THE INNOVATIVE UNIVERSITY

Système Linux et Commandes de Bases

Ilias TOUGUI

L'Ecole Supérieure d'Informatique et du Numérique

INF2132 - Systèmes d'Exploitation

PLAN DU COURS

1. Introduction à la sécurité Linux
2. Système Unix
3. Le Terminal et le Shell
4. Commandes Linux de base

Les couches d'un système Unix

Les systèmes Unix respectent ce qu'on appelle le modèle à trois couches :

1. La couche matérielle, la plus basse, représente l'électronique de l'ordinateur (cartes, processeurs, etc.)
2. La couche noyau (kernel en anglais), intermédiaire, exploite le matériel et fournis des services standardisés à la couche application
3. La couche applications, la plus haute, représente les programmes s'exécutants sur l'ordinateur

Notions de base: Système GNU et Noyau Linux

GNU : Le système d'exploitation

- GNU (GNU is Not Unix) est un projet de système d'exploitation libre lancé par Richard Stallman en 1984
- GNU comprend tous les outils et utilitaires nécessaires au fonctionnement d'un système complet : compilateurs (GCC), éditeurs (Emacs), interpréteurs de commandes (Bash), bibliothèques système (glibc)
- Le projet GNU existait donc huit ans avant la création du noyau Linux

Notions de base: Système GNU et Noyau Linux

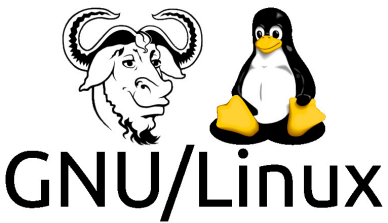
Linux : Le noyau

- Linux, créé par Linus Torvalds en 1991, est uniquement le noyau du système d'exploitation
- Le noyau est le programme central qui gère les ressources matérielles (processeur, mémoire, périphériques) et permet aux autres programmes de fonctionner
- Seul, Linux ne constitue pas un système d'exploitation utilisable

Notions de base: Système GNU et Noyau Linux

GNU/Linux : Le système complet

- Le système que la plupart des gens appellent simplement “Linux” est en réalité GNU/Linux
- Il s’agit du système GNU auquel on a ajouté le noyau Linux pour former un système d’exploitation complet et fonctionnel
- Les premières distributions ont rapidement intégré les outils GNU avec le noyau Linux, créant ainsi des systèmes d’exploitation libres et complets

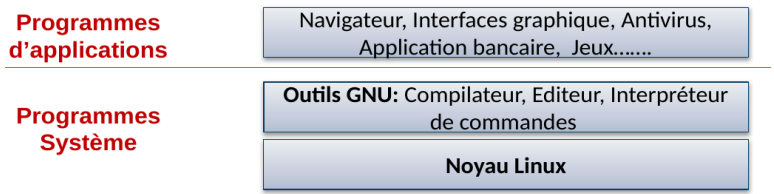


Notions de base: Distribution

Si on vous livrait le noyau Linux seul, accompagné des outils GNU de base, vous auriez peu de fonctionnalités: pas d'interface graphique et juste quelques commandes.

C'est pour cela qu'existe des distributions Linux qui contiennent:

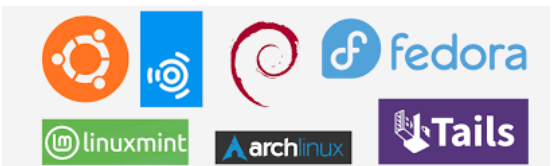
- Le noyau Linux
- Les outils GNU
- Un ensemble de logiciels qu'elles ont choisi de supporter



Notions de base: Distribution

Il existe un grand nombre de distributions Linux différentes. Les principales sont :

- **Slackware :** une des plus anciennes distributions de Linux qui existe toujours
- **Mandriva :** éditée par une entreprise française
- **SuSE :** éditée par l'entreprise Novell
- **Red Hat:** éditée par une entreprise américaine, très répandue sur les serveurs
- **Debian :** gérée par des développeurs indépendants, une des plus populaires
- **Ubuntu :** basée sur Debian, très populaire pour les débutants
- **Kali Linux :** distribution spécialisée en sécurité informatique



Pourquoi Linux ?
Chez les développeurs

- Linux dépasse désormais macOS avec 40,23% d'utilisation professionnelle contre 32,97% pour macOS, selon Stack Overflow 2024.
- Cette tendance marque une progression constante depuis 2018 où Linux représentait seulement 23,2%

Pourquoi Linux ?

Domination serveurs et cloud

Linux domine massivement l'infrastructure critique :

- 49,2% de toutes les charges de travail cloud globalement
- 100% des 500 superordinateurs les plus rapides au monde depuis 2017
- 55,8% du top million de serveurs web selon W3Techs

Pourquoi Linux ?

Sécurité renforcée

Linux offre une architecture sécurisée par conception :

- Système de permissions granulaires
- Corrections rapides des vulnérabilités par la communauté
- Résistance naturelle aux malwares
- Authentification multi-facteurs intégrée

Pourquoi Linux ?

Performance et stabilité

Les systèmes Linux démontrent une fiabilité exceptionnelle :

- Fonctionnement continu sans redémarrages fréquents
- Temps de disponibilité élevés pour applications critiques
- Optimisation hardware personnalisable
- Gestion efficace des ressources système

Pourquoi Linux ?

Économies substantielles

Linux représente une solution économique :

- Système d'exploitation gratuit sans frais de licence
- Réduction du coût total de possession
- Support communautaire étendu et gratuit
- Moins de ressources hardware requises

Pourquoi Linux ?
Flexibilité technique

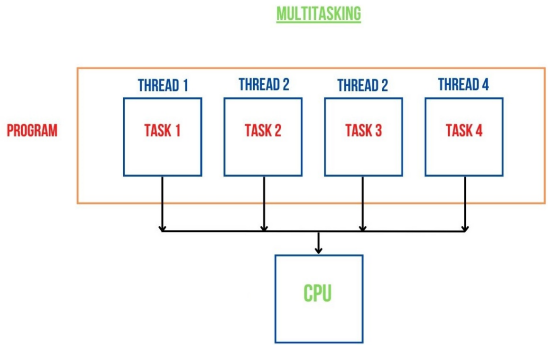
L'adaptabilité de Linux séduit les professionnels :

- Code source ouvert modifiable
- Multiples environnements desktop (GNOME, KDE, XFCE)
- Personnalisation complète du système
- Large écosystème d'outils de développement

Caractéristiques d'un système Linux

Multi-utilisateurs et Multitâches : cela signifie que plusieurs utilisateurs peuvent accéder simultanément au système et exécuter un ou plusieurs programmes.

Temps partagé : c'est-à-dire que les ressources du processeur et du système sont réparties entre les utilisateurs.

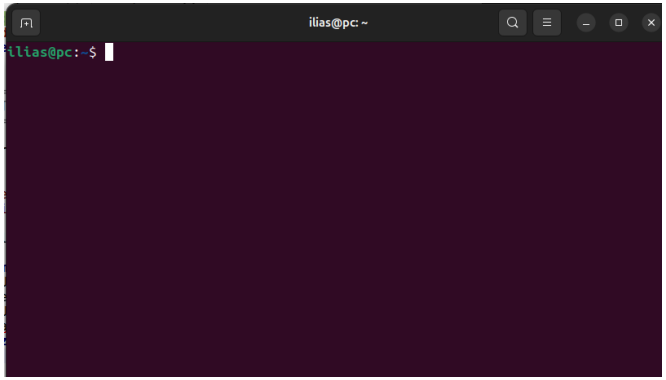


Le Terminal

Porte d'entrée d'un ordinateur

Un terminal offre :

- Un canal pour entrer des données (clavier, souris, écran tactile...)
- Un canal pour afficher des données (écran, imprimante, haut-parleur...)



Notion de shell

Le « Shell » est ce que l'on appelle un « Interpréteur de commandes ».

Il date de l'époque d'UNIX, où le seul moyen de communiquer avec sa machine était d'écrire des lignes textes au clavier, dans un langage compréhensible à la fois par l'humain et la machine.

Le rôle de la machine étant d'exécuter les commandes de l'utilisateur et d'afficher le résultat à l'écran.

Notion de shell

Types de Shell

Le shell a évolué à travers les années, plusieurs types de Shell existent :

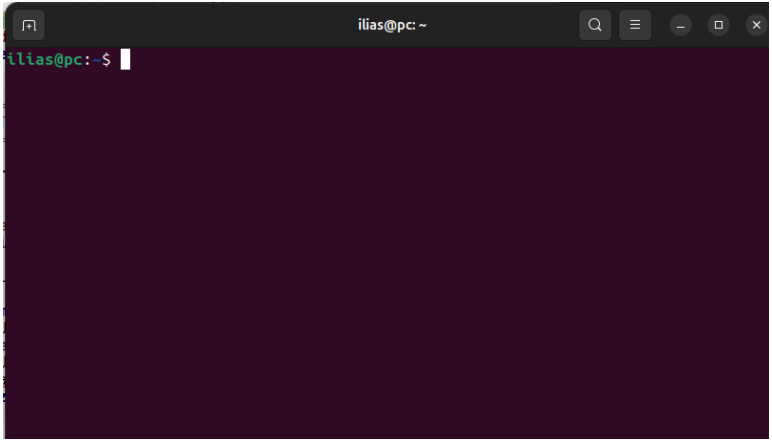
- `/bin/sh`: Bourne shell
- `/bin/bash`: Bourne Again SHell
- `/bin/csh`: C shell
- `/bin/ksh`: Korn shell
- `/bin/tcsh`: C shell amélioré

Le principe de base est toujours resté le même : Les shells sont des interpréteurs, ils lisent chaque commande saisie par l'utilisateur, vérifient et traitent la syntaxe pour l'exécuter.

Notion de shell

Accéder au shell sur Ubuntu

Pour accéder au shell sur Ubuntu, cliquer sur Activités, puis rechercher et ouvrir le terminal:



Notion de shell

Le Prompt

Le shell par défaut (bash) va s'exécuter dans la fenêtre du terminal:

```
ilias@pc:~$
```

Le prompt affiche:

- Le nom d'utilisateur
- Le nom de la machine
- Le répertoire courant (~ représente le répertoire personnel)
- Le symbole $ (ou # pour root)

Avantages du shell

Avantages

- Installé d'office sous tous système GNU/Linux
- Manipule essentiellement des chaînes de caractères
- Adapté au prototypage rapide d'applications
- C'est un langage « glue » : exécuter et agglomérer des composants divers

Difficultés

- Messages d'erreurs parfois difficiles à exploiter
- Temps d'apprentissage : la syntaxe est cohérente mais ardue

Notion de Commande Linux

Une commande est l'exécution d'un programme dans le Shell. Elle prend en entrée des options et/ou des paramètres.

La syntaxe générique d'une commande shell est :

```
commande [options] [argument1] [argument2] [...]
```

Exemple: `ls -l /tmp`

Les options peuvent être :

- Options courtes : `-a`
- Options longues : `-help`
- Options avec argument : `-speed 50` ou `-t 50`

Man utility

Man est une commande disponible sur les systèmes Unix. Elle permet de visionner le manuel d'une commande.

Elle doit être utilisée sous la forme suivante :

```
man [nom_de_commande]
```

Attention

Il ne faut pas garder les [], il faut les remplacer par ce qui est indiqué à l'intérieur!

Exemple: `man ls`

Info & Help utility

Info

Info est un logiciel utilitaire constituant une documentation multipage et hypertextuelle.

```
info [nom_de_commande]
```

Help

Help permet d'avoir des informations plus précises et plus courtes sur les commandes.

```
[nom_de_commande] --help
```

Notion de chemin ou PathName

Quand l'utilisateur se connecte au système Unix, il est positionné automatiquement dans son répertoire personnel (répertoire home).

À un instant donné, il est toujours dans un répertoire appelé le **répertoire courant**.

L'accès à un fichier s'effectue en désignant le nom du chemin (pathname). Ce chemin peut être :

- **Absolu :** partant de la racine (/)
- **Relatif :** à partir du répertoire courant

Notion de chemin ou PathName

Supposons que le répertoire courant est `/home/Groupe1`
Pour accéder à `/home/Groupe2/User2`:

- Chemin absolu : `/home/Groupe2/User2`
- Chemin relatif : `../Groupe2/User2`

Notion de chemin ou PathName

Un chemin vers un fichier s'écrit en séparant les différents nœuds par le caractère `/`

Il y a deux répertoires spéciaux dans chaque répertoire :

- `.` : Pour désigner le répertoire lui-même
- `..` : Pour désigner le répertoire père

Par exemple, si on est dans `/home`:

- `.` désignera le répertoire `/home` lui-même
- `..` désignera le répertoire racine `/`

Commande ls

Syntaxe :

```
ls [options] [fichiers]
```

Description : ls liste les répertoires et les fichiers. Par défaut, la sortie est envoyée à l'écran par ordre alphabétique.

Options courantes:

- `-R` : Traitement récursif
- `-a` : Tous les fichiers (y compris ceux qui commencent par un point)
- `-d` : Affiche le nom des répertoires sans leur contenu
- `-l` : Format long (avec beaucoup de détails)

Exemple: `ls -l /home`

Commande cd

Syntaxe :

```
cd [répertoire]
```

Description : La commande cd permet de changer le répertoire de travail. Si répertoire n'est pas précisé, alors le nouveau répertoire de travail sera le répertoire personnel.

Exemples:

- `cd /dev` : change vers /dev
- `cd ..` : remonte d'un niveau
- `cd` ou `cd ~` : retour au répertoire personnel

Commande pwd

Syntaxe :

```
pwd
```

Description : La commande pwd permet d'afficher le répertoire courant.

Option : La commande pwd n'accepte pas d'option

Exemple:

```
$ pwd
/home/utilisateur/Documents
```

Commande mkdir

Syntaxe :

```
mkdir [options] [nouveau_répertoire]
```

Description : La commande mkdir créé le répertoire spécifié. Si l'un des répertoires intermédiaires n'existe pas, la commande retourne une erreur (sauf si l'option -p est spécifiée).

Options courantes:

- `-p` : permet de créer tous les répertoires intermédiaires qui n'existeraient pas

Exemple: `mkdir -p projet/src/main`

Commande rmdir

Syntaxe :

```
rmdir [options] [répertoire]
```

Description : La commande rmdir supprime le répertoire spécifié. Si il existe des fichiers ou des sous répertoires, la commande retournera une erreur.

Options courantes:

- `-p` : permet de détruire tous les sous-répertoires vides
- `-s` : mode silencieux (aucun affichage)

Exemple: `rmdir ancien_dossier`

Commande cp

Syntaxe :

```
cp [options] [fichier1] [fichier2]
cp [options] [fichier1] [fichier2] ... [répertoire]
```

Description : La commande cp copie le contenu de fichier1 dans fichier2, ou bien elle copie fichier1, fichier2, etc... dans répertoire

Options courantes:

- `-i` : mode interactif, demande la confirmation avant écrasement
- `-p` : conserve les dates du fichier source
- `-r` : copie récursive de répertoires

Exemple: `cp -r projet/ sauvegarde/`

Commande mv

Syntaxe :

```
mv [options] [fichier1] [fichier2]
mv [options] [fichier1] [fichier2] ... [répertoire]
```

Description : La commande mv renomme fichier1 en fichier2, ou bien elle déplace fichier1, fichier2, etc... dans répertoire.

Options courantes:

- `-f` : écrase les fichiers de destination sans demander de confirmation
- `-i` : mode interactif, demande confirmation avant écrasement

Exemples:

- `mv ancien.txt nouveau.txt` : renomme
- `mv fichier.txt /tmp/` : déplace

Commande rm

Syntaxe :

```
rm [options] [fichier...]
```

Description : La commande rm supprime définitivement les fichiers spécifiés.

Options courantes:

- `-r` : suppression récursive (répertoires et contenu)
- `-f` : force la suppression sans confirmation
- `-i` : mode interactif, demande confirmation

Attention

La commande rm est irréversible! Soyez prudent avec `rm -rf`

Exemple: `rm -i fichier.txt`

Commande ln

Syntaxe :

```
ln [options] [fichier] [fichier_lien]
```

Description : La commande ln permet de créer des entrées multiples dans l'arborescence pour un même fichier physique. Si l'on modifie un fichier, ses liens le sont aussi.

ln permet aussi de faire des liens symboliques (comme les raccourcis dans Windows).

Option :

- `-s` : permet de faire un lien symbolique

Exemple: `ln -s /chemin/long/fichier.txt raccourci.txt`

Commande cat

Syntaxe :

```
cat [fichier...]
```

Description : La commande cat visualise et/ou concatène les fichiers spécifiés sur la ligne de commande.

Par défaut, cat lit sur l'entrée standard et affiche le résultat sur la sortie standard.

Exemples:

- `cat fichier.txt` : affiche le contenu
- `cat fichier1.txt fichier2.txt` : concatène et affiche
- `cat fichier1.txt fichier2.txt > resultat.txt` : concatène dans un nouveau fichier

Commande less

Syntaxe :

```
less [fichier...]
```

Description : La commande less affiche page par page le contenu d'un fichier.

Navigation dans less:

- Espace : page suivante
- b : page précédente
- / : rechercher
- q : quitter

Exemple: `less /var/log/syslog`

Commande head & tail

Syntaxe :

```
head [-n] [fichier...]
tail [-n|+n] [fichier]
```

Description : head affiche les n premières lignes d'un fichier, tail affiche les dernières lignes. Si n n'est pas précisé, il prend la valeur 10.

Options courantes:

- `-n` : nombre de lignes à afficher
- `+n` : affichage à partir de la ligne numéro n (tail seulement)
- `-f` : suit les modifications en temps réel (tail seulement)

Exemples:

- `head -5 fichier.txt`
- `tail -f /var/log/syslog`

Commande sort

Syntaxe :

```
sort [options] [fichier...]
```

Description : La commande sort trie les lignes des fichiers et affiche le résultat. Par défaut, tri par ordre alphabétique.

Options courantes:

- `-u` : n'affiche qu'une seule fois les lignes multiples
- `-f` : ne différencie pas minuscules et MAJUSCULES
- `-n` : effectue un tri numérique
- `-r` : ordre décroissant
- `-o fic` : enregistre la sortie dans fic

Exemple: `sort -n notes.txt`

Commande wc

Syntaxe :

```
wc [options] [fichier...]
```

Description : La commande wc compte le nombre de lignes, mots, ou caractères d'un fichier texte. Si aucune option n'est précisée, wc compte le nombre de lignes, mots, et caractères.

Options courantes:

- `-l` : compte le nombre de lignes
- `-w` : compte le nombre de mots
- `-c` : compte le nombre de caractères

Exemple: `wc -l fichier.txt`

Commande touch

Syntaxe :

```
touch [fichier]
```

Description : La commande touch permet de créer un fichier vide appelé fichier. Si le fichier existe, elle va changer sa dernière date de modification à la date actuelle du système.

Exemples:

- `touch nouveau.txt` : crée un fichier vide
- `touch fichier_existant.txt` : met à jour la date
- `touch fichier1.txt fichier2.txt fichier3.txt` : crée plusieurs fichiers

Commande nano

Syntaxe :

```
nano [fichier]
```

Description : nano est un éditeur de texte en ligne de commande. Elle permet d'éditer simplement le contenu d'un fichier.

Raccourcis clavier principaux:

- `Ctrl + O`: sauvegarder le fichier
- `Ctrl + W`: rechercher dans le fichier
- `Ctrl + K`: couper la ligne
- `Ctrl + U`: coller
- `Ctrl + X`: fermer le fichier

Commande find

Syntaxe :

```
find [chemin] [options] [expression]
```

Description : La commande find recherche des fichiers et répertoires dans l'arborescence selon des critères spécifiques.

Options courantes:

- `-name` : recherche par nom
- `-type f` : fichiers uniquement
- `-type d` : répertoires uniquement
- `-size` : recherche par taille
- `-mtime` : recherche par date de modification

Exemples:

- `find . -name "*.txt"` : trouve tous les fichiers .txt
- `find /home -type d -name "projet"` : trouve les dossiers nommés "projet"

Commande du

Syntaxe :

```
du [options] [fichier/répertoire]
```

Description : La commande du (disk usage) affiche l'espace disque utilisé par les fichiers et répertoires.

Options courantes:

- `-h` : affichage lisible (human-readable)
- `-s` : affiche uniquement le total
- `-a` : affiche tous les fichiers, pas seulement les répertoires
- `-max-depth=N` : limite la profondeur de recherche

Exemples:

- `du -h` : taille de tous les sous-répertoires
- `du -sh *` : taille de chaque élément du répertoire courant

Commande df

Syntaxe :

```
df [options] [système_de_fichiers]
```

Description : La commande df (disk free) affiche l'espace disque disponible sur les systèmes de fichiers montés.

Options courantes:

- `-h` : affichage lisible (human-readable)
- `-T` : affiche le type de système de fichiers
- `-i` : affiche les informations sur les inodes

Exemples:

- `df -h` : espace disque de tous les systèmes montés
- `df -h /home` : espace disque de /home

Commande file

Syntaxe :

```
file [fichier...]
```

Description : La commande file détermine le type d'un fichier en examinant son contenu.

Exemples:

```
$ file document.pdf
document.pdf: PDF document, version 1.4

$ file image.jpg
image.jpg: JPEG image data

$ file script.sh
script.sh: Bourne-Again shell script
```

Commande clear

Syntaxe :

```
clear
```

Description : La commande clear efface tout le contenu affiché dans le terminal et replace le curseur en haut de l'écran.

Raccourci équivalent: `Ctrl + L`

Cette commande est utile pour nettoyer l'écran lorsque le terminal devient encombré.

Commande history

Syntaxe :

```
history [options]
```

Description : La commande history affiche la liste des commandes précédemment exécutées dans le shell.

Utilisation:

- `history` : affiche toutes les commandes
- `history 10` : affiche les 10 dernières commandes
- `!n` : exécute la commande numéro n
- `!!` : exécute la dernière commande
- `!cat` : exécute la dernière commande commençant par "cat"

Raccourci: `Ctrl + R` pour rechercher dans l'historique

Commande echo

Syntaxe :

```
echo [options] [texte]
```

Description : La commande echo affiche du texte ou des variables à l'écran.

Exemples:

```
$ echo "Bonjour le monde"
Bonjour le monde

$ echo "Texte" > fichier.txt     # Écrit dans un fichier
$ echo "Ajout" >> fichier.txt    # Ajoute à la fin

$ echo $HOME                      # Affiche une variable
/home/utilisateur
```

Commande date

Syntaxe :

```
date [options] [+format]
```

Description : La commande date affiche ou modifie la date et l'heure du système.

Exemples:

```
$ date
Sat Oct 25 13:30:00 CET 2025

$ date +%Y-%m-%d
2025-10-25

$ date +"%d/%m/%Y %H:%M"
25/10/2025 13:30
```


Formats utiles: `%Y` (année), `%m` (mois), `%d` (jour), `%H` (heure), `%M` (minute)

Commande which

Syntaxe :

```
which [commande]
```

Description : La commande which localise l'emplacement d'une commande exécutable dans le PATH.

Exemples:

```
$ which ls
/usr/bin/ls

$ which python3
/usr/bin/python3

$ which nano
/usr/bin/nano
```

Résumé des commandes essentielles

Commande	Description
`ls`	Liste les fichiers et répertoires
`cd`	Change de répertoire
`pwd`	Affiche le répertoire courant
`mkdir`	Crée un répertoire
`rmdir`	Supprime un répertoire vide
`cp`	Copie des fichiers
`mv`	Déplace ou renomme des fichiers
`rm`	Supprime des fichiers
`cat`	Affiche le contenu d'un fichier
`touch`	Crée un fichier vide
`find`	Recherche des fichiers
`du/df`	Espace disque utilisé/disponible