Université Mohammed Premier

Faculté des Sciences

Oujda

Systèmes d'Exploitation 1

Génie Informatique – S3

Informatique Appliquée – S3

Pr. AZDAD Nabila

Département d'Informatique

Email: n.azdad@ump.ac.ma

Année Universitaire: 2025 - 2026

03

Les commandes de base du système Unix

Introduction à l’interpréteur de commandes

- **Le shell** est un programme (un fichier exécutable) qui a la charge d’analyser et d’exécuter les commandes:
  – Il lit et interprète les commandes.
  – Il transmet ces commandes au système.
  – Il retourne le résultat.
- Le shell sert d’interface entre le noyau (le système d’exploitation) et l'utilisateur. Toutes les commandes sont envoyées au noyau à travers le shell.

Introduction à l’interpréteur de commandes

Principe du fonctionnement du shell en mode ligne de commande:

En mode ligne de commande :

- Le shell affiche dans un terminal ou dans une console virtuelle, une chaine de caractères appelée « prompt » (ou invite de commande) et attend la saisie d’une commande.
- Quand on tape une commande suivie de la touche « Entrée », le shell exécute cette commande et ensuite réaffiche le prompt et reste en attente d’une nouvelle commande.
- En générale la convention pour le prompt:
  - $ ou % pour l’utilisateur normal
  - # pour root (super-utilisateur ou administrateur) dans tous les shells

Les principaux shells

Il existe plusieurs shells sous Unix :

- **Le bourne shell; sh (/bin/sh):** ancêtre de tous les shells. Il est disponible sur toute plateforme UNIX.
- **Le Bourne again shell; bash (/bin/bash):** version améliorée de « sh ». Disponible dans le domaine public (Fourni le plus souvent avec Linux).
- **Le Korn shell; ksh (/bin/ksh):** Bourne Shell étendu par l'AT&T.
- **Le C shell (C veut dire California); csh (/bin/csh):** Il est développé par BSD. Il offre plus de facilités (rappel des commandes avec les flèches, gérer l'historique des commandes, etc).
- **Le Tenex C shell (TC Shell ou le C shell amélioré); tcsh (/bin/tcsh):** c'est une extension du C Shell (csh).

Les principaux shells

- Le fichier **« /etc/shells »** contient les shells que l'utilisateur peut utiliser. Pour les lister on utilise la commande:

$ cat /etc/shells

- Pour afficher le shell utilisé, on tape la commande:

$ echo $SHELL

Notion de commande: ligne de commande

- En général, une commande est un programme. Pour l’exécuter:

    1. On tape son nom, éventuellement suivi d’options et d’arguments.

    2. A la fin de la saisie, on tape la touche « Entrée », pour valider la saisie.

    3. Lorsqu’on appuie sur la touche « Entrée », le shell exécute la commande.

Notion de commande: ligne de commande

- Une commande Unix est une instruction donnée au système via un shell (interpréteur de commandes).

Syntaxe générale : commande [options] [arguments]

commande : le nom du programme ou utilitaire.

options : permettent de modifier le comportement de la commande (souvent précédées d'un **-** ou **--**).

arguments : fichiers, répertoires ou autres données sur lesquels la commande agit.

N.B: Les crochets désignent un élément facultatif

Exemple :

ls -l /home/user

- ls : liste les fichiers.
- -l : option d'affichage détaillé.
- /home/user : argument indiquant le dossier à lister.

Commandes liées à l’environnement

La commande « lsb_release -a»

Cette commande permet d’afficher les informations sur la distribution Linux utilisée:

La Commande « hostname »

La commande « hostname » : affiche le nom de la machine.

La Commande « logname »

La commande « logname » affiche le nom de login de l'utilisateur (le nom de connexion).

La commande « who »

Cette commande donne la liste des utilisateurs connectés sur la même machine.

Commande « passwd »:

Cette commande permet à l’utilisateur de changer son mot de passe.

Commandes liées à l’environnement

La commande « pwd » (Print Working Directory)

La commande « pwd » affiche le chemin d’accès absolu du répertoire courant.

La commande « which »: permet de localiser une commande, elle indique le chemin complet du fichier exécutable d’un programme (ex: **which** ls).

La commande « exit » : termine le shell (possible aussi CTRL-D, logout)

Commande « su »

La commande « su » (Switch User) permet d'ouvrir une session pour un autre utilisateur.

Syntaxe: su [-] utilisateur

Commandes liées à l’environnement

Commande « su » : Exemples

Cas 1: commande « su » sans tiret « - ».

Dans ce cas le nouveau utilisateur, après connexion, se trouvera dans le même répertoire de travail avant connexion.

$ pwd
/home/etudiant
$ su prof
passwd: <il faut taper ici le mot de passe de l’utilisateur « prof »>
$ pwd
/home/etudiant

Commandes liées à l’environnement

Commande « su » : Exemples

Cas 2: commande « su » avec tiret « - ».

Si le tiret « - » est précisé, le nouveau utilisateur se connecte avec son environnement de travail.

```
$ pwd
/home/etudiant
$ su - prof
passwd: <il faut taper ici le mot de passe de l’utilisateur « prof »>
$ pwd
/home/prof
```

Commandes liées à l’environnement

La commande echo: affiche un texte ou une variable (ex: **echo** "Bonjour Unix").

- L’option -e de la commande **echo** sert à interpréter les séquences d’échappement dans la chaîne de caractères.

Exemple:

- Dans **echo "Bonjour\nUnix "**, le \n est affiché tel quel.
  - L’affichage: Bonjour\nUnix
- Dans **echo -e "Bonjour\nUnix"**, le \n est interprété comme un retour à la ligne
  - L’affichage: Bonjour
    Unix

Commandes d’information et d’affichage

La commande « man »: affiche la documentation d’une commande (ex: **man** ls).

[**Astuce : q** pour quitter, **/mot** pour chercher un mot.]

La commande « date »: affiche la date et l’heure.

La commande « clear » : efface l’écran.

Commandes liées au système de fichiers

La commande « ls »

La commande « ls » liste le contenu d'un répertoire.

Syntaxe : ls [options] [argument]

- La commande « ls » sans arguments, liste le contenu du répertoire courant sauf les fichiers cachés (les fichiers commençant par un point).
- Si l'argument de la commande « ls » est un nom de fichier alors:
  - elle liste ce fichier (affiche le nom de ce fichier) s'il existe.
  - si le fichier spécifié en argument n'existe pas alors elle affiche un message d'erreur.
- Si l'argument utilisé est un nom de répertoire alors la commande « ls » liste le contenu du répertoire passé en argument.

Commandes liées au système de fichiers

Les options les plus utilisées de la commande « ls » (1/2)

- L’option « -l »: liste les fichiers avec des informations supplémentaires sur chaque fichier (le type de fichier, les protections, le nombre de liens , le propriétaire, le groupe, la taille en octets, la date de la dernière modification).
- L’option « -a »: affiche tous les fichiers y compris les fichiers cachés (fichiers dont le nom commence par « . » .
- L’option « -i » : affiche les numéros des inodes des fichiers.
- L’option « -h » : utilisé avec -l généralement « ls –lh ». Elle affiche la taille en unités lisibles : K (Ko), M (Mo), G (Go).

Commandes liées au système de fichiers

Les options les plus utilisées de la commande « ls » (2/2)

- L’option « -F »: ajoute un « / » après le nom de chaque répertoire, un «*» après chaque fichier possédant le droit d’exécution et un « @ » après chaque fichier lien.
- L’option « -R »: liste les fichiers et les répertoires de façon récursive.
- L’option « -d »: si le paramètre est un nom de répertoire, il vérifie s’il existe, si oui il affiche son nom et ne descend pas dans le répertoire (ne liste pas son contenu).

Commandes liées au système de fichiers

La commande touch

La commande « touch » est utilisée pour créer des fichiers vides ou mettre à jour les dates d’accès et de modification d’un fichier existant.

Syntaxe: touch [options] fichier1 fichier2 ...

- fichier1 fichier2 ... : noms des fichiers à créer ou modifier.
- Si le fichier n’existe pas, il sera créé vide.
- Si le fichier existe, sa date de modification et/ou d’accès sera mise à jour.

Commandes liées au système de fichiers

La commande touch - Les options principales

« **-t** AAAAMMJJhhmm.ss » → Définit une date précise pour l’accès et modification

« **-r** fichier_ref » → Prend les dates d’un fichier existant comme référence

Commandes liées au système de fichiers

La commande touch - Exemples

1. Créer un fichier vide si inexistant:

$ touch fichier.txt

2. Créer plusieurs fichiers:

$ touch fichier1.txt fichier2.txt fichier3.txt

3. Définir la date de modification et d'accès au 27 août 2025, 12h30:

$ touch -t 202508271230.00 fichier.txt

4. Copier les dates d'un fichier existant:

$ touch -r ref.txt nouveau.txt → nouveau.txt prendra les mêmes dates que ref.txt.

Commandes liées au système de fichiers

La commande cd (Change Directory)

La commande « cd » permet de changer le répertoire courant.

Syntaxe: cd [répertoire]

- La commande « cd » sans arguments, change le répertoire courant par le répertoire de connexion.
- La commande « cd » avec argument, remplace le répertoire courant par le répertoire spécifié. Le répertoire peut être spécifié avec un chemin absolu ou un chemin relatif.

Commandes liées au système de fichiers

La commande cd (Change Directory) - Exemple:

Supposons que « cours » et « TD » sont deux sous-répertoires du répertoire « INFO » qui est un sous-répertoire du répertoire de connexion «etudiant».

$pwd ➔ /home/etudiant	// répertoire courant est « etudiant »
$ cd /home/etudiant/INFO/TD	// chemin absolu
$ pwd ➔ /home/etudiant/INFO/TD	// répertoire courant est « TD »
$cd ../cours	// chemin relatif
$ pwd ➔ /home/etudiant/INFO/cours	// répertoire courant est « cours »
$cd ~prof	
$pwd ➔ /home/prof	// répertoire courant est « prof »

Commandes liées au système de fichiers

La commande « mkdir » (make directory)

La commande « mkdir » permet de créer des répertoires.

Syntaxe : mkdir [-p] [repertoire]

- Les arguments de « mkdir » sont les noms des répertoires à créer.
- Si le chemin n'est pas spécifié, le répertoire est créé dans le répertoire courant.
- Si le chemin est spécifié, la commande crée le répertoire dont le nom et le chemin sont spécifiés en argument de la commande. Le chemin peut être relatif ou absolu.

N.B: L'option « **-p** » signifie parents. Elle permet de créer toute une arborescence de répertoires d'un coup, même si les répertoires parents n'existent pas encore. Sans -p, si le répertoire parent n'existe pas → erreur.

Commandes liées au système de fichiers

La commande « mkdir » (make directory) - Exemble

Créer la hiérarchie de répertoires « INFO/OS/cours ». Pour cela :

- Soit on le fait en plusieurs étapes (on utilise 3 fois la commande «mkdir»):

```
$ mkdir INFO
$ mkdir INFO/OS
$ mkdir INFO/OS/cours
```

- Soit on le fait à l'aide de l'option « -p » en une seule commande de la manière suivante:

```
$ mkdir -p INFO/OS/cours
```

Remarque: avec l'option « -p », il n'y a pas de message d'erreurs si le répertoire existe. Dans ce cas la commande ne fait rien.

Commandes liées au système de fichiers

La commande « rmdir » (remove directory)

La commande « rmdir » permet de supprimer des répertoires.

Syntaxe: rmdir [-p] [repertoire]

- Les arguments de « rmdir » sont les noms des répertoires existants.
- Si le chemin n'est pas spécifié, le répertoire à supprimer est situé dans le répertoire courant.
- Si le chemin est spécifié, la commande supprime le répertoire dont le nom et le chemin sont spécifiés en argument de la commande. Le chemin peut être relatif ou absolu.

Commandes liées au système de fichiers

La commande « rmdir » (remove directory) - Exemple

$ rmdir tp supprime le répertoire « tp » situé dans le répertoire courant.

$ rmdir ../exam supprime le répertoire « exam» situé dans le répertoire père du répertoire courant.

$ rmdir /home/etudiant/INFO/tp supprime le répertoire « tp» situé dans le répertoire « INFO » qui est un sous-répertoire du répertoire de connexion «etudiant».

Attention: Les répertoires à supprimer doivent être vides (ils ne contiennent que « . » et « .. »).

L'option « **-p** » permet de supprimer une hiérarchie de sous-répertoires:

$ rmdir -p INFO/OS/cours supprime le répertoire **cours** dans « INFO/OS/ » s'il est vide. Ensuite, Si **OS** devient vide, OS est également supprimé. Puis, si **INFO** devient vide, INFO est aussi supprimé.

Commandes liées au système de fichiers

La commande « rmdir » (remove directory)

Attention: En cas de plusieurs répertoires à supprimer, la commande « rmdir » supprime les répertoires dans l'ordre dans lesquels ils ont été précisés sur la ligne de commande. Par conséquent, on doit faire attention à l'ordre des arguments.

Exemple:

$ rmdir INFO INFO/OS INFO/OS/cours Affiche un message d'erreur: « échec de suppression « SMI »: le dossier est non vide ».

Il faut respecter l'ordre en commençant par le répertoire le plus interne pour être sur que le répertoire est vide: $ rmdir INFO/OS/cours INFO/OS INFO

Commandes liées au système de fichiers

La commande « rm »

La commande « **rm** » (remove) permet de supprimer des fichiers ou des répertoires, à condition que les permissions le permettent.

Syntaxe : **rm** [options] [argument]

- Si l'argument est un nom de fichier alors il sera supprimé.
- Si l'argument est un répertoire alors il faut rajouter l'option « -r ».
    - « rm -r » supprime **un dossier et tout son contenu**, même s'il n'est pas vide.
- Si plusieurs arguments sont spécifiés, alors ils seront tous supprimés.

Commandes liées au système de fichiers

La commande « rm »

Les options:

• « -r » (récursif): efface le répertoire ainsi que son contenu (fichiers et sous-répertoires).

• « -i » (interactive): demande une confirmation (y ou n) sur chaque fichier à effacer.

• « -f » (force) : Ne demande pas de confirmation d'effacement, ne renvoie pas de code d'erreur lorsque le fichier n'existe pas.

Attention : il faut utiliser les options « -r » et « -f » avec précaution.

Commandes liées au système de fichiers

La commande cp (copy) :

La commande « cp » permet de copier un ou plusieurs fichiers vers un autre fichier ou vers un répertoire.

Syntaxe : cp [option] fich-source fich_destination

Options de la commande cp:

- **L'option « -i » :** l'utilisateur doit confirmer avant d'écraser un fichier existant.
- **L'option « -p » :** préservation des permissions, dates d'accès de modification.
- **L'option « -r » :** récursif. Si la source **est un répertoire**, alors on copie les fichiers et les sous-répertoires

Commandes liées au système de fichiers

La commande cp (copy) :

Premier cas: un seul fichier source

- Si « fich_destination » **est un répertoire**, alors le fichier «fich_source» sera dupliqué dans le répertoire «fich_destination».

Attention: si « fich_source » existe dans le répertoire destination alors il est écrasé et remplacé par le nouveau fichier « fich_source ».

- Si « fich_destination » **n'est pas un répertoir**e, Le contenu de fich_destination est écrasé par celui de fich_source.

Commandes liées au système de fichiers

La commande cp (copy) :

Deuxième cas: plusieurs fichiers sources

- Si la commande « cp » possède plusieurs fichiers sources alors la destination **doit être un répertoire**. Dans ce cas, la commande « cp », duplique les fichiers sources dans le répertoire spécifié.

Attention: Si l'un des fichiers existe dans le répertoire spécifié alors il sera écrasé et remplacé par le nouveau fichier (le fichier source).

Commandes liées au système de fichiers

La commande cp (copy) - Exemples

1. Le fichier destination est un répertoire: copie du fichier «image.jpg» dans le répertoire « INFO »

$ cp image.jpg INFO/

2. Un seul fichier source et le fichier destination n'est pas un répertoire: copie du fichier « cv.txt » dans le fichier «cv_back.txt»

$ cp cv.txt cv_back.txt

3. Plusieurs fichiers sources séparés par « espace », la destination est un répertoire: les fichiers « cv.txt » et « cv_back.txt » sont copiés dans le répertoire «INFO».

$ cp cv.txt cv_back.txt INFO/

Commandes liées au système de fichiers

La commande mv (move) :

Elle permet de renommer et/ou déplacer un fichier. Sa syntaxe est similaire à la commande « cp ».

Syntaxe : mv [option] fich-source fich_destination

N.B: Avec l'option « **-i** », l'utilisateur doit confirmer avant d'écraser un fichier existant.

Commandes liées au système de fichiers

La commande mv (move) :

Premier cas: un seul fichier source

- Si « fich_destination » est un répertoire alors la commande « mv » déplace le fichier « fich_source » dans le répertoire « fich_destination ».

Attention: si « fich_source » existe dans le répertoire de destination alors il est écrasé et remplacé par le nouveau fichier « fich_source ».

- Si « fich_destination » n'est pas un répertoire, alors la commande « mv » renomme le fichier source « fich_source » en lui donnant le nouveau nom « fich_destination ».

Attention: si « fich_destination » existe alors son contenu est écrasé et remplacé par celui de « fich_source ».

Commandes liées au système de fichiers

La commande mv (move) :

Deuxième cas: plusieurs fichiers sources:

- Si la commande « mv » possède plusieurs fichiers sources alors la destination **doit être un répertoire.**
- Dans ce cas, la commande « mv » déplace les fichiers sources vers le répertoire spécifié.

Attention: Si l'un des fichiers existe dans le répertoire spécifié alors son contenu sera écrasé et remplacé par le nouveau fichier (fichier source).

Commandes liées au système de fichiers

La commande mv (move) - Exemples

- Un seul fichier source.

$ mv cv.txt cv_old.txt

➔ Renomme le fichier source « cv.txt » en lui donnant le nouveau nom « cv_old.txt ».

- Plusieurs fichiers source, la destination doit être un répertoire.

$ mv image.jpg cv_old.txt cours/ documents/

➔ Déplacement des fichiers « image.jpg », « cv_old.txt » et du répertoire «cours» vers le répertoire « documents »