améliorée du shell C sous Unix). La commande help affiche la liste des commandes internes du shell. Par défaut, c'est le shell Bash qui est installé avec Linux. C'est aussi le plus puissant et le plus utilisé, c'est pourquoi c'est celui-ci qui sera utilisé dans les sections suivantes.

Chaque utilisateur possède un shell par défaut, qui sera lancé à l'ouverture d'une invite de commande. Le shell par défaut est précisé dans le fichier de configuration **/etc/passwd** dans le dernier champ de la ligne correspondant à l'utilisateur. Il est possible de changer de shell dans une session en exécutant tout simplement le fichier exécutable correspondant, par exemple : **/bin/bash**

2. Commandes pour débuter

Avant de commencer, il faut savoir que Linux est **sensible à la casse** (*case sensitive* en anglais), c'est à dire qu'il distingue les majuscules des minuscules. Ainsi, pour créer un répertoire, la commande est 'mkdir', ce n'est pas la peine d'essayer MKDIR ou mKdiR, cela ne fonctionnera pas. De même, les noms de fichiers et de répertoires sont également sensibles à la casse.
De plus, sous Unix, les chemins sont séparés par des slash : /etc//init/xfs mais jamais etc\init\xfs.

Répertoires spéciaux :
. représente le répertoire courant,
.. représente le répertoire parent
~ représente le répertoire maison (home) de l'utilisateur

Fichiers cachés :
sous Unix, les fichiers cachés commencent par un point. Par exemple, ~/.bashrc est un fichier caché, dans le répertoire maison de l'utilisateur, qui contient la configuration de son shell.

Jokers : ? et *
Les caractères ? et * dans les noms de fichiers et de répertoires permettent de représenter des caractères quelconques. '?' représente un seul caractère, tandis que '*' en représente un nombre quelconque.

Par exemple "*.jpg" représente tous les fichiers se terminant par jpg ; "*toto*" tous les fichiers contenant "toto".

Il faut également savoir que c'est le shell qui interprète ces caractères avant de transmettre la ligne de commande. Par exemple, si vous tapez : rm f *.tmp, le shell transformera cette ligne de commande en : rm truc1.tmp truc2.tmp truc3.tmp.

Aliases

Plutôt que de taper de longues commandes, ou bien parce que vous préférez vous rappeler d'un nom plutôt que du vrai nom Unix, vous pouvez définir des aliases. Pour ce faire, utilisez la commande **alias** comme suit :

Si votre shell est bash ou sh ou ash (par défaut) :
alias md=mkdir
alias ls='ls --color'

Ainsi pourrez-vous taper md au lieu de mkdir, et; la commande ls affichera une sortie en couleurs...

Partie II : La commande man

Construire un tableau suivant le modèle donné, contenant les commandes suivantes (avec ou sans option).

man, man `-k`,
ls, ls `-a`, ls `-l`, ls `-al`, pwd, mkdir, rmdir, cp,
cp `-R`, mv, rm, rm `-R`, touch, echo, history,
history `-c`,
who, passwd,
cat.

Pour chacune de ces commandes, étudier la page d'aide et donner une description de la commande. Tester chacune de ces commandes sur des exemples concrets et analyser la réponse du système.

commande	Description Syntaxe Exemple
man	retourne le mode d'emploi de la commande s'il existe. man `[options] nom_commande` ex : man `man`
man -k	liste les commandes associées au mot clé donné. man `-k mot_cle` ex : man `-k copy`
ls	... ...

Si vous ne connaissez pas l'utilisation d'une commande, la recherche avec la commande man doit devenir un réflexe. Il est fortement conseillé de conserver ce tableau et de le compléter au fur et à mesure.