E-Concept Applications

L'informatique au service des professionnels

Introduction aux systèmes d'exploitation

SOMMAIRE

1) GÉNÉRALITÉS

1.1) Historique de Linux
1.2) Architecture globale de Linux
1.3) Les fichiers
1.4) Nommage des fichiers
1.5) Les jokers
1.6) Les utilisateurs
1.7) Droits des fichiers et répertoires

2) COMMANDES DE BASE DE LINUX

2.1) Syntaxe générale d'une commande
2.2) La documentation intégrée
2.3) Manipulation de fichiers
2.4) Gestion des droits
2.5) Personnaliser son terminal
2.6) Communication entre commandes et filtrage

3) ARCHITECTURE DE LINUX ET DU SYSTÈME DE FICHIERS

3.1) Noyau, démarrage, processus
3.2) Arborescence générale de Linux
3.3) Les démons
3.4) Visualiser les informations systèmes

SOMMAIRE

4) GESTION DES UTILISATEURS

4.1) Utilisateurs et groupes
4.2) Lister les utilisateurs et les groupes
4.3) Créer, modifier et supprimer des utilisateurs et des groupes

5) GESTION DES PAQUETS ET UTILISATION À DISTANCE

5.1) Le gestionnaire de paquets
5.2) Installer/mettre à jour des paquets
5.3) Utiliser linux à distance
5.4) Palier aux coupures de réseau en SSH

6) PROGRAMMATION SHELL

6.1) Introduction aux scripts shell
6.2) Les variables en shell
6.3) Calculs mathématiques
6.4) Paramètres et codes de retour
6.5) Tests et conditions
6.6) Les boucles
6.7) Les fonctions

1) Généralités

1.1) Historique de Linux

1.1) HISTORIQUE DE LINUX

- Né de la rencontre entre Unix et le monde des Logiciels Libres
- Noyau créé en 1991, diffusé sous licence libre GNU GPL à partir de 1992
- Son nom vient du créateur du noyau Linux : **Linus** Torvalds
- Répond aux spécifications de la norme POSIX
- Décliné rapidement sous forme de plusieurs distributions
- Exemples de distributions : Ubuntu, Debian, CentOS, Slackware, RedHat, etc.

1.2) Architecture globale de Linux

1.2) ARCHITECTURE GLOBALE DE LINUX

- Linux est avant tout un noyau, autour duquel va s'organiser l'ensemble du système d'exploitation (OS).
- Le noyau permet à l'OS de communiquer avec les différents composants matériels (processeur, mémoire, stockage, composants multimédias, etc.)
- Une distribution Linux est composée du noyau et d'un ensemble de paquets fournissant des outils et logiciels, formant un système d'exploitation complet et prêt à l'emploi.

1.2) ARCHITECTURE GLOBALE DE LINUX

Parmi les paquets fournis dans une distribution, on trouve généralement :

- Un terminal, permettant de manipuler l'OS en ligne de commande.
  Le terminal le plus répandu est « bash », également disponible sous Windows (en installant WSL) et Mac OS.
- Des pilotes pour la plupart des matériels existants
- Une interface graphique pour les systèmes à vocation bureautique
- Une sélection plus ou moins large et variée de logiciels selon la distribution.

1.3) Les fichiers

1.3) LES FICHIERS

- Dans un système Linux, absolument tout est représenté sous forme de fichiers.
- Il existe plusieurs types de fichiers :
  - Les fichiers réguliers : Fichiers « classiques », tels que documents, programmes, images, musique, etc.
  - Les répertoires : Equivalent des dossiers Windows. Fonctionnement identique, sauf pour la gestion des droits.
  - Les fichiers spéciaux : Liens, périphériques, canaux de communication & sockets.

1.3) LES FICHIERS

- L'ensemble du système est représenté sous forme d'une arborescence de fichiers.
- Il n'y a pas de lettre de lecteur comme Windows, même si plusieurs partitions peuvent être utilisées.
- Les partitions, comme les lecteurs externes (USB, optique, etc.) sont « montés » dans des répertoires de l'arborescence.

1.3) LES FICHIERS

- Les fichiers sont identifiés par un inode.
- L'inode contient entre autres les informations suivantes pour un fichier :
  - Le numéro d'inode
  - La taille du fichier
  - L'identifiant du périphérique contenant le fichier
  - L'identifiant du propriétaire du fichier, ainsi que celui du groupe
  - Le mode du fichier (ses droits d'accès)
  - L'horodatage du fichier (date de modification de l'inode, du fichier et de dernier accès)

1.3) LES FICHIERS

- Attention ! L'inode d'un fichier ne contient pas son nom !
- Un fichier peut en effet être présent à plusieurs endroits dans l'arborescence de fichiers, sous différents noms.
- Ce point sera abordé plus tard dans la partie sur les liens physiques et symboliques.

1.3) LES FICHIERS

- La racine du système est « / ».
- « / » sert également à séparer les répertoires dans le chemin de fichier.
- Par exemple, le fichier « `intro_système.txt` » situé dans le répertoire « `Documents` » de l'utilisateur « `mickael` » aura pour chemin d'accès : `/home/mickael/Documents/intro_système.txt`

1.4) Nommage des fichiers

1.4) NOMMAGE DES FICHIERS

- Un nom de fichier peut être composé de n'importe quel caractère alphanumérique et de la plupart des caractères spéciaux.
- Certains caractères spéciaux nécessitent d'être « échappés » quand on les utilise dans un nom de fichier en ligne de commande.
- Pour échapper un caractère, on le précède d'un « \ ». Par exemple « \$ ».
- Pour échapper le caractère « \ », on l'écrit simplement deux fois : « \\ ».

1.4) NOMMAGE DES FICHIERS

Les caractères interdits ou à éviter sont :

- « / » : Interdit, sert à séparer les noms de répertoire dans un chemin
- « \ » : Nécessite d'être « échappé », et peut poser des problèmes de compatibilité avec d'autres systèmes d'exploitation comme Windows. A éviter.
- « - » : Peut être utilisé, mais à éviter au **début** d'un nom de fichier.
  Le terminal interprète le « - » comme un paramètre pour une commande. Cela entrainerait donc une erreur de syntaxe avec un fichier nommé ainsi.

1.4) NOMMAGE DES FICHIERS

- « * », « ? », « : », « ' », « " », « # », « $ », « ; », « ! », « & » , « | », parenthèses « ( ) », accolades « { } », chevrons « <> », crochets « [ ] » : A éviter.
- Espace : Autorisé, mais nécessite d'être échappé.
- De façon générale, il est fortement recommandé de n'utiliser que les caractères alphanumériques (avec ou sans accents), le tiret « - », le souligné « _ », le point « . » et éventuellement l'espace (toutefois déconseillé).

1.4) NOMMAGE DES FICHIERS

Autres règles de nommage :

- Les noms de fichiers sont sensibles à la **casse**. Ainsi, « `bonjour.txt` » et « `Bonjour.txt` » sont deux fichiers **différents**.
- Un fichier commençant par un « . » est un fichier caché.
- Un nom de fichier est limité à 255 caractères. Il est toutefois conseillé de ne pas dépasser quelques dizaines de caractères au maximum, pour plus de lisibilité et de compatibilité avec d'autres systèmes d'exploitation.

1.4) NOMMAGE DES FICHIERS

- Le chemin d'accès d'un fichier peut être relatif ou absolu.
- Un chemin absolu se définit depuis la racine du système de fichiers « / » (cf exemple en partie 1.3, diapo 15.
- Un chemin relatif se détermine à partir du répertoire courant, en utilisant les noms spéciaux « . » et « . . »
- Si on écrit un nom de fichier avec un « / » à la fin, cela indique au système qu'on désigne explicitement un nom de répertoire. A noter que si le fichier existe, mais n'est pas un répertoire, le système renverra une erreur.

1.4) NOMMAGE DES FICHIERS

- « . » représente le répertoire courant
- « . . » représente le répertoire parent.
- Exemple : vous êtes dans le répertoire « `/home/mickael/Documents/cours_1` » et vous voulez faire référence au fichier « `iut.png` » situé dans le répertoire « `/home/mickael/Images` »
  Vous pouvez écrire : « `../../Images/iut.png` »

1.4) NOMMAGE DES FICHIERS

- Autre exemple : Vous souhaitez accéder à un fichier portant le même nom qu'une commande, par exemple, « `screen` ».
- Pour y accéder, vous devez indiquer explicitement que vous souhaitez accéder au **fichier** « `screen` » du répertoire courant (et non la commande homonyme) en écrivant « `./screen` ».
- Une autre solution est d'appeler le fichier « `screen` » en utilisant son chemin d'accès absolu.

1.5) Les jokers

1.5) LES JOKERS

- Les caractères « ? » et « * » sont appelés des jokers.
- « ? » permet de remplacer un et un seul caractère (n'importe lequel) dans un nom de fichier.

  Exemple :

  « `co?plet` » correspond aussi bien à « `couplet` » qu'à « `complet` » ou n'importe quel fichier dont le nom commence par « `co` », suivi d'un caractère, suivi de « `plet` ».

1.5) LES JOKERS

- « * » permet de remplacer un nombre quelconque de caractères, **y compris aucun caractère**.

  Exemple :

  « `bon*.txt` » peut correspondre entre autre aux noms de fichiers suivants : « `bonjour.txt` », « `bonsoir.txt` », « `bon_à_tirer.txt` », « `bon_je_mets_un_nom_de_fichier_pour_l_exemple.txt` », ou encore « `bon.txt` », mais pas « `bonjour` », « `bon` » ou encore « `jour.txt` ».

1.5) LES JOKERS

- Exception : Le caractère « . » ne peut pas être remplacé par un joker s'il est au début d'un nom de fichier.
- Conséquence directe : une chaine avec joker porte soit sur les fichiers non-cachés, soit sur les fichiers cachés, mais pas les deux en même temps.

1.6) Les utilisateurs

1.6) LES UTILISATEURS

- Toute commande est exécutée par un utilisateur
- L'utilisateur « maître » est l'utilisateur nommé « root »
- Compte « root » à utiliser avec prudence : Il peut faire « ce qu'il veut » !
- Utilisateurs réunis par groupes
- Un utilisateur n'accède qu'aux fichiers dont lui ou son groupe est propriétaire et/ou a les droits nécessaires. Exception : « root », encore une fois.

1.6) LES UTILISATEURS

- Un compte utilisateur ne sert pas forcément à se connecter à une machine.
- Il peut être utilisé uniquement pour lancer certaines commandes de façon automatisée.

1.7) Droits des fichiers et répertoires

1.7) DROITS DES FICHIERS ET RÉPERTOIRES

- Chaque fichier ou répertoire possède des droits sur trois niveaux :
  - Le propriétaire
  - Le groupe (pas forcément un de ceux du propriétaire)
  - Les autres
- Chacun de ces trois niveaux peut posséder trois droits :
  - r : Droit de lecture
  - w : Droit d'écriture
  - x : Droit d'exécution

1.7) DROITS DES FICHIERS ET RÉPERTOIRES

- Droits représentés sous forme d'une chaine de caractères :
  « rwx rwx rwx »
- Chaque groupe de 3 lettres correspond à un niveau (dans l'ordre cité précédemment)
- Si un droit n'est pas attribué pour un niveau, on met un tiret à la place.

1.7) DROITS DES FICHIERS ET RÉPERTOIRES

Exemple

- Pour un fichier donné, le propriétaire a tous les droits, les utilisateurs membres du groupe du fichier ont le droit de lecture et d'écriture, et les autres ne peuvent que lire le fichier. Les droits s'écrivent :

rwx rw- r--

1.7) DROITS DES FICHIERS ET RÉPERTOIRES

- Représentation binaire : 1 pour un droit attribué, 0 pour un droit refusé, sur 3 bits.
- L'exemple précédent (rwx rw- r--) s'écrit donc :

  111 110 100
- En décimal, on convertit chaque groupe de 3 bits en décimal :

  764

1.7) DROITS DES FICHIERS ET RÉPERTOIRES

- Droits affichés par une commande : généralement préfixés par une lettre :
- « - » : Fichier ordinaire
- « d » : Répertoire
- « l » : Lien symbolique
- Autres lettres : Fichiers spéciaux. De façon générale : **Ne pas toucher !**
- **Note : Le droit d'exécution est nécessaire pour pouvoir lister le contenu d'un répertoire.**

2) Commandes de base de Linux

2) COMMANDES DE BASE DE LINUX

Ce qui est demandé de retenir :

- Le nom de chaque commande
- A quoi elles servent, indépendamment de leurs paramètres
- Savoir à quoi servent les principaux paramètres présentés ici en les voyant.

Ce qui n'est pas demandé :

- Savoir lister par cœur les paramètres et leur rôle.

2) COMMANDES DE BASE DE LINUX

- **Entrainez-vous à utiliser les commandes vues au quotidien !**
- Le temps gagné en ligne de commande est **conséquent** dès lors qu'on a un tout petit peu l'habitude de l'utiliser, par rapport à une interface graphique.
- Astuce en ligne de commande : la touche **tab** complète énormément de choses : Le nom d'une commande, d'un répertoire, d'un fichier, etc.
  Un double appui sur tab affichera la liste des possibilités s'il y en a plusieurs.

2.1) Syntaxe générale d'une commande

2.1) SYNTAXE GÉNÉRALE D'UNE COMMANDE

- Une commande s'appelle directement par son nom
- On peut spécifier un ou plusieurs paramètres à une commande.
- Les paramètres peuvent être soit directement une valeur (par exemple, un nom de fichier), soit un nom de paramètre précédé d'un « – » et suivi de la valeur correspondante.
- Par exemple :

```
ls -h /home/mickael
```

2.2) La documentation intégrée

2.2) LA DOCUMENTATION INTÉGRÉE

- Les systèmes Linux disposent généralement d'une documentation intégrée.
- Celle-ci est accessible via les commandes « man » et « whatis » et permet d'obtenir la documentation détaillée de la commande spécifiée en paramètre.
- La documentation d'une commande indique sa syntaxe générale, ainsi que le détail des paramètres possibles.
- Dans la syntaxe générale, si un paramètre est indiqué entre [ ], alors ce paramètre est facultatif.

2.2) LA DOCUMENTATION INTÉGRÉE

- « `man` »

  Toutes les commandes de linux disposent d'un manuel intégré. Pour le consulter, il vous suffit d'utiliser la commande « `man` ».

  - Syntaxe :

    `man [x] commande`

  - Paramètres :
    - `x` : Facultatif – Précise le n° de la page du manuel
    - `commande` : Commande dont vous souhaitez consulter le manuel

2.2) LA DOCUMENTATION INTÉGRÉE

- « `whatis` »

  « `whatis` » permet d'avoir un descriptif des différentes pages de manuels pour la commande donnée. Cela permet d'identifier la page pour « `man` » contenant l'information recherchée.

  - Syntaxe :

    `whatis` *`commande`*

  - Paramètres :

    - `commande` : nom de la commande sur laquelle porte la demande

2.2) LA DOCUMENTATION INTÉGRÉE

- Que fait la commande suivante ?

```
man man
```

- Elle affiche la documentation intégrée de la commande « `man` » !

Plus de détails sur la commande « man » à l'adresse suivante :
https://www.malekal.com/commande-man-linux/

2.3) Manipulation de fichiers

2.3) MANIPULATION DE FICHIERS

- « `pwd` »

  Indique le répertoire courant

  - Syntaxe :

    ```
    pwd
    ```

2.3) MANIPULATION DE FICHIERS

- « `cd` »

  Permet de changer de répertoire.

  - Syntaxe :

    `cd` *nom*

  - Rappel : Comme sous Windows et Mac, nous retrouvons deux répertoires spéciaux dans tous les répertoires :
    - « `.` » : Ce répertoire désigne le répertoire courant
    - « `..` » : Ce répertoire désigne le répertoire parent

2.3) MANIPULATION DE FICHIERS

- « `ls` »

  Permet de lister le contenu d'un répertoire.

  - Syntaxe :

    `ls` *`[options] [fichier]`*

  - options (liste partielle) :
    - -C : Affiche la liste en colonnes , triée verticalement
    - -R : Affiche récursivement le contenu des sous-répertoires

2.3) MANIPULATION DE FICHIERS

- -a : Affiche tous les fichiers, y compris ceux commençant par un « . » (Fichiers cachés)
- -l : Affiche également le type de fichier, les droits d'accès, le nom du propriétaire et du groupe, la taille en octet et l'horodatage (entre autres).
- -r : Inverse l'ordre de tri
- -t : Tri par date et non par ordre alphabétique (t comme « time »)
- -h : « Human readable » : Ajout une lettre pour l'unité de taille, comme k pour kilo-octet, plus lisible qu'en octets.

- fichier : Le(s) nom(s) de fichiers/répertoires à lister. Si plusieurs noms donnés, listera chacun des éléments donnés en paramètre. Si non spécifié, utilise le répertoire courant.

2.3) MANIPULATION DE FICHIERS

- « `mkdir` »

  Crée un répertoire

  - Syntaxe :

    `mkdir [options] repertoire`

  - options (liste partielle) :

    - -p : Crée les répertoires parents si nécessaire

      Exemple : mkdir -p a/b/c => Crée a et b en plus de c si nécessaire.

  - repertoire : Nom du répertoire à créer

2.3) MANIPULATION DE FICHIERS

- « `cp` »

  Gère la copie de fichiers

  - Syntaxe :

    `cp [options] source destination`

  - options (liste partielle) :

    - -f : Efface automatiquement les fichiers dans la destination si ceux-ci existent déjà. Attention ! Ne demande aucune confirmation ! A utiliser prudemment, surtout en combinaison avec -r.
    - -r : Copie récursivement tout le contenu des répertoires

2.3) MANIPULATION DE FICHIERS

  - -H : Si un lien symbolique est spécifié dans les fichiers à copier, suit le lien plutôt que de copier le lien lui-même. Les liens rencontrés par une copie récursive (donc, non spécifiés directement dans les fichiers à copier) ne seront pas suivis.
- source : Le(s) nom(s) de fichiers/répertoires à copier (séparés par des espaces si plusieurs)
- destination : Nom du répertoire de destination ou nom de fichier à utiliser pour la copie (Permet ainsi de renommer un fichier en même temps que de le copier. Un seul fichier source possible, dans ce cas)

2.3) MANIPULATION DE FICHIERS

- IMPORTANT : **Les copies appartiennent à l'utilisateur qui a effectué la copie !** Si vous faites la copie en tant que « root » pour un autre utilisateur, pensez à rétablir le bon propriétaire et le bon groupe après la copie sur les fichiers copiés !

- « `mv` »

Gère le déplacement de fichiers

- Même syntaxe que « `cp` », à l'exception du paramètre -r qui n'a aucun sens avec cette commande.
- Note : Contrairement à « `cp` », « `mv` » **conserve** le propriétaire et le groupe des fichiers.

2.3) MANIPULATION DE FICHIERS

- « `rm` »

  Gère la suppression de fichiers

  - Syntaxe :

    `rm [options] fichier`

  - options (liste partielle) :

    - -f : Efface automatiquement les fichiers sans demander de confirmation.
      A utiliser très prudemment, particulièrement en combinaison avec -r.
      Par exemple : « `rm -rf /` » supprimera purement et simplement tous les fichiers du système

2.3) MANIPULATION DE FICHIERS

- -r : Efface récursivement tous les fichiers, y compris les fichiers des sous-répertoires et les sous-répertoires eux-mêmes
- fichier : Le(s) nom(s) de fichiers/répertoires à effacer

2.3) MANIPULATION DE FICHIERS

- « `rmdir` »

  Gère la suppression de répertoires

  - Syntaxe :

    `rmdir` *`[options] répertoire`*

  - options (liste partielle) :

    - -p : Efface également le répertoire parent si celui-ci devient vide après suppression du répertoire spécifié. Par exemple, `rmdir -p a/b/c` est équivalent à `rmdir a/b/c; rmdir a/b; rmdir a`

2.3) MANIPULATION DE FICHIERS

- repertoire : Le(s) nom(s) de répertoire(s) à supprimer.
- IMPORTANT :

Un répertoire doit être vide pour pouvoir être supprimé avec rmdir !!!