

# Administration des systèmes



*Année académique :2021-2022*

# Présentation du formateur



M. OUEDRAOGO W. A. Marc Christian

Ingénieur des travaux en réseaux et maintenance  
informatique

Ingénieur de conception en analyse et programmation

Doctorant en Intelligence Artificielle

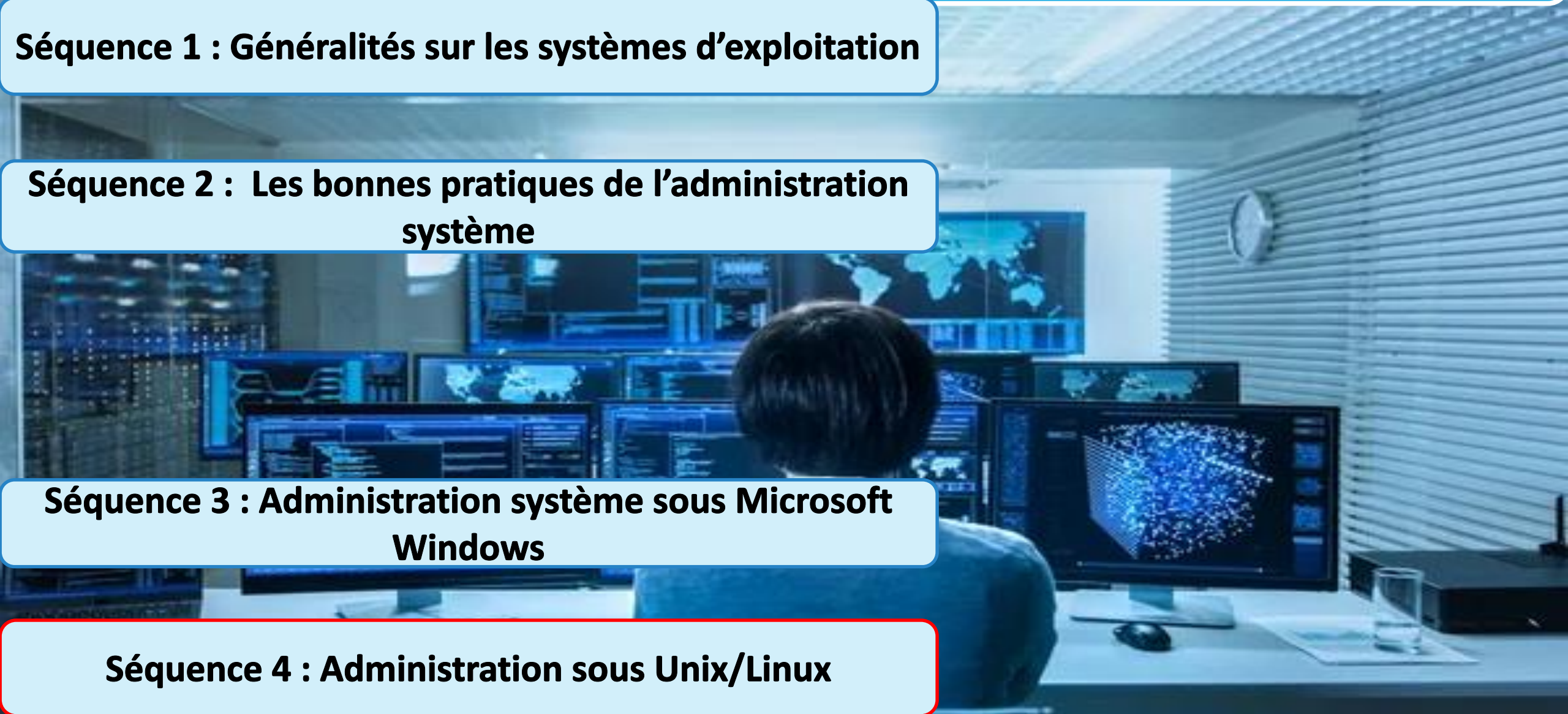
# PLAN GENERAL DU COURS

**Séquence 1 : Généralités sur les systèmes d'exploitation**

**Séquence 2 : Les bonnes pratiques de l'administration système**

**Séquence 3 : Administration système sous Microsoft Windows**

**Séquence 4 : Administration sous Unix/Linux**





# Linux



# UNIX

Séquence 4 : Administration système sous Unix/Linux

## OG4 : Appliquer l'administration sous Unix/Linux

- S'informer sur les systèmes d'exploitation Unix/Linux
- Utiliser le terminal (shell) Unix
- Utiliser les script shell (bash)

# PLAN

**Chapitre IX : Présentation de Unix/Linux**

**Chapitre X : Le terminal Linux**

**Chapitre XI : Introduction au scripting Linux**



# Chapitre X : Le terminal Linux

**I. Introduction à l'interpréteur de commandes : le shell**

**II. Commandes Unix en mode console**

**III. Les commandes liées à l'environnement**

**IV. Les commandes liées à la gestion des utilisateurs**

**V. Les permissions de fichiers sur Linux**

**VI. Gestionnaire de package Linux**

**VII. Commandes liées au file system**

**VIII. Redirection des entrées-sorties standards et les tubes (Les pipes )**

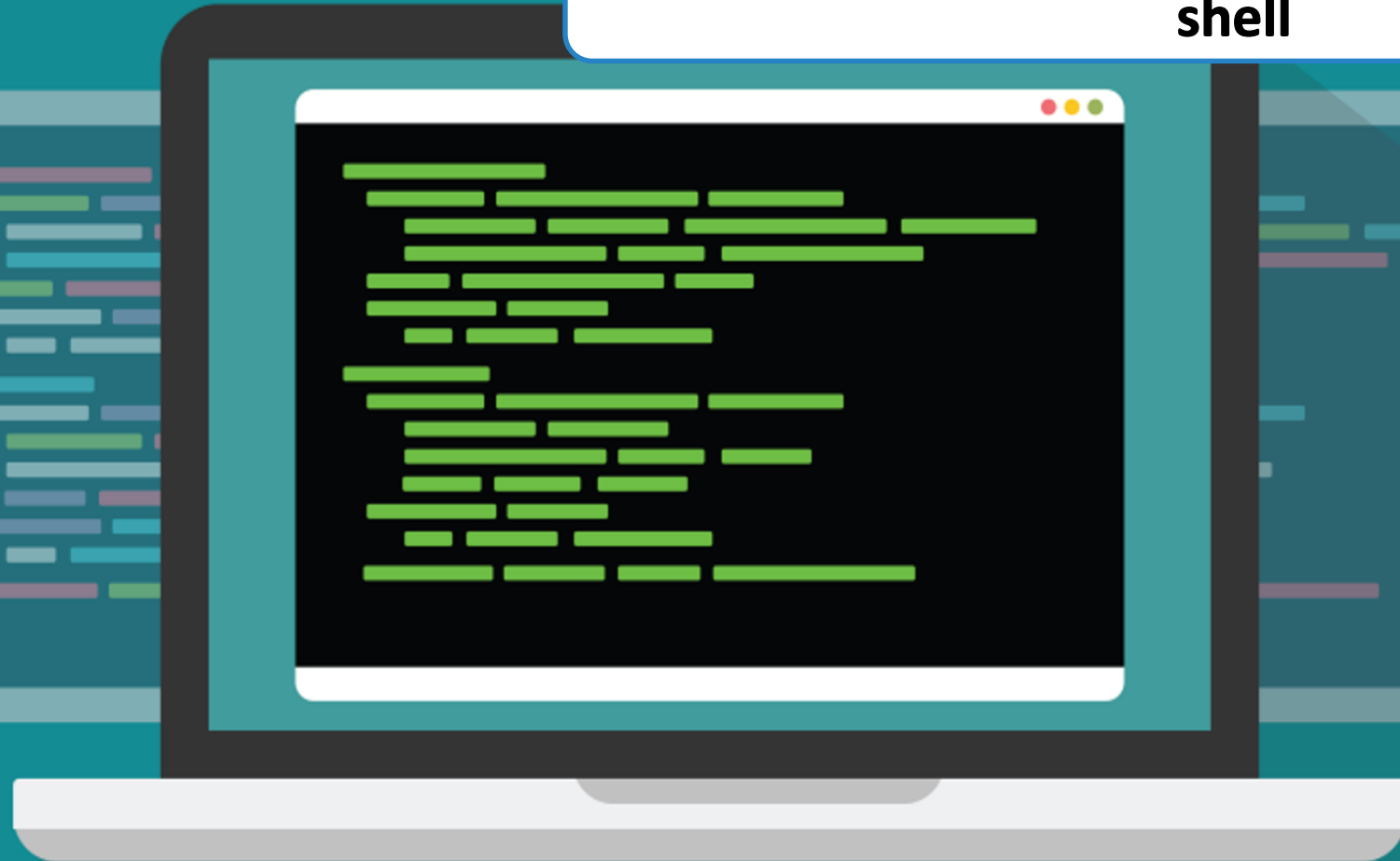
**IX. Contrôle des processus**

**X. Composition des commandes**

**XI. Les partitions de disque sur Linux**

# Chapitre X : Le terminal Linux

## I. Introduction à l'interpréteur de commandes : le shell





# I. Introduction à l'interpréteur de commandes : le shell

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 ;

- soit en ligne de commande,
- soit via une interface graphique.

# I. Introduction à l'interpréteur de commandes : le shell

**Remarque :** En mode ligne de commandes, on peut utiliser toutes les options des commandes contrairement à l'interface graphique qui n'offre, en général, qu'une partie.

Ce chapitre est consacré à **l'introduction et l'utilisation des commandes en mode ligne de commande (appelé aussi mode console ou mode interactif)**.

# I. Introduction à l'interpréteur de commandes : le shell

## I.1. Le terminal GNU/Linux

Un terminal est un programme qui émule une console dans une interface graphique, il permet de lancer des commandes.

Sur Ubuntu, on utilise l'application **"Terminal"** pour ouvrir **une console (le terme terminal est équivalent à console)**.

Il est parfois plus simple de taper une commande que d'effectuer des manipulations demandant beaucoup de clics de souris dans une interface graphique.

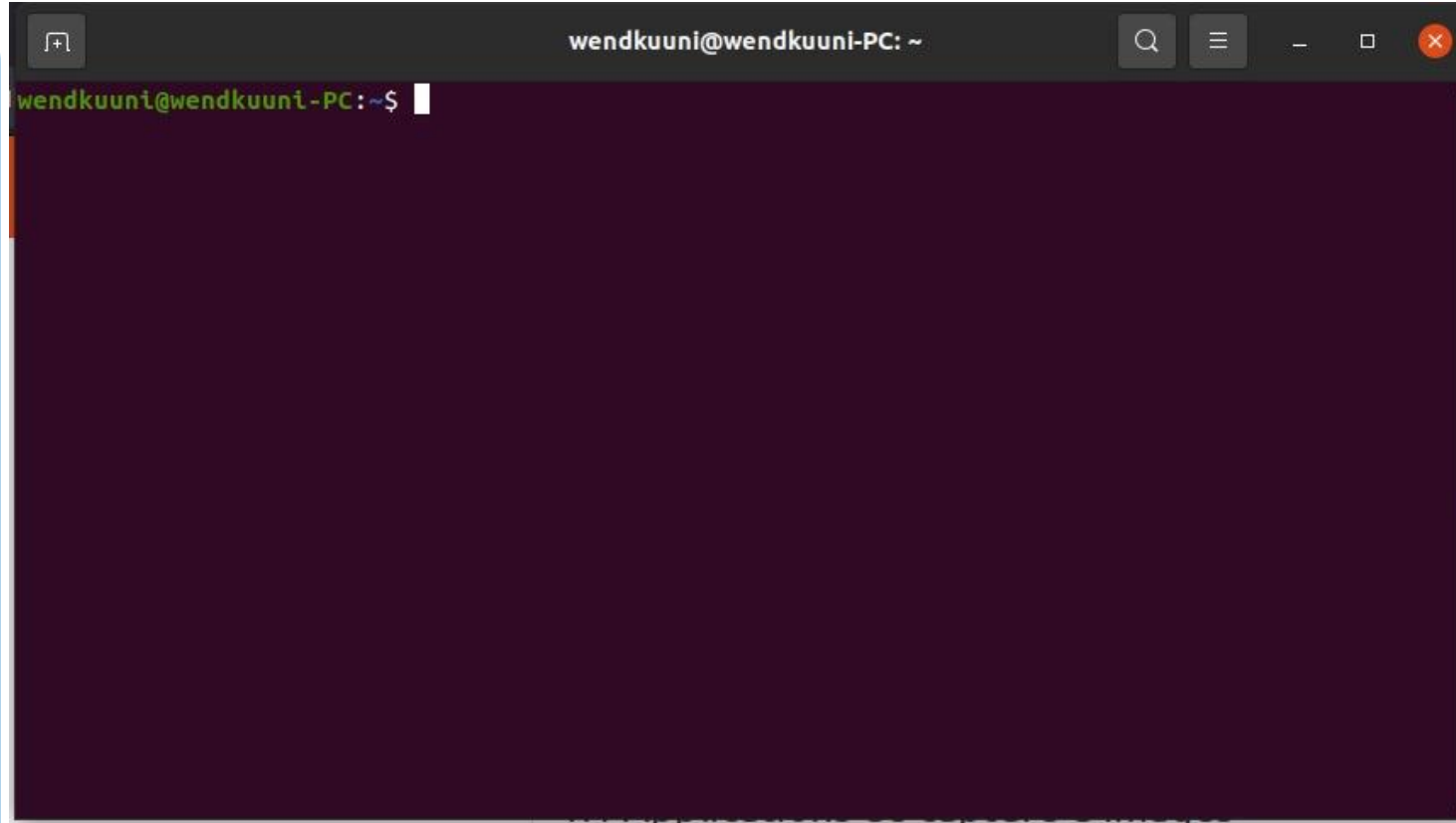
**C'est aussi un moyen plus simple pour expliquer comment faire quelque chose à quelqu'un (sur un forum par exemple), puisqu'il suffit d'indiquer la commande et non la suite de clics à effectuer sur l'interface graphique.**

# I. Introduction à l'interpréteur de commandes : le shell

## I.1. Le terminal GNU/Linux

Cependant, même si le terminal peut être beaucoup plus efficace qu'une interface graphique sous les doigts d'un utilisateur avancé, il est moins abordable que les interfaces graphiques.

Il est probable qu'aucune des deux méthodes (commandes ou interface graphique) ne remplacera complètement l'autre car elles se complètent plus qu'elles ne rivalisent.





# I. Introduction à l'interpréteur de commandes : le shell

## I.2. Le prompt

La première chose que l'on constate est ce qui s'appelle **le prompt**.

C'est le texte qui se trouve à gauche de la zone pour entrer la commande, et qui indique le contexte dans lequel on se trouve.

Le prompt se compose de en 3 parties principales :

- ☐ **le nom d'utilisateur**
- ☐ **le nom de la machine**
- ☐ **le dossier dans lequel on se trouve**

# I. Introduction à l'interpréteur de commandes : le shell

## I.2. Le prompt

Le nom d'utilisateur et le nom de la machine sont séparés par le caractère @, le caractère : permet de séparer le nom de la machine et le nom du dossier.

Dans notre cas le prompt est :



```
wendkuuni@wendkuuni-PC:~$
```

Ce qui signifie donc que :

- ☐ l'utilisateur est wendkuuni
- ☐ la machine se nomme wendkuuni-PC
- ☐ le dossier dans lequel on se trouve est ~ (c'est un raccourci pour dire que l'on est dans le répertoire de l'utilisateur)

Le symbole \$ à la fin signifie simplement que l'utilisateur est "normal", par opposition à un super-utilisateur (ou administrateur) symbolisé par le caractère #.

# I. Introduction à l'interpréteur de commandes : le shell

## I.3. Ligne de commande

Une **commande Linux** est un programme ou utilitaire qui s'exécute sur la ligne de commande Linux. La **ligne de commande (CLI)** est un terminal qui permet à l'utilisateur de taper des instructions texte et de les faire exécuter par le système.

**Une ligne de commande est donc une interface qui accepte des lignes de texte et les traite en instructions pour votre ordinateur.**

**En réalité, toute interface utilisateur graphique (GUI) n'est qu'une abstraction des programmes en ligne de commande.**

Par exemple, lorsque vous fermez une fenêtre en cliquant sur le « X », une commande est exécutée derrière cette action.

# I. Introduction à l'interpréteur de commandes : le shell

## I.3. Ligne de commande

**Un flag** est un moyen de passer des options à la commande que vous exécutez. La plupart des commandes Linux ont une page d'aide que l'on peut appeler avec le flag **-h**. La plupart du temps, les flags sont optionnels.

**Un argument ou paramètre** est l'entrée que nous donnons à une commande pour qu'elle puisse s'exécuter correctement. Dans la plupart des cas, l'argument est un chemin d'accès à un fichier, mais il peut s'agir de tout ce que vous saisissez dans le terminal.

Vous pouvez **invoquer des flags en utilisant des tirets (-) et des doubles tirets (--)**, tandis que l'exécution des arguments dépend de l'ordre dans lequel vous les passez à la fonction.



# I. Introduction à l'interpréteur de commandes : le shell

## I.4. 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 chaîne 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 sur 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**

Toutefois, on peut personnaliser le prompt.

# I. Introduction à l'interpréteur de commandes : le shell

## I.4. Principe du fonctionnement du shell en mode ligne de commande

### Remarques :

- Pour lancer un terminal depuis l'interface graphique : on utilise le menu du bureau ou le raccourcis clavier « **CTRL** » + « **Alt** » + « **T** » .
- Pour se connecter à une console virtuelle (un écran noir avec une invite de commande), depuis l'interface graphique : on utilise la combinaison des touches « **Ctrl+Alt+FN** », où N est un chiffre de 1 à 6 (il y a 6 consoles virtuelles désignées par « **tty1** » , ... « **tty6** »).
- Pour revenir au mode graphique depuis une console virtuelle, on utilise la combinaison des touches « **Ctrl+ALT+F7** ».

**N.B. Le mode graphique est désigné par « tty7 ».**

# I. Introduction à l'interpréteur de commandes : le shell

## I.5. Différence entre Terminal, Console, Shell et Ligne de commande :

Le tableau ci-dessous illustre les distinctions entre Terminal, Console, Shell et Ligne de commande :

Terminal	Console	Shell	Ligne de commande
Un terminal est un environnement de saisie et de sortie de texte.	Un terminal physique est appelé une console.	Le shell est un interpréteur de ligne de commande.	Une ligne de commande, également appelée invite de commande, est un type d'interface.
Un terminal est un programme wrapper qui exécute un shell et nous permet d'entrer des commandes.	La console est un type de terminal. C'est une fenêtre dans laquelle vos programmes en mode texte sont actifs.	Le shell est le programme qui traite réellement les commandes et affiche les résultats.	Une interface de ligne de commande est tout type d'interface utilisée pour saisir des commandes (textuelles). L'un d'eux est le terminal, mais certains programmes ont leurs propres interfaces de ligne de commande.
Le terminal est un programme qui affiche une interface graphique et permet d'interagir avec le shell.	La console se composait d'un seul clavier et d'un moniteur branchés sur un port de console série dédié sur un ordinateur pour une communication directe de bas niveau avec le système d'exploitation.	Un shell est une interface utilisateur permettant d'accéder aux services d'un système d'exploitation.	Une interface de ligne de commande (CLI) est un programme informatique qui traite les commandes sous forme de lignes de texte. L'utilisateur interagit généralement avec le shell via une interface de ligne de commande (CLI).
Le terme terminal peut également désigner un appareil qui permet aux utilisateurs d'interagir avec des ordinateurs, généralement via un clavier et un écran.	Une console est un terminal physique qui est le terminal principal directement connecté à une machine. La console est reconnue par le système d'exploitation comme un terminal (implémenté par le noyau).	Un shell est une interface principale que les utilisateurs voient lorsqu'ils se connectent, et sa fonction principale est de lancer d'autres programmes.	Une ligne de commande est une interface qui permet à un utilisateur de taper une commande (exprimée sous la forme d'une séquence de caractères, généralement un nom de commande suivi de certains paramètres), puis d'appuyer sur la touche Retour pour exécuter cette commande.

# Chapitre X : Le terminal Linux

## II. Commandes Unix en mode console





## II. Commandes Unix en mode console

N.B. Toutes les commande dans ce cours sont testées, sous **Linux Ubuntu version 20.04**, avec l'interpréteur de commandes le « **Bourne again shell : bash** ».

En rappel, une commande est un programme. Pour l'exécuter

- On tape son **nom**, éventuellement suivi d'**options** et d'**arguments**.
- A la fin de la saisi, on tape la touche « **Entrée** », pour valider la saisi.
- Lorsqu'on appui sur la touche « **Entrée** », **le shell exécute la commande**.

# II. Commandes Unix en mode console

## II.1. Syntaxe d'une commande:

**nom\_commande [options] [arguments]**

- « **nom\_commande** » est le nom de la commande à exécuter;
- « **options** » : une ou plusieurs options. Les options permettent de modifier le comportement de la commande.
- « **arguments** » : les arguments à passer à la commande.

### Remarques :

- Les crochets désignent un élément facultatif.
- Chaque mot est séparé des autres par un espace ou une tabulation.
- Une options est composée d'un tiret (-) suivi d'un seul caractère.
- Les options sont séparées par des espaces, par exemple « **-a -s -l -i** ». Mais il est possible d'accoler les options:

Par exemple: **-asli** désigne les options « **-a -s -l -i** ».

- Une commande peut avoir plusieurs arguments. Par exemple : **ls /etc/usr**  
la commande « **ls** » a deux arguments /etc et /usr .

# II. Commandes Unix en mode console

## II.1. Syntaxe d'une commande:

### Exemples :

#### ❖ ls

Liste (affiche) tous les fichiers du répertoire courant sauf les fichiers cachés (les fichiers commençant par « . »).

#### ❖ ls fichA

Si le fichier « fichA » existe, son nom sera affiché. Si le fichier « **fichA** », n'existe pas alors un message d'erreur est affiché.

#### ❖ ls -l

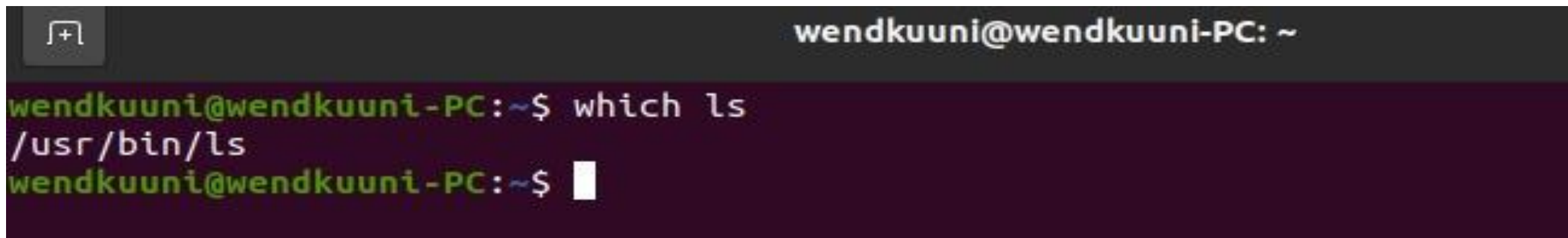
L'option -l permet de fournir plus de détails sur les fichiers contenus dans le répertoire courant.

## II. Commandes Unix en mode console

### II.1. Syntaxe d'une commande:

Pour **connaître la localisation de la commande** (connaître le chemin correspondant à une commande) on utilise la commande « **which** ».

Par exemple :

A terminal window with a dark background. The title bar at the top right says 'wendkuuni@wendkuuni-PC: ~'. The prompt is 'wendkuuni@wendkuuni-PC:~\$'. The command 'which ls' has been entered. The output is '/usr/bin/ls'. The prompt is now 'wendkuuni@wendkuuni-PC:~\$' followed by a cursor.

```
wendkuuni@wendkuuni-PC:~$ which ls
/usr/bin/ls
wendkuuni@wendkuuni-PC:~$
```

Le résultat d'exécution montre que la commande « **ls** » est dans le répertoire « **bin** » qui est un sous répertoire de « **usr** » qui est un sous-répertoire du répertoire « **root** ».



# II. Commandes Unix en mode console

## II.2. Les Types de commandes

Le shell est un processus, qui s'exécute sur la machine. Quand une commande est saisie:

❖ **Soit c'est le processus shell qui l'exécute (commande interne ou builtin).** Une commande interne ne possède pas d'exécutable associé à cette commande. Elle est intégrée au shell lui-même.

Lors de l'exécution d'une commande interne, il n'y a pas de création de processus fils pour exécuter cette commande.

❖ **Soit le shell crée un processus fils pour exécuter cette commande (commande externe).** Dans ce cas, le processus shell ne pourra pas exécuter une nouvelle commande qu'après la fin d'exécution du processus fils.

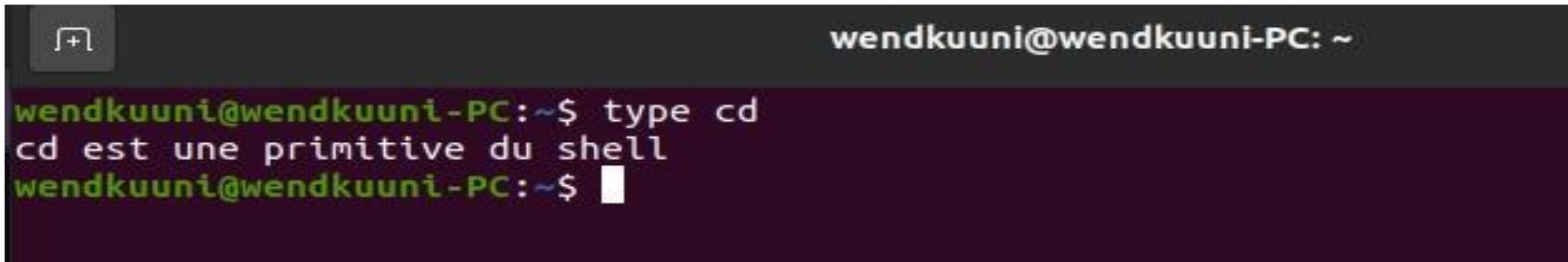
Pour identifier les types des commandes, on utilise la commande interne « **type** ».

# II. Commandes Unix en mode console

## II.2. Les Types de commandes

Exemple :

❖ type cd



```
wendkuuni@wendkuuni-PC: ~  
wendkuuni@wendkuuni-PC:~$ type cd  
cd est une primitive du shell  
wendkuuni@wendkuuni-PC:~$
```

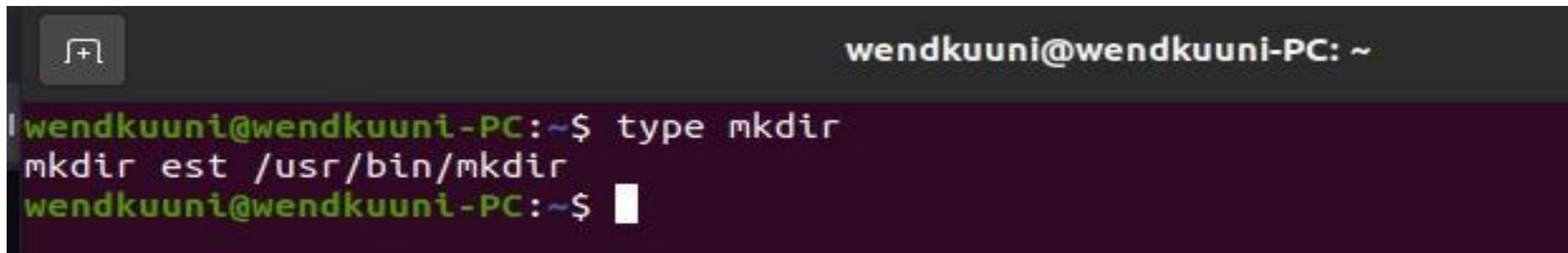
Le résultat de cette commande montre que la commande « cd » est une commande interne (shell builtin).

# II. Commandes Unix en mode console

## II.2. Les Types de commandes

Exemple :

❖ **type mkdir**

A terminal window with a dark background. The title bar at the top right says 'wendkuuni@wendkuuni-PC: ~'. The terminal shows the command 'type mkdir' being entered at the prompt 'wendkuuni@wendkuuni-PC:~\$'. The output is 'mkdir est /usr/bin/mkdir', followed by a new prompt 'wendkuuni@wendkuuni-PC:~\$' with a cursor.

```
wendkuuni@wendkuuni-PC: ~  
wendkuuni@wendkuuni-PC:~$ type mkdir  
mkdir est /usr/bin/mkdir  
wendkuuni@wendkuuni-PC:~$
```

L'exécution de cette commande affiche le chemin absolu de la commande « mkdir » qui montre que cette commande est située dans le sous-répertoire « /bin ». Donc la commande « mkdir » n'est pas une commande interne (n'est pas un builtin).

# II. Commandes Unix en mode console

## II.2. Les Types de commandes

### Liste de quelques commandes internes (builtins)

alias	bg	builtin	bind
cd	chdir	command	echo
eval	exec	exit	export
fc	fg	getopts	hash
jobid	jobs	pwd	read
readonly	set	setvar	shift
trap	type	ulimit	umask
unalias	unset	wait	

# II. Commandes Unix en mode console

## II.2. Les Types de commandes

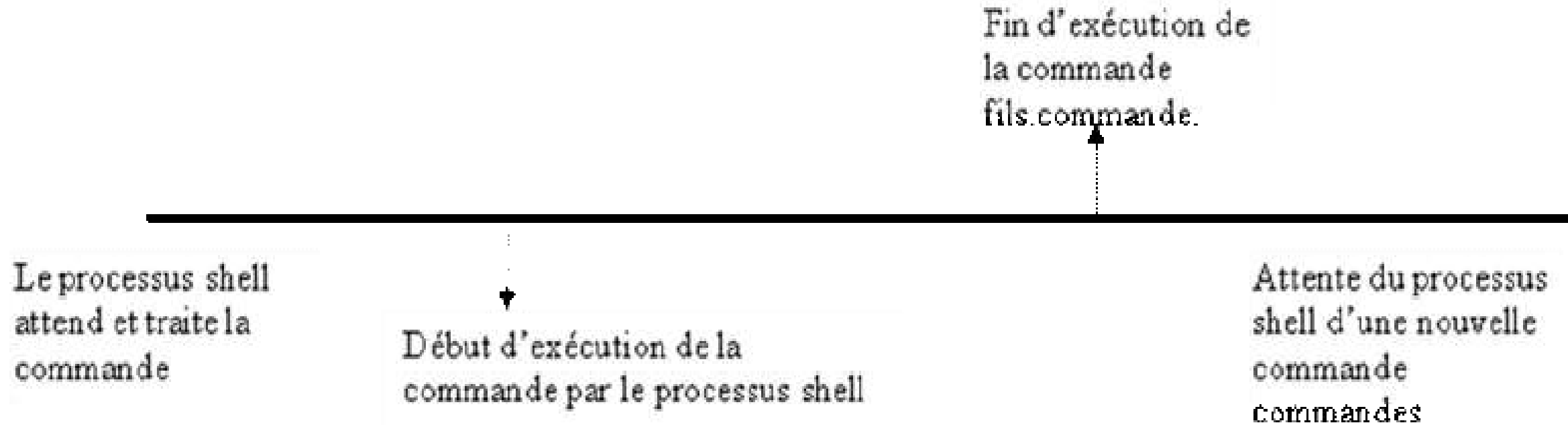
### Liste de quelques commandes externes

Les commandes externes au shell sont des exécutables (ne sont pas intégrées au shell lui-même) qui sont placées dans des répertoires; par exemple dans « /bin », « /sbin », « /usr/bin », ...etc.

/usr/bin/bg	/usr/bin/chgrp	/usr/bin/cmp	/usr/bin/comm
/usr/bin/cut	/usr/bin/diff	/usr/bin/dirname	/usr/bin/find
/usr/bin/grep	/usr/bin/head	/usr/bin/join	/usr/bin/man
/usr/bin/more	/usr/bin/nohup	/usr/bin/paste	/usr/bin/sed
/usr/bin/sort	/usr/bin/tail	/usr/bin/time	/usr/bin/top
/usr/bin/touch	/usr/bin/uniq	/usr/bin/vi	/usr/bin/w
/usr/bin/wc	/usr/bin/xargs	/usr/sbin/chown	

# II. Commandes Unix en mode console

## II.2. Les Types de commandes





# II. Commandes Unix en mode console

## II.2. Les Types de commandes

Si la commande est externe,  
alors le processus shell crée un  
processus fils pour exécuter la  
nouvelle commande

Le processus fils  
exécute la commande.

Fin d'exécution  
du processus  
fils commande.

Le processus shell est bloqué  
en attente de la fin  
d'exécution du processus fils

Attente du processus  
shell d'une nouvelle  
commande  
commandes

**Schéma d'exécution d'une commande externe**

# II. Commandes Unix en mode console

## II.3. La documentation Unix

Pour connaître les différentes options sur les commandes, on peut utiliser la commande « **man** », qui permet d'accéder au manuel en ligne où toutes les commandes sont documentées.

**man nom\_de\_la\_commande**

L'affichage des pages du manuel se fait par page. Pour se déplacer dans le manuel, on utilise

- La touche « **ENTRÉE** ou **RETURN** » pour avancer d'une ligne ;
- La touche « **ESPACE** » pour avancer d'une page ;
- Le caractère « **b** » pour reculer d'une page ;
- Pour quitter on tape le caractère « **q** ».

# II. Commandes Unix en mode console

## II.3. La documentation Unix

On peut aussi chercher la signification d'un élément, en utilisant la commande « **whatis** »

Exemple :

**whatis kill**

On peut aussi utiliser la commande « **help** », quand c'est possible, qui donne plus de détail sur la commande, mais moins d'informations que la commande « man ».

# II. Commandes Unix en mode console

## II.3. La documentation Unix

Exemple:

**help kill**

```
wendkuuni@wendkuuni-PC: ~  
wendkuuni@wendkuuni-PC:~$ whatis kill  
kill (1)          - send a signal to a process  
kill (2)          - send signal to a process  
wendkuuni@wendkuuni-PC:~$ help kill  
kill: kill [-s sigspec | -n signum | -sigspec] pid | jobspec ... ou kill -l [sigspec]  
    Envoie un signal à une tâche.  
  
    Envoie le signal nommé par SIGSPEC ou SIGNUM au processus identifié par  
    PID ou JOBSPEC. Si SIGSPEC et SIGNUM ne sont pas donnés, alors SIGTERM est  
    envoyé.  
  
Options :  
  -s sig      SIG est un nom de signal  
  -n sig      SIG est un numéro de signal  
  -l          affiche la liste des noms de signaux ; si des arguments suivent « -l »,  
              ils sont supposés être des numéros de signaux pour lesquels les noms doivent  
              être affichés  
  -L          synonyme de -l  
  
« kill » est une commande intégrée pour deux raisons : elle permet aux IDs de  
tâches d'être utilisés à la place des IDs de processus et elle permet aux  
processus d'être tués si la limite du nombre de processus que vous pouvez créer  
est atteinte.  
  
Code de sortie :  
Renvoie le code de succès à moins qu'une option non valable soit donnée ou qu'une  
erreur ne survienne.  
wendkuuni@wendkuuni-PC:~$
```

# Chapitre X : Le terminal Linux

## III. Les commandes liées à l'environnement



# III. Les commandes liées à l'environnement

## 1. La commande « **lsb\_release -a** »

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

**lsb\_release -a**

## 2. La Commande « **hostname** »

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

**hostname**

## 3. La commande **pwd** (Print Working Directory)

La commande « **pwd** » affiche le chemin d'accès absolu du répertoire de travail courant (le répertoire courant).

**pwd**

## 4. La Commande « **logname** »

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

**logname**



# III. Les commandes liées à l'environnement

## 5. La commande « who » :

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

**who**

## 6. La commande « whoami »

Cette commande affiche le nom d'utilisateur actuellement connecté à la session du terminal.

**whoami**

## 7. Commande « passwd »:

Cette commande permet à l'utilisateur de changer son mot de passe :

**Passwd**

Pour des raisons de sécurité, il faut taper tout d'abord le mot passe actuel (Old password), pour s'assurer que c'est l'utilisateur lui-même qui veut changer son mot de passe.

**N.B. :** L'administrateur peut définir des règles de sécurité sur les mots de passes, comme par exemple; le nombre minimum de caractères, les types des caractères, ... etc.

# III. Les commandes liées à l'environnement

## 8. Commande sudo

La commande **sudo** vous permet d'exécuter des programmes avec les privilèges de sécurité du **superutilisateur (root)**.

Il vous demande votre mot de passe personnel et confirme votre demande d'exécution d'une commande en vérifiant un fichier, appelé **sudoers (/etc/sudoers)**, que l'administrateur système configure.

À l'aide du fichier **sudoers**, les administrateurs système peuvent donner à certains utilisateurs ou groupes l'accès à certaines ou à toutes les commandes sans que ces utilisateurs aient à connaître le mot de passe du **root**.

Pour utiliser la commande sudo, à l'invite de commande, entrez :

**sudo nom\_de\_la\_commande**

# III. Les commandes liées à l'environnement

## 9. Commande « su »

La commande « **su** » (**Switch User**) permet d'ouvrir une session pour un autre utilisateur, par exemple, l'administrateur, peut se connecter en tant que « **root** », pour des tâches administratives, à partir de la session d'un utilisateur normal.

**Remarque :** Le mot de passe du nouvel utilisateur sera demandé.

**Syntaxe:**

**su [-] utilisateur**

# III. Les commandes liées à l'environnement

## 9. Commande « su »

**Exemple 1 : La commande « su » sans le tiret « - ».**

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

**su wendkuuni**

**Exemple 2 : Commande avec tiret « - » (su - utilisateur).**

Si le tiret « - » est précisé, le nouveau utilisateur se connecte avec son environnement de travail. Il est conseillé au super utilisateur « root » d'utiliser la commande avec tiret :su -

**su - wendkuuni**

# III. Les commandes liées à l'environnement

## 10. Commande « id »: notion d'identité (propriétaire et groupe) sous Unix

L'administrateur du système affecte à chaque utilisateur un identifiant unique (**UID : User Identifier**) lors de la création de leur compte.

Il associe aussi à chaque utilisateur, au moins un groupe (groupe principal) : le **GID (« Group Identifier »)**. Les utilisateurs sont définis dans le fichier « **/etc/passwd** » et les groupes sont définis dans le fichier « **/etc/group** ».

Les commandes

**cat /etc/passwd**

et

**cat /etc/group**

Permettent d'afficher les contenus de ces fichiers.

# III. Les commandes liées à l'environnement

## 10. Commande « id »: notion d'identité (propriétaire et groupe) sous Unix

Dès qu'un utilisateur est connecté sur une machine Unix/Linux, tous les processus générés, auront la même identité de l'utilisateur, c'est à dire il seront lancés avec l'UID et le GID de l'utilisateur.

La commande « id » permet d'obtenir les informations sur l'UID et le GID



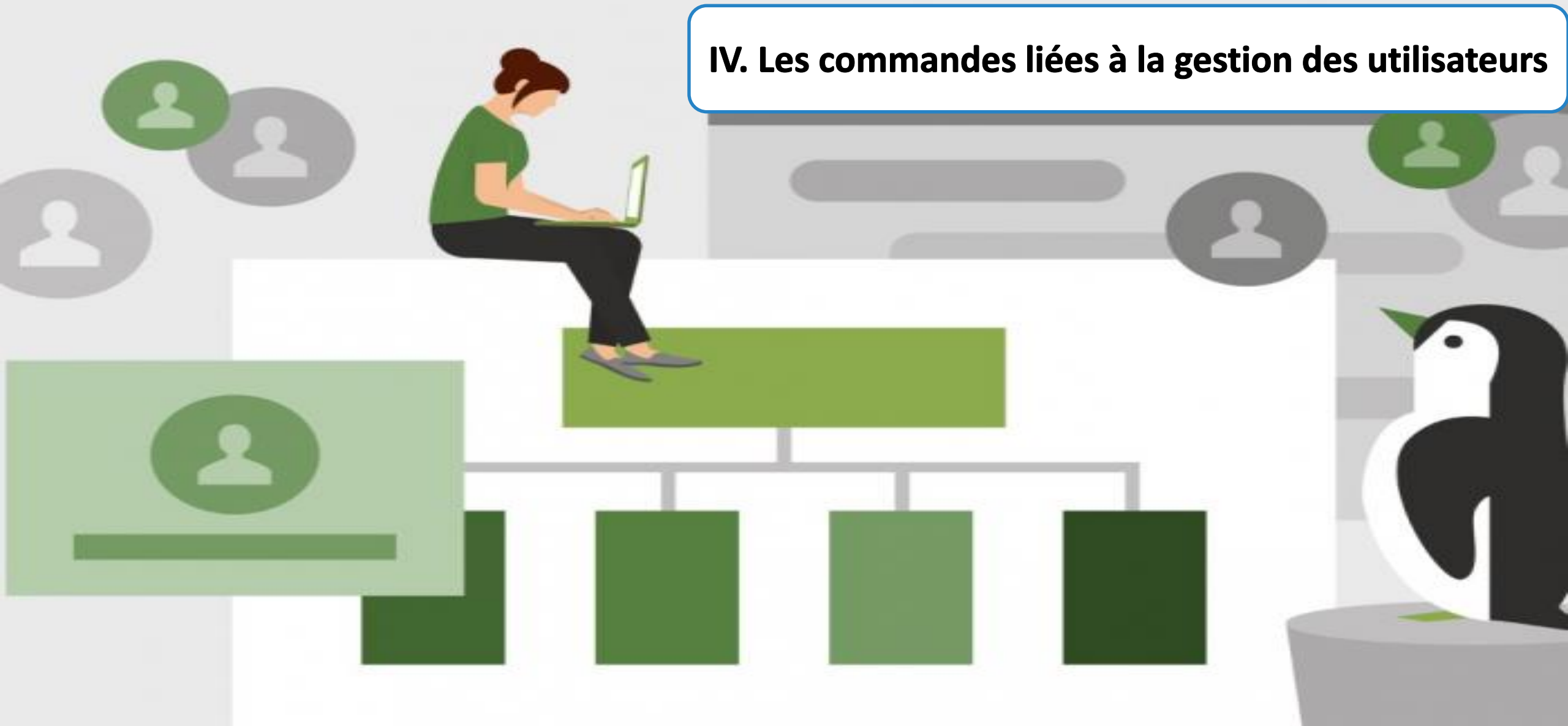
# III. Les commandes liées à l'environnement

## 11. Autres commandes

- La commande « **date** » : affiche la date et l'heure.
- La commande « **clear** » : efface l'écran.
- La commande « **exit** » : termine le shell (possible aussi **CTRL-D**, **logout**)

# Chapitre X : Le terminal Linux

## IV. Les commandes liées à la gestion des utilisateurs



## IV. Les commandes liées à la gestion des utilisateurs

Sur Linux, la configuration des informations utilisateur est stockée dans des fichiers et plus particulièrement dans le dossier de configuration **/etc**.

Les informations des utilisateurs sur Linux sont stockés dans quatre fichiers **/etc/** :

**/etc/passwd** décrit le nom et UID des utilisateurs Linux et leurs dossiers home

**/etc/groups** contient les informations des groupes utilisateurs

**/etc/shadow** stocke les mots de passe des utilisateurs

**/etc/gshadow** contient les mots de passe des groupes utilisateurs Linux

Pour voir leur contenu utilisez la commande **cat** : **cat /etc/passwd**

# IV. Les commandes liées à la gestion des utilisateurs

## IV.1. Créer, ajouter ou supprimer des utilisateurs Linux

A tout moment, il est possible d'ajouter ou supprimer des utilisateurs. Cela peut se faire graphiquement à partir des paramètres de la distribution Linux. Bien entendu, cela peut se faire en ligne de commandes à partir d'un terminal.

Pour se faire, vous devez être identifié en tant que root ou utiliser un utilisateur administrateur avec la commande sudo.

- ❑ **adduser ou useradd** : pour ajouter un utilisateur linux
- ❑ **userdel ou deluser** : supprimer un utilisateur linux
- ❑ **usermod** : La commande usermod modifie les fichiers d'administration des comptes du système selon les modifications qui ont été indiquées sur la ligne de commande
- ❑ **groupmod** : pour modifier la configuration d'un groupe utilisateur
- ❑ **passwd** : changer le mot de passe d'un utilisateur Linux

# IV. Les commandes liées à la gestion des utilisateurs

## IV.1. Créer, ajouter ou supprimer des utilisateurs Linux

### ❖ Ajouter un utilisateur Linux : **adduser**

La commande **adduser** peut s'utiliser de manière très simple.

Dans ce cas, les paramètres vont être pris par défaut par exemple le dossier home sera /home/toto :

**adduser toto**

Mais on peut aller plus loin en utilisant divers paramètres afin de pouvoir créer l'utilisateur comme on le souhaite.

# IV. Les commandes liées à la gestion des utilisateurs

## IV.1. Créer, ajouter ou supprimer des utilisateurs Linux

❖ Ajouter un utilisateur Linux : **adduser**

Par exemple ici on fixe le shell, l'uid et le répertoire du home de l'utilisateur

**adduser toto --shell /bin/bash --home /home/supertoto --uid 2000 --disabled-password**

L'option **--disabled-password** permet d'interdire l'identification par mot de passe, **cela peut être intéressant pour un utilisateur SSH afin de forcer l'authentification par clé RSA.**

# IV. Les commandes liées à la gestion des utilisateurs

## IV.1. Créer, ajouter ou supprimer des utilisateurs Linux

❖ Ajouter un utilisateur Linux : **useradd**

TAF : Avec la commande **useradd** créer un utilisateur



# IV. Les commandes liées à la gestion des utilisateurs

## IV.1. Créer, ajouter ou supprimer des utilisateurs Linux

### ❖ Ajouter un utilisateur Linux dans un groupe

Enfin **adduser** vous permet d'ajouter un utilisateur Linux dans un groupe utilisateur.

La syntaxe est la suivante :

**adduser NomUtilisateur NomGroupe**

Mais on peut aussi utiliser **usermod** pour ajouter un utilisateur dans un ou plusieurs groupes.

Cela donne :

**usermod -a -G groupe1,groupe2 NomUtilisateur**

# IV. Les commandes liées à la gestion des utilisateurs

## IV.1. Créer, ajouter ou supprimer des utilisateurs Linux

❖ Supprimer un utilisateur Linux : **deluser**

Cette commande permet de supprimer un utilisateur :

**deluser nomUtilisateur**

Cela ne supprime pas son home, ainsi il faudra le faire manuellement.

Par exemple :

**sudo rm -R /home/utilisateur**

# IV. Les commandes liées à la gestion des utilisateurs

## IV.1. Créer, ajouter ou supprimer des utilisateurs Linux

❖ Supprimer un utilisateur Linux : **userdel**

TAF : Après avoir créé un compte, supprimez le avec la commande **userdel**

# IV. Les commandes liées à la gestion des utilisateurs

## IV.1. Créer, ajouter ou supprimer des utilisateurs Linux

### ❖ Ajouter un utilisateur à un groupe : **addgroup**

Le commande **addgroup** permet de créer un groupe. Ce dernier va donc être ajouté aux fichiers **/etc/group**.

La syntaxe est la suivante pour créer un groupe utilisateur :

**addgroup [--gid ID] NomDuGroupe**

Si vous ne spécifié pas l'ID, Linux prendra le numéro libre suivant.

Enfin pour créer un groupe système :

**addgroup --system [--gid ID] NomDuGroupe**

# IV. Les commandes liées à la gestion des utilisateurs

## IV.1. Créer, ajouter ou supprimer des utilisateurs Linux

Ajouter un utilisateur à un groupe : **groupadd**

TAF : Avec la commande **groupadd** ajouter un utilisateur dans un groupe existant dans le fichier **/etc/group**.

# IV. Les commandes liées à la gestion des utilisateurs

## IV.1. Créer, ajouter ou supprimer des utilisateurs Linux

### ❖ Modifier la configuration utilisateur : **usermod**

**usermod** permet de modifier la configuration d'un utilisateur. Plus haut on a vu qu'il pouvait modifier l'appartenance à un group utilisateur mais **il permet aussi de modifier le nom d'un utilisateur ou UID.**

Ajouter des informations à un utilisateur Linux :

**usermod -c "C'est un super utilisateur" nomUtilisateur**

Pour changer l'identifiant d'un utilisateur :

**usermod --login nouvelIdentifiant**

ou

**usermod -l nouvelIdentifiant**

Modifier l'UID d'un utilisateur Linux :

**usermod -u 888 nomUtilisateur**

# IV. Les commandes liées à la gestion des utilisateurs

## IV.1. Créer, ajouter ou supprimer des utilisateurs Linux

❖ Modifier la configuration utilisateur : **usermod**

Enfin **usermod** permet **de jouer sur les dates d'expirations du mot de passe.**

Par exemple pour verrouiller un compte utilisateur :

**usermod -e YYYY-MMM-DD nomUtilisateur**

Pour le réactiver :

**usermod --expiredate "" nomUtilisateur**

ou

**usermod -U nomUtilisateur**

# IV. Les commandes liées à la gestion des utilisateurs

## IV.1. Créer, ajouter ou supprimer des utilisateurs Linux

❖ Modifier la configuration d'un groupe utilisateur : **groupmod**

**groupmod** permet de modifier la configuration d'un groupe utilisateur.

Par exemple pour changer le nom d'un groupe :

**groupmod -n new-name nouveauNomGroupe nomGroupeActuel**

Ou encore modifier le GID d'un groupe :

**groupmod -g nouveauID nomGroupe**



# IV. Les commandes liées à la gestion des utilisateurs

## IV.1. Créer, ajouter ou supprimer des utilisateurs Linux

❖ Changer/supprimer le mot de passe d'un utilisateur : **passwd**

Enfin la commande **passwd** permet de changer le mot de passe d'un utilisateur linux.

Le nouveau mot de passe sera demandé en confirmation.

Mais la commande **passwd** va plus loin.

# IV. Les commandes liées à la gestion des utilisateurs

## IV.1. Créer, ajouter ou supprimer des utilisateurs Linux

❖ Changer/supprimer le mot de passe d'un utilisateur : **passwd**

Par exemple pour supprimer un mot de passe d'un utilisateur :

**passwd -l nomUtilisateur**

Enfin pour remettre en place le mot de passe :

**passwd -u nomUtilisateur**

Mais aussi verrouiller un utilisateur :

**passwd -l nomUtilisateur**

Mettre une date d'expiration de l'utilisateur qui sera verrouillé ensuite :

**passwd -e YYYY-MM-DD nomUtilisateur**

**On peut aussi l'utiliser pour réinitialiser le mot de passe root.**

# IV. Les commandes liées à la gestion des utilisateurs

## IV.1. Créer, ajouter ou supprimer des utilisateurs Linux

❖ Changer/supprimer le mot de passe d'un utilisateur : **passwd**

**TAF** : Réinitialiser le mot de passe root perdu ou oublié Ubuntu

**TAF** : Réinitialiser le mot de passe root avec un **Live USB Rescatux**

# IV. Les commandes liées à la gestion des utilisateurs

## IV.1. Créer, ajouter ou supprimer des utilisateurs Linux

❖ Afficher les informations d'un utilisateur Linux : **id** et **chage**

La commande **chage** permet d'afficher les dates d'expirations d'un utilisateur Linux.

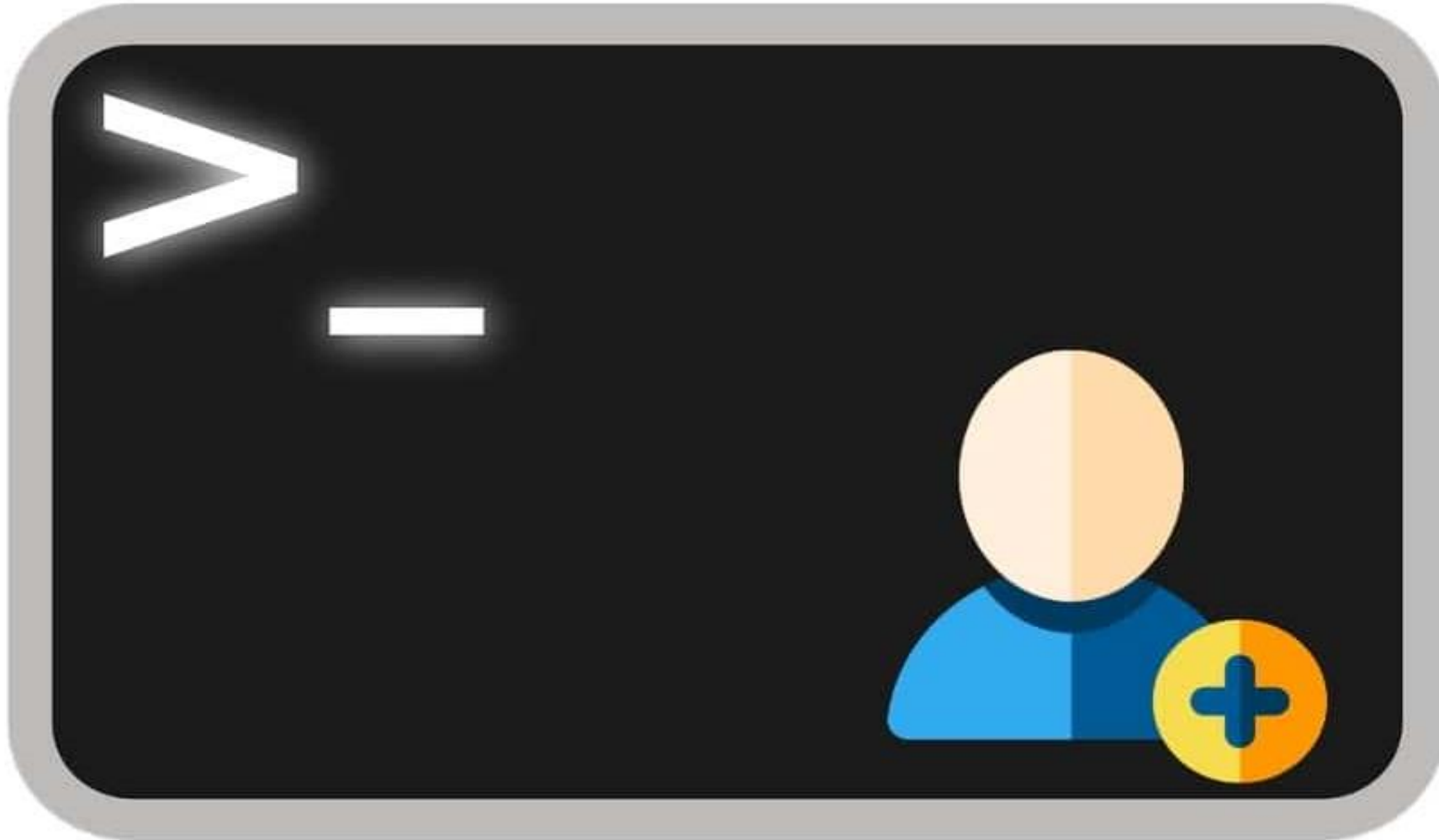
Alors que la commande **id** affiche le shell, Home, UID, etc.

```
chage -l nom_utilisateur
```

```
id nom_utilisateur
```

# Chapitre X : Le terminal Linux

## V. Les permissions de fichiers sur Linux



# V. Les permissions de fichiers sur Linux

Sur Linux, **pour protéger l'accès à un répertoire ou fichier ou contre sa modification ou suppression, on attribue des autorisations et permissions de fichiers.**

Dans cette partie, **nous allons examiner le fonctionnement des permissions et autorisations de fichiers fonctionnent sur Linux.**

# V. Les permissions de fichiers sur Linux

## V.1. Propriétaire, groupes et tes types de permissions

Le compte d'utilisateur vous permet de vous connecter à Linux et définit les permissions et autorisations que vous pouvez effectuer.

Ainsi on distingue :

- ❑ **Propriétaire** : L'utilisateur et le groupe possédant le fichier
- ❑ **Autorisations** : représentent l'ensemble des actions qu'un utilisateur peut effectuer selon le type de son compte et le groupe auquel il appartient.

La commande **whoami** permet d'afficher l'utilisateur avec lequel vous êtes connecté.  
La commande **groups** ou **id** affiche les groupes auquel l'utilisateur appartient.

# V. Les permissions de fichiers sur Linux

## V.1. Propriétaire, groupes et tes types de permissions

Sur Linux, il existe **trois principales autorisations de fichiers** qui permettent de déterminer les accès ou les modifications de fichiers.

Permission	Description
read (r)	Les autorisations de lecture d'un fichier ou répertoire
write (w)	Les autorisations d'écrire sur un fichier ou répertoire
execute (x)	Les autorisations pour exécuter un fichier ou accéder à un répertoire



# V. Les permissions de fichiers sur Linux

## V.1. Propriétaire, groupes et tes types de permissions

Les fichiers et les répertoires peuvent appartenir au propriétaire du **fichier (u)**, du **groupe (g)** ou **d'autres (o)**.

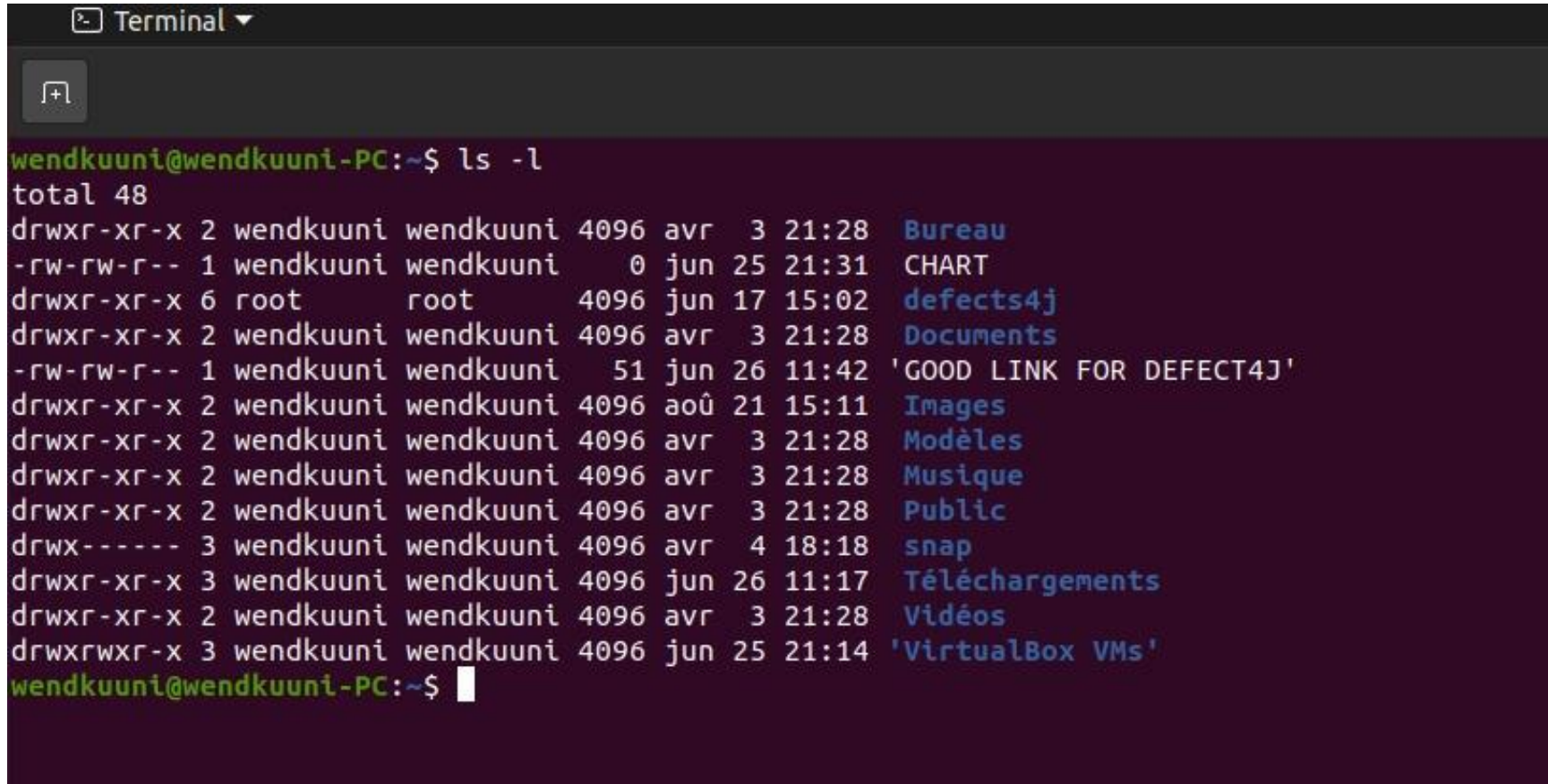
Les appartenances	Description
u (user)	Le propriétaire du fichier ou répertoire
g (group)	Le groupe
o (other)	Autres

# V. Les permissions de fichiers sur Linux

## V.1. Propriétaire, groupes et tes types de permissions

Pour **afficher les permissions**, on peut utiliser la commande **ls** avec l'option **-l** :

**ls -l**



```
wendkuuni@wendkuuni-PC:~$ ls -l
total 48
drwxr-xr-x 2 wendkuuni wendkuuni 4096 avr  3 21:28 Bureau
-rw-rw-r-- 1 wendkuuni wendkuuni   0 jun 25 21:31 CHART
drwxr-xr-x 6 root      root      4096 jun 17 15:02 defects4j
drwxr-xr-x 2 wendkuuni wendkuuni 4096 avr  3 21:28 Documents
-rw-rw-r-- 1 wendkuuni wendkuuni  51 jun 26 11:42 'GOOD LINK FOR DEFECT4J'
drwxr-xr-x 2 wendkuuni wendkuuni 4096 aoû 21 15:11 Images
drwxr-xr-x 2 wendkuuni wendkuuni 4096 avr  3 21:28 Modèles
drwxr-xr-x 2 wendkuuni wendkuuni 4096 avr  3 21:28 Musique
drwxr-xr-x 2 wendkuuni wendkuuni 4096 avr  3 21:28 Public
drwx----- 3 wendkuuni wendkuuni 4096 avr  4 18:18 snap
drwxr-xr-x 3 wendkuuni wendkuuni 4096 jun 26 11:17 Téléchargements
drwxr-xr-x 2 wendkuuni wendkuuni 4096 avr  3 21:28 Vidéos
drwxrwxr-x 3 wendkuuni wendkuuni 4096 jun 25 21:14 'VirtualBox VMs'
wendkuuni@wendkuuni-PC:~$
```

# V. Les permissions de fichiers sur Linux

## V.1. Propriétaire, groupes et tes types de permissions

Ce qui donne la structure suivante qui se lit de gauche à droite :

```
drwxr-xr-x 2 wendkuuni wendkuuni 4096 avr  3 21:28 Documents
```

Tout à gauche , nous avons les permissions de fichiers et répertoires qui vont par trois pour **rwX**.

Le signe **-** indique que la permission n'est pas présente.

- ☐ **rwX-** : correspond aux **permissions du propriétaire** soit donc ici lecture, écriture (pour **rw**) et exécution (**x**). Lorsqu'il s'agit d'un répertoire, la première lettre est **d** pour **directory**.
- ☐ **xr-** : aux **permissions du groupe**, ici que l'exécution et lecture car **xr**
- ☐ **x-** : aux **permissions pour autres**

# V. Les permissions de fichiers sur Linux

## V.1. Propriétaire, groupes et tes types de permissions

```
drwxr-xr-x 2 wendkuuni wendkuuni 4096 avr  3 21:28 Documents
```

Puis ensuite sont indiqués **le propriétaire et le groupe**, ici le propriétaire est **wendkuuni** et le groupe porte le même nom.

Enfin, **la dernière colonne** correspond **au nom du fichier ou du dossier** ici **Document**.

# V. Les permissions de fichiers sur Linux

## V.2. Les permissions de fichiers Linux en nombres décimaux

En plus de l'utilisation **rwX**, il existe une notation en nombre décimal.

Linux utilise le système binaire pour stocker les permissions dans le système de fichiers.

Ainsi, il est possible de faire correspondre un nombre décimal.

Chaque catégorie calcule le nombre symbolique en convertissant les autorisations existantes à sa forme décimale attribuant un "**un**" si vous avez la permission et un "**zéro**" si vous n'avez pas la permission.

Ainsi on obtient ceci :

# V. Les permissions de fichiers sur Linux

## V.2. Les permissions de fichiers Linux en nombres décimaux

Ainsi on obtient ceci :

Permissions	Notation décimale
r (lecture)	4
w (écriture)	2
x (exécution)	1

# V. Les permissions de fichiers sur Linux

## V.2. Les permissions de fichiers Linux en nombres décimaux

### Correspondance permissions en nombre décimal

Ensuite on additionne les valeurs pour calculer le nombre décimal :

Permissions	Notation décimale
<b>rwX</b>	7 (4+2+1)
<b>rw</b>	6 (4+2)
<b>rx</b>	5 (4+1)
<b>wX</b>	3 (2+1)

- Le premier chiffre correspond **aux droits d'accès pour le propriétaire.**
- Le deuxième chiffre correspond **aux droits d'accès pour le groupe.**
- Le troisième chiffre correspond **aux droits d'accès pour les autres utilisateurs.**

# V. Les permissions de fichiers sur Linux

## V.2. Les permissions de fichiers Linux en nombres décimaux

### Correspondance permissions en nombre décimal

Ensuite on additionne les valeurs pour calculer le nombre décimal :

Permissions	Notation décimale
<b>rwX</b>	7 (4+2+1)
<b>rw</b>	6 (4+2)
<b>rx</b>	5 (4+1)
<b>wX</b>	3 (2+1)

Par exemple **756**

**7 = rwX** représente les droits pour le propriétaire.

**6 = rw-** représente les droits pour le groupe.

**5 = r-X** représente les droits pour les autres.



# V. Les permissions de fichiers sur Linux

## V.3. CHMOD

**chmod** est la commande Linux pour **attribuer, changer, modifier ou supprimer les permissions de fichiers et répertoires.**

Dans les systèmes **Linux / Unix**, l'accessibilité aux fichiers et aux répertoires est déterminée par le propriété et les autorisations de fichiers.

# V. Les permissions de fichiers sur Linux

## V.3. CHMOD

La syntaxe chmod est la suivante :

**chmod [OPTIONS] MODE fichier**

- ❑ **Options** : les paramètres de la commande par exemple +R pour rendre récursif
- ❑ **MODE** : les permissions à expliquer soit en texte, soit en décimale (chiffre)
- ❑ **Fichier** : le nom du fichier

# V. Les permissions de fichiers sur Linux

## V.3. CHMOD

### ❖ Changer des permissions à l'aide de la notation numérique

Pour affecter des autorisations de lecture, d'écriture et d'exécution du propriétaire et de lire des autorisations uniquement au groupe et aux autres utilisateurs (**rw~~x~~rw-rw-**), exécutez la commande :

**chmod 744 fichier.txt**

**7 = rwx** représente les droits pour le propriétaire.

**4 = rw-** représente les droits pour le groupe.

**4 = rw-** représente les droits pour les autres.

# V. Les permissions de fichiers sur Linux

## V.3. CHMOD

### ❖ Changer des permissions à l'aide de la notation numérique

Pour affecter toutes les autorisations au propriétaire du fichier, lire et exécuter des autorisations au groupe et aucune autorisation à d'autres utilisateurs (**rwxr-x-**), exécuter :

**chmod 750 coders.txt**

# V. Les permissions de fichiers sur Linux

## V.3. CHMOD

### ❖ Changer des permissions à l'aide de la notation de texte

Voici comment utiliser chmod avec les permissions en texte ; la syntaxe est la suivante :

**chmod [OPTIONS] [ u g o a ] [ - + = ] [ r, w, x ] fichier**

Le premier groupe de paramètres définit à quelle catégorie vous appliquez les permissions (Les appartenances de fichiers sur Linux).

Les catégories	Description
<b>u (user)</b>	Affecter les permissions à l'utilisateur
<b>g (group)</b>	Affecter les permissions au groupe
<b>o (other)</b>	Affecter les permissions à autre
<b>a (all)</b>	Pour appliquer à toutes les catégories

# V. Les permissions de fichiers sur Linux

## V.3. CHMOD

### ❖ Changer des permissions à l'aide de la notation de texte

Voici comment utiliser chmod avec les permissions en texte ; la syntaxe est la suivante :

**chmod [OPTIONS] [ u g o a ] [ - + = ] [ r, w, x ] fichier**

Le deuxième ensemble d'options définit l'action, si vous ajoutez ou supprimez des permissions (Les drapeaux d'autorisations de chmod).

Drapeaux	Description
-	Supprime les autorisations de fichier à partir d'un utilisateur spécifié.
+	Ajoute des autorisations à un utilisateur spécifié.
=	Attribue des autorisations distinctes des utilisateurs spécifiés et supprime les autorisations précédentes du segment utilisateur.

# V. Les permissions de fichiers sur Linux

## V.3. CHMOD

### ❖ Changer des permissions d'écriture d'un fichier

De la même manière pour changer les permissions en écriture seules sur un fichier pour le propriétaire :

**chmod o=w fichier**

Pour changer les permissions en écriture seules sur le groupe à un fichier :

**chmod g=w fichier**

# V. Les permissions de fichiers sur Linux

## V.3. CHMOD

❖ **Changer les permissions d'exécution au fichier au propriétaire et groupe**

Pour ajouter les autorisations d'exécution au propriétaire et groupe à un fichier :  
**chmod ug+x fichier**

Pour n'avoir que les permissions seulement en exécution :  
**chmod ug=x fichier**



# V. Les permissions de fichiers sur Linux

## V.3. CHMOD

❖ **Attribuer différentes autorisations de fichier au groupe et aux autres**

Pour attribuer plusieurs permissions en les séparant par des virgules, par exemple :  
**chmod ug+rxw,o-rwx fichier**

Ou encore :  
**chmod ug+rxw,o= fichier**

Ce qui équivaut aussi à :  
**chmod 770 fichier**

# V. Les permissions de fichiers sur Linux

## V.3. CHMOD

### ❖ Supprimer toutes les autorisations pour les autres utilisateurs

Pour supprimer toutes les autorisations pour les autres utilisateurs, vous pouvez utiliser le drapeaux = sans rien spécifier.

**chmod o= fichier**

ou encore utilisez chmod comme ceci :

**chmod o-rwx fichier**

Mais aussi en mettant 0 sur la dernière catégorie, par exemple :

**chmod 660 fichier**

**chmod 770 fichier**

# V. Les permissions de fichiers sur Linux

## V.4. CHOWN

**chown** est une commande qui permet de changer le propriétaire d'un fichier sur Linux.

La syntaxe n'est pas très complexe mais on peut avoir besoin d'aide pour utiliser la commande chown. Dans cette partie, on verra comment utiliser la commande chown avec quelques exemples.

Voici la syntaxe de **chown** :

**chown [OPTIONS] UTILISATEUR[:GROUPE] FICHIER(s)**

Le nom d'utilisateur est indiqué suivi du séparateur : et le nom du groupe puis on spécifie le nom du fichier.

# V. Les permissions de fichiers sur Linux

## V.4. CHOWN

❖ Changer le propriétaire de fichier\_1 avec l'utilisateur User\_1.

```
chown User_1 fichier_1
```

Vous pouvez aussi changer le propriétaire d'un fichier ou répertoire, en spécifiant les deux.

```
chown User_1 fichier_1 repertoire_1
```

Enfin, on peut aussi spécifier l'**UID** de l'utilisateur à la place du nom du propriétaire.

En rappel, la liste des UID se trouve dans **/etc/passwd**

```
chown 1000 fichier_1
```

# V. Les permissions de fichiers sur Linux

## V.4. CHOWN

### ❖ Changer le propriétaire et le groupe d'un fichier

La syntaxe de **chown** pour changer le propriétaire et le groupe d'un fichier (le nom d'utilisateur et le groupe sont séparés par le caractère :) :

**chown UTILISATEUR:GROUPE FICHIER**

Ainsi, pour changer le propriétaire en User\_1 et le groupe en utilisateurs sur fichier\_1 :

**chown User\_1:utilisateurs fichier1**

Si vous omettez le nom du groupe après le séparateur (:) Le groupe du fichier est remplacé par le groupe de connexion de l'utilisateur spécifié :

**chown User\_1: file1**

# V. Les permissions de fichiers sur Linux

## V.4. CHOWN

### ❖ Changer le groupe d'un fichier

Pour changer le groupe d'un fichier, on utilise le séparateur : suivi du nom de groupe.

**chown :GROUPE FICHIER**

Par exemple pour attribuer le groupe www-data au fichier /var/www/fichier1 :

**chown :www-data /var/www/fichier1**

Il existe aussi la commande **chgrp** qui s'utilise comme ceci :

**chgrp www-data /var/www/fichier1**

# V. Les permissions de fichiers sur Linux

## V.4. CHOWN

### ❖ Changer récursivement le propriétaire d'un répertoire

Voici la syntaxe pour changer le propriétaire d'un répertoire en récursif, soit donc modifier le propriétaire de toute l'arborescence fichiers et sous-répertoire, on utilise l'option -R :

**chown -R UTILISATEUR:GROUPE REPERTOIRE**

Par exemple pour changer l'utilisateur en www-data du répertoire /var/www et les sous fichiers et sous-répertoire :

**chown -R www-data: /var/www**

# Chapitre X : Le terminal Linux

## VI. Gestionnaire de package Linux





# VI. Gestionnaire de package Linux

**APT pour Advanced Package Tool** est un utilitaire présent sur les distributions Linux Debian comme **Ubuntu, Mint, Kali**, etc

Cet outil essentiel permet d'installer, supprimer un logiciel ou encore mettre à jour ce dernier.

Enfin avec apt, vous pouvez aussi mettre à jour votre distribution Linux.  
Plusieurs commandes existent comme **apt-get, apt-cache**, etc.

Cet partie vous présente l'utilisation d'APT :

- ☐ comment **installer ou supprimer un paquet (package en anglais)**
- ☐ comment **mettre à jour un logiciel**
- ☐ et enfin comment **mettre à jour votre distribution Linux**

En effet **apt** permet d'installer des logiciels sur les distributions à base de Debian depuis les **dépôts (repository)**.

# VI. Gestionnaire de package Linux

## VI.1. Les packages de distribution

Un mainteneur s'occupe de maintenir un package qui permet d'installer un logiciel sur la distribution.

Il s'agit d'un fichier **.deb** que le mainteneur crée et maintient.

Ce fichier permet de jouer des scripts afin de placer les fichiers de configuration et de l'application dans Linux : **en clair donc d'installer les paquets et applications dans Linux.**

# VI. Gestionnaire de package Linux

## VI.1. Les packages de distribution

### ❖ DPKG

**dpkg (Debian Package)** est le **gestionnaire de paquets** des **distributions Linux Debian** et ses dérivés **Ubuntu, Mint**, etc.

Elle se présente sous la forme d'une commande pour lister, installer ou supprimer un paquet **.deb**.

Les paquets Debian étant des fichiers **.deb** que l'on peut télécharger sur internet.

Il contient les fichiers d'installation et de configuration pour installer une application, librairies dans le système d'exploitation.

# VI. Gestionnaire de package Linux

## VI.1. Les packages de distribution

### ❖ DPKG

**Les dépôts (respositories en anglais)** sont des serveurs mis en ligne qui stockent tous les **.deb** d'une distribution.

Ainsi, **apt** se connecte à l'un de ses serveurs afin de télécharger le **.deb** et l'installer.

On appelle cela les sources. Pour ce fait, apt appelle la commande **dpkg**.

Un dépôt peut stocker plus de **20 000 fichiers .deb**.

# VI. Gestionnaire de package Linux

## VI.1. Les packages de distribution

### ❖ DPKG

Enfin apt peut aussi mettre à jour une distribution Linux.

Cela consiste donc à télécharger tous les **.deb** de la nouvelle version et les installer.

Les logiciels systèmes comme **udev, libc** ou encore le noyau linux sont aussi mis à jour par cette méthode.

Du côté des distributions **Redhat**, l'équivalent est **yum/DNF**. Son fonctionnement est assez proche.

# VI. Gestionnaire de package Linux

## VI.2. Les fichiers de configuration

Afin de fonctionner **apt** utilise plusieurs fichiers et dossiers.

Voici une liste de ces derniers.

- ❑ **/etc/apt/sources.list** : stocke les sources avec l'adresse des dépôts
- ❑ **/etc/apt/sources.list.d/** : sources additionnels. Ainsi on peut ajouter des dépôts non officielles.
- ❑ **/etc/apt/apt.conf** : fichier de configuration APT
- ❑ **/etc/apt/apt.conf.d/** : fichiers de configuration additionnels.
- ❑ **/etc/apt/preferences.d/** : Les fichiers de préférences additionnels.
- ❑ **/var/cache/apt/archives/** : Stocke les packages (fichiers .deb)
- ❑ **/var/lib/apt/lists/** : stocke la liste et informations sur les packages du systèmes.

# VI. Gestionnaire de package Linux

## VI.3. Les commandes apt

Apt regroupe différentes commandes selon les besoins. Voici une liste de ces commandes :

- ❑ **apt-get** : commande principale qui se décline en différents paramètres. Vous trouverez une liste ci-dessous.
- ❑ **apt-cache** : afin d'effectuer une recherche de package dans les dépôts. Obtenir la version, les dépendances, etc.

Comme vous modifiez la configuration système, ces opérations nécessitent des droits root.

Ainsi, il faut s'identifier en **root** ou utiliser **sudo**.

# VI. Gestionnaire de package Linux

## VI.3. Les commandes apt

### ❖ La commande apt-get update

**apt-get update** permet de **mettre à jour l'indexation** du dépôt sur votre Linux.

En effet, lorsque votre indexation est trop ancien, il n'est plus synchronisé avec celui en ligne.

Ainsi, vous pouvez demander une version qui n'existe plus et obtenir une erreur.



# VI. Gestionnaire de package Linux

## VI.3. Les commandes apt

### ❖ La commande apt-get install

Cette commande permet d'**installer un logiciel** et paquet sur votre Linux. Ici on installe l'éditeur emacs avec la commande :

**apt-get install mon\_package**

Lors de son utilisation la liste des paquets et leurs dépendances apparaît. L'espace disque nécessaire s'affiche. Enfin une confirmation apparaît pour continuer l'installation.

# VI. Gestionnaire de package Linux

## VI.3. Les commandes apt

### ❖ La commande apt-get remove

**apt-get remove** permet de **supprimer un paquet et logiciel**. Il fonctionne donc comme install mais effectue l'opération inverse.

Cette commande ne supprime pas les fichiers de configuration.

Pour tout supprimer, il faut utiliser **apt-get purge**.

Exemple ici avec :

**apt-get remove mon\_package**

# VI. Gestionnaire de package Linux

## VI.3. Les commandes apt

### ❖ La commande apt-get upgrade

Cette commande met à jour les paquets de la distribution Linux.

Cela ne change pas la version de la distribution mais mets à jour des paquets.

En effet, des mises à jour de sécurité sont publiées chaque jour.

On lance la mise à jour des paquets avec la commande suivante :

**apt-get upgrade**

# VI. Gestionnaire de package Linux

## VI.3. Les commandes apt

### ❖ La commande apt-get dist-upgrade

Enfin la commande apt-get dist-upgrade permet la mise à niveau et mise à jour de la distribution.

Pour plus d'informations, reportez-vous au paragraphe plus bas à cet effet.

### ❖ La commande apt-get download

Enfin on peut télécharger un paquet pour l'installer manuellement par exemple.

Par exemple pour télécharger le paquet emacs depuis les dépôts APT.

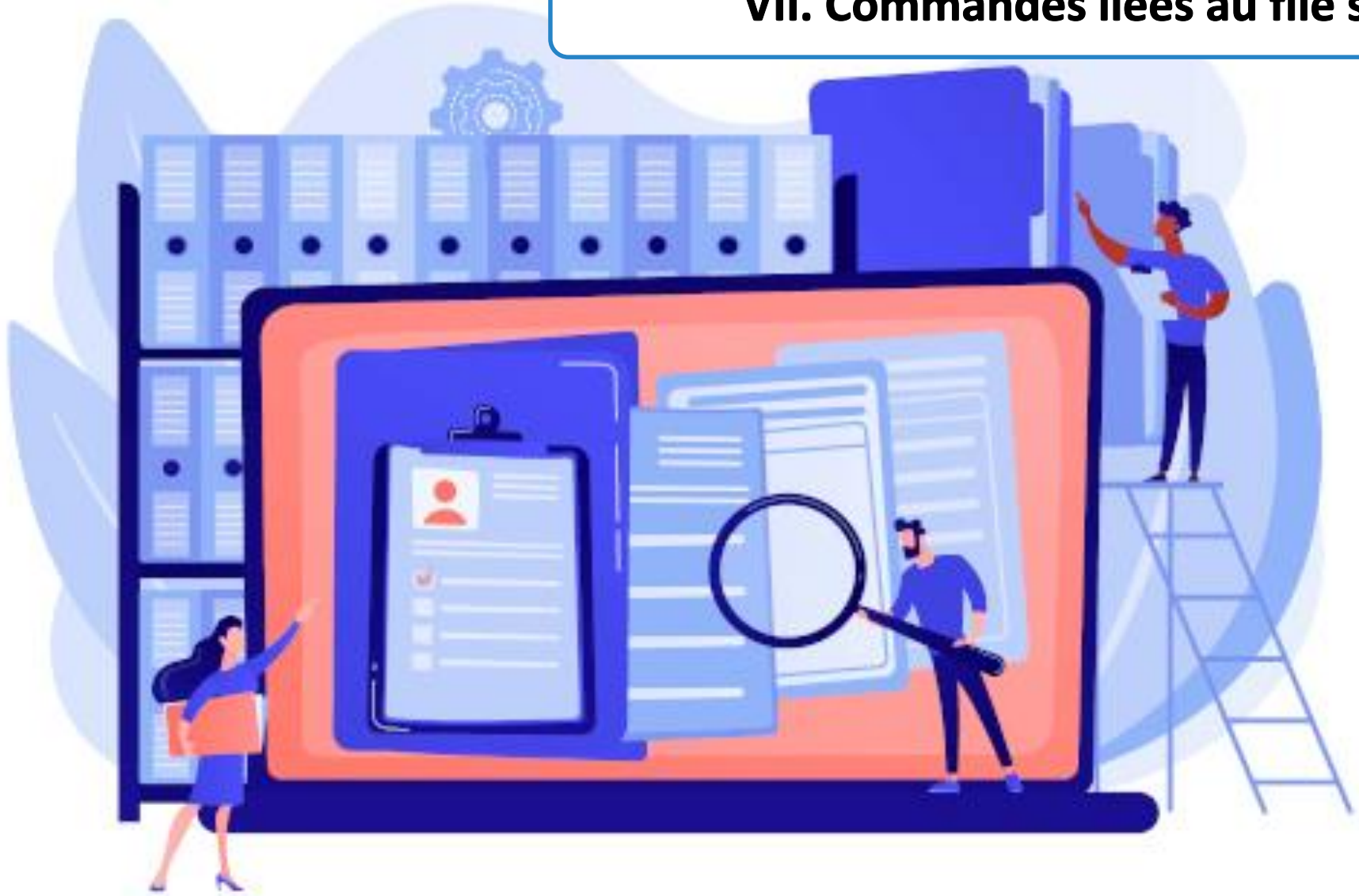
**apt-get download emacs**

puis on peut l'installer avec dpkg :

**dpkg -i <package>**

# Chapitre X : Le terminal Linux

## VII. Commandes liées au file system



# VII. Commandes liées au file system

## 1. La commande cd (Change Directory)

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

### Syntaxe

**cd [répertoire]**

- La commande « **cd** » sans argument, change le répertoire courant par le répertoire de connexion.
- La commande « **cd** » avec argument, remplace le répertoire courant par le répertoire spécifié.

# VII. Commandes liées au file system

## 1. La commande cd (Change Directory)

Le répertoire peut être spécifié avec un chemin absolu ou un chemin relatif.

**Exemple :**

**Chemin absolu**

```
cd /home/wendkuuni/Documents/TD
```

**Chemin relatif**

```
cd ../cours
```

# VII. Commandes liées au file system

## 2. Les commandes « mkdir » et « rmdir »

### 2.1. 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.



# VII. Commandes liées au file system

## 2.1. La commande « mkdir » (make directory)

### ❖ Créer une hiérarchie de répertoires

L'option « -p » (parent) permet de créer un ensemble de sous-répertoire de manière hiérarchique.

**Exemple:** Créer la hiérarchie de répertoires « wendkuuni/OS/cours » , pour cela :

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

```
mkdir wendkuuni
```

```
mkdir wendkuuni/OS
```

```
mkdir wendkuuni/OS/cours
```

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

```
mkdir -p wendkuuni/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.

# VII. Commandes liées au file system

## 2.2. La commande « rmdir » (remove directory)

La commande « **rmdir** » permet de supprimer des répertoires.

**Syntaxe:**

**rmdir [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.

**Attention :** Les répertoires à supprimer **doivent être vides**.

Au cas contraire utiliser l'option **-r** (pour récursif)

# VII. Commandes liées au file system

## 2.2. La commande « rmdir » (remove directory)

❖ Supprimer une hiérarchie de répertoires

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

**rmdir -p wendkuuni/OS/cours**

# VII. Commandes liées au file system

## 3. Les commandes de copies et de déplacements de fichiers

### 3.1. 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**

# VII. Commandes liées au file system

## 3. Les commandes de copies et de déplacements de fichiers

### ❖ Premier cas: un seul fichier source

**cp [option] fich-source fich\_destination**

- Si « fich\_destination » est un répertoire, alors le fichier « fich\_source » sera dupliqué dans le répertoire « fich\_destination ».

**Attention:** si « fich\_destination » existe dans le répertoire destination alors il est écrasé et remplacé par le nouveau fichier « fich\_source ».

- Si « fich\_destination » n'est pas un répertoire, alors la commande « cp » effectue une copie du fichier source « fich\_source » vers le fichier destination « fich\_destination ».

**Attention:** si « fich\_destination » existe alors il est écrasé et remplacé par « fich\_source ».

# VII. Commandes liées au file system

## 3. Les commandes de copies et de déplacements de fichiers

### ❖ 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).

### 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 le fichier source est un répertoire, alors on copie les fichiers et les sous-répertoires.

# VII. Commandes liées au file system

## 3.2. 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**

### ❖ 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 il est écrasé et remplacé par « fich\_source ».

# VII. Commandes liées au file system

## 3.2. 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**

❖ **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 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 (fichier source).



# VII. Commandes liées au file system

## 4. La commande « ls »

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

**Syntaxe :**

**ls [options] [arguments ]**

- 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 s'il existe. Sinon elle affiche une erreur.

# VII. Commandes liées au file system

## 4. La commande « ls »

**Les options les plus utilisées de la commande « ls ».**

- 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 « **-s** » : affiche la taille en Ko de chaque fichier.
- 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 et ne descend pas dans un répertoire.

# VII. Commandes liées au file system

## 5. Effacement d'un fichier ou un répertoire - 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 ordinaire alors, il sera supprimé.
- Si l'argument est un répertoire alors il faut rajouter l'option « **-r** ».
- Si plusieurs arguments sont spécifiés, alors ils seront tous supprimés.

# VII. Commandes liées au file system

## 5. Effacement d'un fichier ou un répertoire - 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) : supprime le fichier, même s'il est protégé (sans tenir compte des protections du fichier), il ne vérifie que le propriétaire. Vous pouvez donc effacer vos fichiers, même s'ils sont protégés.

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

# VII. Commandes liées au file system

## 6. Affichage du contenu d'un fichier

### 6.1. La commandes « cat »

La commande « cat » permet d'afficher, sur la sortie standard, le contenu des fichiers spécifiés en argument de la commande, l'un après l'autre.

#### Syntaxe :

**cat [options] fichier1 [fichier2,..]**

#### Options:

« **-n** » : permet d'afficher, en plus, les numéros de lignes des fichiers.

« **-b** » : identique à « -n » mais les lignes vides ne seront pas numérotées.

#### Exemple:

Afficher sur la sortie standard le contenu du fichier «**/etc/passwd**»

**cat /etc/passwd**

# VII. Commandes liées au file system

## 6. Affichage du contenu d'un fichier

### 6.1. La commandes « cat »

#### Remarque:

La commande « cat » permet de faire la concaténation des fichiers en redirigeant la sortie standard à l'aide des opérateurs de redirections « > » ou « >> ».

**Exemple :** concaténer les fichiers « fiche1 » et « fiche2 » dans « fiche 3 »  
**cat fiche1 fiche2 > fiche3**

# VII. Commandes liées au file system

## 6.2. La commande more

**Syntaxe :**

**more fichiers**

Affiche sur la sortie standard, le contenu des fichiers, spécifiés en argument de la commande, l'un après l'autre et page écran par page écran.

- Pour visualiser la page suivante, on tape sur la touche « Space ».
- Pour visualiser la ligne suivante, on tape sur la touche « Entrée » (ou touche « return »)
- Pour terminer la visualisation, on tape sur la touche « q » ou «Q».

# VII. Commandes liées au file system

## 6.3. les commandes « head » et « tail ».

### Commande « head »

Permet d'afficher sur la sortie standard les débuts des fichiers passés en argument de la commande, l'un après l'autre.

#### Syntaxe:

**head [-q] [-c nbcars] [-n nblignes] [fiche,...]**

- L'option « -q » n'affiche pas le nom du fichier.
- L'option « -v » affiche le nom du fichier avant d'en afficher l'entête.
- Par défaut, la commande affiche les 10 premières lignes de chaque fichier passé en argument.



# VII. Commandes liées au file system

## 6.3. les commandes « head » et « tail ».

### ❖ Commande « head »

#### Exemple:

**head -n 4 fiche** ou **head -4 fiche**

Affiche les 4 premières lignes du fichier « fiche »

L'option « -c » : elle peut s'écrire :

**head -c nb fiche** ou **head -cnb fiche**

Permet d'afficher les « nb » premiers caractères du fichier « fiche ».

#### Exemple :

**head -c 12 fiche** ou **head -c12 fiche**

Affiche les 12 premiers caractères du fichier « fiche ».

# VII. Commandes liées au file system

## 6.3. les commandes « head » et « tail ».

### ❖ Commande « tail »

Elle permet d'afficher sur la sortie standard les dernières lignes des fichiers passés en argument de la commande fichier.

#### **Syntaxe:**

**tail [-c nbcars] [-n +nb/-nb] [fiche,...]**

- Par défaut, la commande affiche les 10 dernières lignes de chaque fichier passé en argument.

#### **Exemple:**

**tail fiche1 fiche2**

Affiche les 10 dernières lignes du fichiers « fiche1 » ensuite affiche les 10 dernières lignes du fichiers « fiche2 ».

# VII. Commandes liées au file system

## 6.3. les commandes « head » et « tail ».

### ❖ Commande « tail »

- L'option « -n -nb »: elle peut s'écrire :

**tail -n -nb fiche** ou **tail -n nb fiche** ou **tail -nb fiche**

Affiche les « nb » dernières lignes du fichier « fiche »

**Exemple:**

**tail -n -4 fiche** ou **tail -n 4 fiche** ou **tail -4 fiche**

Affiche les 4 dernières lignes du fichier « fiche »

- L'option « -n +nb » : elle peut s'écrire:

**tail -n +nb fiche**

Affiche toutes les lignes du fichier à partir de la ligne « nb »

**Exemple:**

**tail -n +4 fiche**

Affiche le contenu du fichier « fiche » à partir de la 4ème ligne.

# VII. Commandes liées au file system

## 7. Les métacaractères du shell

Les métacaractères sont utilisés pour regrouper (générer) les noms des fichiers qui ont des parties communes (qui ont certaines similitudes).

### ❖ Le métacaractère **\*** (jocker, wildcard) :

Remplace n'importe quelle chaîne de caractères dans le nom d'un fichier, sauf le caractère «.» en première position (fichier cachés).

Par exemple l'expression «**test\***» désigne tous les noms de fichiers, relativement à un répertoire, qui commencent par le mot « test ».

# VII. Commandes liées au file system

## 7. Les métacaractères du shell

❖ Le métacaractère **\*** (jocker, wildcard) :

### Exemples:

- Lister tous les fichiers, du répertoire courant, qui commence avec le caractère « c ».

**ls -d c\***

- Afficher le contenu de tous les fichiers qui terminent avec « .c » et qui sont localisés dans le répertoire « /home/daoudi/cour »

**cat /home/daoudi/cours/\*.c**

- Lister tous les fichiers dont les noms contiennent « . »

**ls -d \*.\***

**Attention :** « ls \*.\* » va lister tous les fichiers dont les noms contiennent « . » sauf les fichiers

# VII. Commandes liées au file system

## 7. Les métacaractères du shell

### ❖ Le métacaractère « ? » :

Remplace (masque) n'importe quel caractère sauf le point (« . ») en première position.

#### Exemples:

- Lister tous les fichiers et les répertoires qui commencent par la chaîne « uvbf » suivie d'un caractère quelconque et se terminent par « .txt »

**ls uvbf?.txt**

- Lister tous les fichiers dont le deuxième caractère est la lettre « b ».

**ls -d ?b\***

# VII. Commandes liées au file system

## 7. Les métacaractères du shell

### ❖ Les métacaractères les crochets [ ]:

désigne un seul caractère parmi les caractères qui sont donnés entre crochets (« [ ] »).

**N.B :** Les caractères sont accolés ou séparés par « , ». Pas d'espaces après le crochet ouvrant et avant le crochet fermant et entre les caractères.

### Exemple :

- Lister tous les noms des fichiers et répertoires, du répertoire courant, qui commencent avec la chaîne « test. » suivie par la lettre « c » ou par la lettre « j », et qui terminent par n'importe quelles autres chaînes de caractères.

**ls -d test.[cj]\***

ou

**ls -d test.[c,j]\***

# VII. Commandes liées au file system

## 8. Recherches

Permet de chercher dans un répertoire (repère de base) et dans toute l'arborescence depuis ce répertoire (on explore récursivement le répertoire et ces sous répertoires), les fichiers vérifiant certains critères.

### Syntaxe:

**find repertoires critères options**

- Si le critère n'est pas spécifié, alors la commande « **find** » affiche récursivement tous les répertoires et fichiers du répertoire précisé dans la commande.
- L'option « **-print** ». Elle est implicite sur la plupart des systèmes Unix. Elle permet d'afficher les résultats de la recherche sur la sortie standard (écran).
- L'option « **-name** » : fait une recherche sur le nom de fichier. On peut utiliser les métacaractères.



# VII. Commandes liées au file system

## 8. Recherches

### Exemples

- Recherche de tous les fichiers nommés « **test.c** » à partir du répertoire « home ».

**find /home -name test.c -print**

### Remarque:

- Si le répertoire est précisé avec un chemin relatif, les résultats sont aussi affichés avec un chemin relatif.
- Si le répertoire est précisé avec un chemin absolu, les résultats sont aussi affichés avec un chemin absolu.

# VII. Commandes liées au file system

## 9. La commande « sort »

La commande « sort » permet

- de trier les lignes de caractères saisies sur la ligne de commande.
- de trier les lignes des fichiers passés en argument à la commande « sort ».
- de faire le tri sur un champ particulier.

# VII. Commandes liées au file system

## 9. La commande « sort »

### Par défaut :

- La sortie (le résultat de la commande) est dirigée sur la sortie standard.
- Le tri s'effectue par ordre alphanumérique (selon l'ordre ASCII standard).

### 9.1. Trie des lignes saisies sur la ligne de commandes

La commande « sort » : lit les lignes saisies sur la ligne de commande, trie ces lignes par ordre alphanumérique, ensuite, affiche sur l'écran les lignes triées.

# VII. Commandes liées au file system

## 9.2. Trie des lignes à partir d'un fichier

**Syntaxe :**

**sort [option] fiche**

La commande « sort » trie les lignes du fichier « fiche » et affiche le résultat sur la sortie standard.

- L'option « -o » permet de spécifier le fichier dans lequel sera écrit le résultat du tri.
- Par défaut, le tri s'effectue selon l'ordre ASCII standard. Les options suivantes permettent de choisir un autre ordre pour le tri.
  - L'option « -n » : spécifie l'ordre numérique.
  - L'option « -d » : le tri est effectué selon l'ordre défini dans un dictionnaire.
  - L'option « -f » : on ignore majuscules et minuscules.
  - L'option « -r » : renverse l'ordre

## VII. Commandes liées au file system

### 10. La commande grep (Global Regular Expression Printer)

La commande « grep » cherche, dans un fichier passé en argument, les lignes qui vérifient le critère de recherche (les lignes qui contiennent l'expression régulière précisée sur la ligne de commande), et affiche les lignes trouvées sur la sortie standard (écran).

**Syntaxe :**

**grep [options] expression-régulière [liste-de-fichiers]**

**Remarque:** Pour éviter toute interprétation du shell des caractères spéciaux, il vaut mieux mettre l'expression régulière entre simple quotes « ' » ou entre doubles quotes « "" ».

**Exemple :** Recherche la chaîne « étudiant » dans le fichier « /etc/passwd ».

```
grep "etudiant" /etc/passwd
```

# VII. Commandes liées au file system

## 10. La commande grep (Global Regular Expression Printer)

### Les principales options de grep:

- L'option « -c » : affiche le nombre de lignes trouvées,
- L'option « -i » : ignore les majuscules/minuscules,
- L'option « -l » : liste les fichiers, donnés en argument (en entrée), dans lesquels au moins une ligne a été trouvée,
- L'option « -n » : affiche les lignes trouvées ainsi que leurs numéros.
- L'option « -v » : affiche les lignes qui ne contiennent pas l'expression de recherche.

# VII. Commandes liées au file system

## 11. La commande wc

Syntaxe :

**wc [option] fiche**

La commande "wc" compte et affiche sur la sortie standard:

- Le nombre de lignes du fichier « fiche ». Les lignes sont séparées par un retour chariot (<CR>), lorsqu'on tape sur la touche « Entrée » ou « Return ».
- Le nombre de mots. Les mots sont séparés par des espaces (la barre space) ou par des tabulations (touche tab).
- Le nombre de caractères.

**Remarque :** Si plusieurs fichiers sont passés sur la ligne de commande alors le comptage se fait par fichier.

# VII. Commandes liées au file system

## 11. La commande wc

Syntaxe :

**wc [option] fiche**

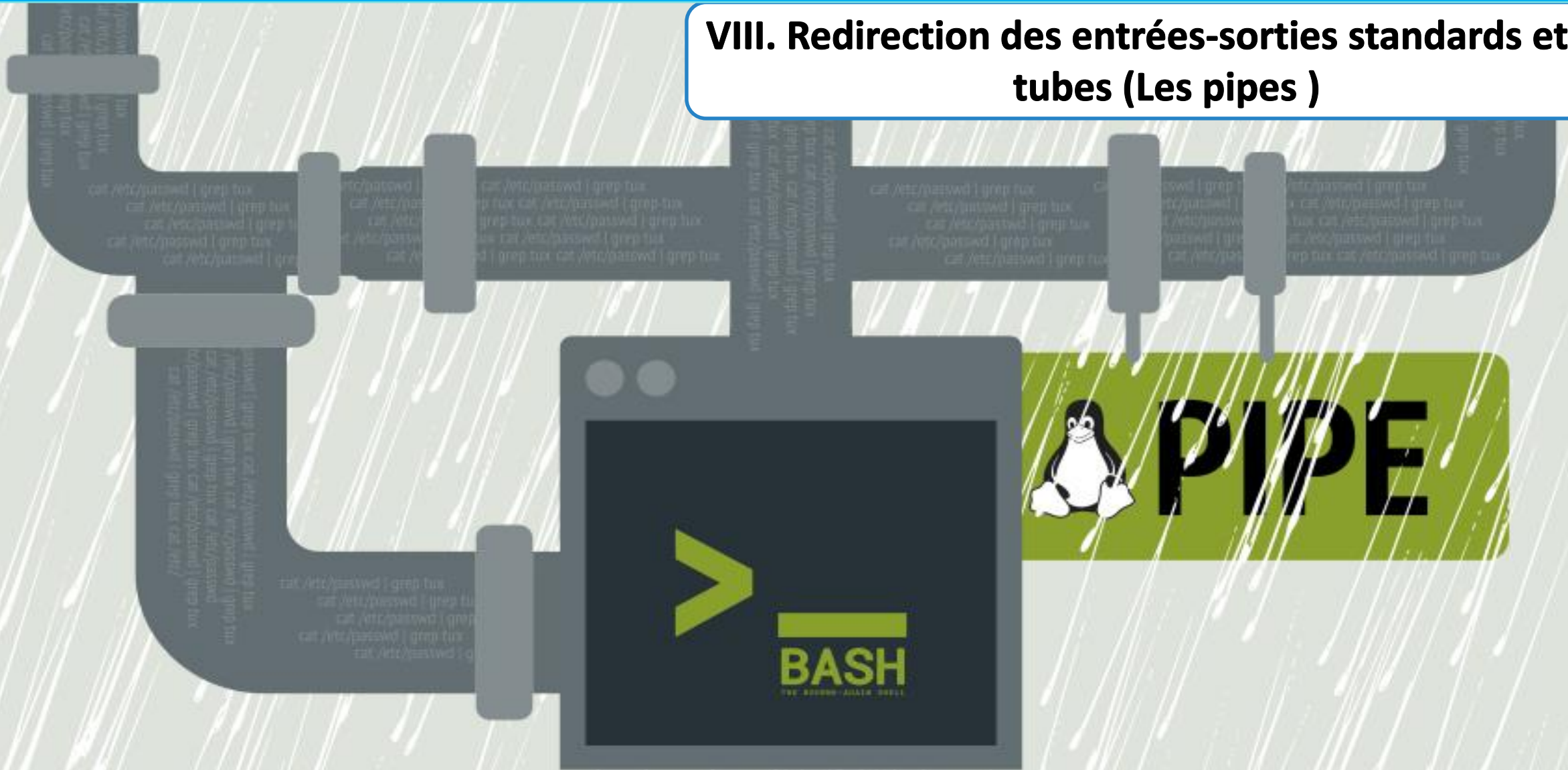
**Les options de la commande « wc » sont :**

- Option « -l » : compte seulement le nombre de lignes,
- Option « -w » compte seulement le nombre de mots,
- Option « -c » compte seulement le nombre de caractères.



# Chapitre X : Le terminal Linux

## VIII. Redirection des entrées-sorties standards et les tubes (Les pipes )



# VIII. Redirection des entrées-sorties standards et les tubes (Les pipes )

## ❖ Redirection des entrées-sorties standards

Lors de l'exécution d'une commande, un processus est créé.

Par défaut le processus: lit ses données d'entrée à partir de l'entrée standard (par défaut le clavier) et écrit les résultats et les erreurs dans la sortie standard (par défaut l'écran).

Ces canaux de communications (**canaux d'entrée :sortie**) sont désignés par les fichiers :

- « **stdin** » : pour l'entrée standard, identifiée par l'entier **0**.
- « **stdout** » : pour la sortie standard, identifiée par l'entier **1**.
- « **stderr** » : pour la sortie d'erreur standard, identifiée par l'entier **2**.

Il est possible de rediriger les entrées/soties standards des commandes afin que la lecture se fasse à partir d'un fichier et que l'écriture se fasse dans fichier.

# VIII. Redirection des entrées-sorties standards et les tubes (Les pipes )

## ❖ Redirection des entrées-sorties standards

### 1. Redirection de l'entrée standard

On peut rediriger l'entrée standard d'une commande afin que la lecture se fasse à partir d'un fichier grâce au symbole « < ».

#### Syntaxe :

**commande < fichier\_entree**

La commande effectue ses entrées à partir du fichier « fichier\_entree »

#### Exemple

**cat < donnees\_entree**

La commande lit le fichier « donnees\_entree » et l'affiche sur la sortie standard, au lieu de lire les données au clavier.

# VIII. Redirection des entrées-sorties standards et les tubes (Les pipes )

## ❖ Redirection des entrées-sorties standards

### 2. Redirection de la sortie standard

On peut rediriger la sortie standard d'une commande afin que l'écriture se fasse dans un fichier grâce au symbole « > » ou « >> ».

**Syntaxe :**

```
commande > fichier_sortie  
commande >> fichier_sortie
```

- Si le fichier « fichier\_sortie » n'existe pas, alors il sera créé.
- Si le fichier « fichier\_sortie » existe, alors:
  - Si le symbole « > » est utilisé alors le fichier « fichier\_sortie » sera écrasé et remplacé par la nouvelle sortie standard de la commande.
  - Si le symbole « >> » est utilisé, alors la nouvelle sortie standard sera rajoutée à la fin du fichier « fichier\_sortie ».

# VIII. Redirection des entrées-sorties standards et les tubes (Les pipes )

## ❖ Redirection des entrées-sorties standards

### 2. Redirection de la sortie standard

#### Exemple:

- On trie le fichier « fiche1 » et on écrit le résultat dans le fichier « fiche\_trie » au lieu d'afficher le résultat sur l'écran.

**sort fiche1 > fiche\_trie**

- On trie le fichier « fiche2 » et on écrit le résultat à la fin du fichier « fiche\_trie ».

**sort fiche2 >> fiche\_trie**

- Dans l'exemple suivant on montre qu'on peut rediriger l'entrée et la sortie en même temps.

**sort < fiche > fichier\_trie**

# VIII. Redirection des entrées-sorties standards et les tubes (Les pipes )

## ❖ Les tubes (Les pipes )

Les pipes (tubes) permettent de rediriger la sortie d'un processus vers l'entrée d'un autre. Les processus s'exécutent en parallèles.

### Syntaxe :

#### **cmdA | cmdB**

- Le symbole « | » appelé « pipe ». IL permet de relier deux commandes entre elles.
- La sortie standard de la commande « cmdA » est utilisée comme entrée standard de la commande « cmdB ».

# VIII. Redirection des entrées-sorties standards et les tubes (Les pipes )

## ❖ Les tubes (Les pipes )

### Exemple

- Exécuter la commande « `ls -l` » et rediriger la sortie de la commande vers le fichiers « `resultat.txt` »

**`ls -l > resultat.txt`**

- Compter le nombre de lignes, le nombre de mots et le nombres de caractères du fichier « `resultat.tx` ». Et rediriger le résultat vers le fichier « `cp.txt` »

**`wc < resultat.txt > cp.txt` ou `wc resultat.txt > cp.txt`**

- Les deux commandes peuvent être combinées pour élaborer une seule:

**`ls -l | wc > cp.txt`**

# Chapitre X : Le terminal Linux

## IX. Contrôle des processus





# IX. Contrôle des processus

## 1. Commande « ps »

La commande ps (Process Status) permet de lister les processus qui sont en cours même sur la machine.

**Syntaxe:**

**ps [options]**

La commande « ps » sans options, n'affiche que les processus en cours qui sont lancés par l'utilisateur sur le terminal actuel (par le processus shell).

# IX. Contrôle des processus

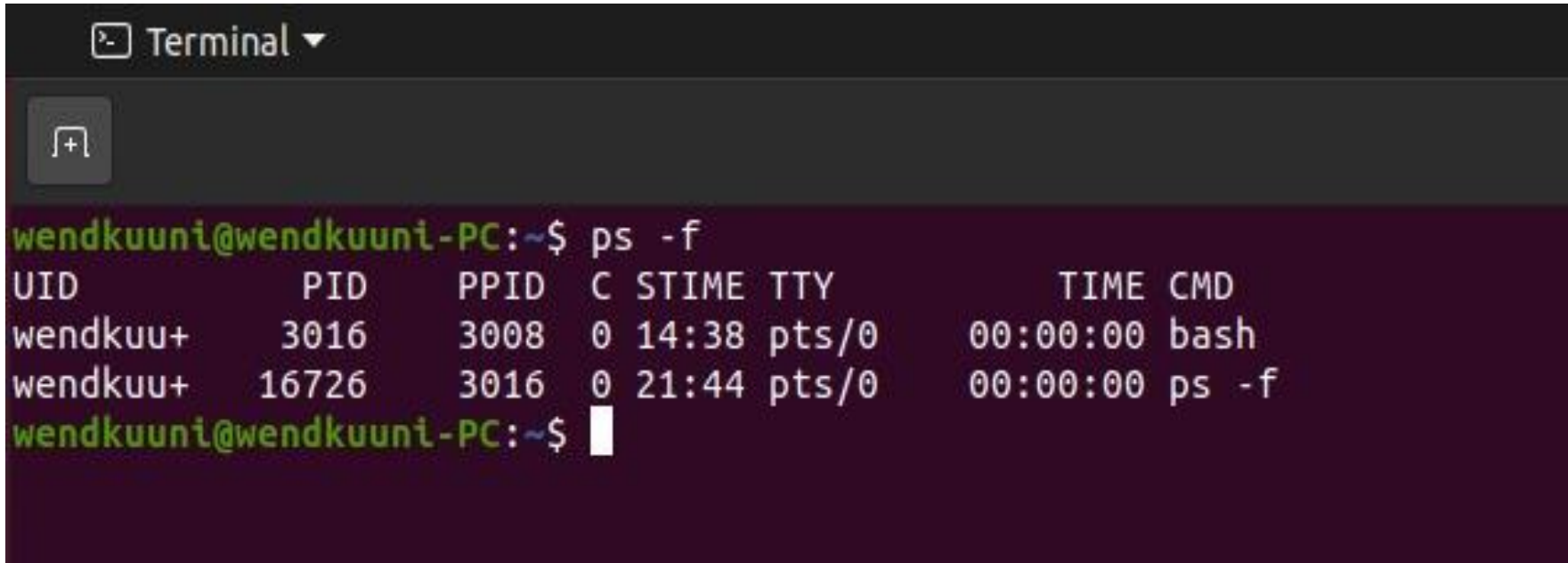
## 1. Commande « ps »

### Quelques options:

Pour plus d'option utiliser la commande « **man** » ou exécuter la commande « **ps --help** ».

- L'option « **-f** »

**ps -f**



```
Terminal ▾  
wendkuuni@wendkuuni-PC:~$ ps -f  
UID          PID     PPID  C  STIME TTY          TIME CMD  
wendkuu+    3016     3008  0  14:38 pts/0        00:00:00 bash  
wendkuu+   16726     3016  0  21:44 pts/0        00:00:00 ps -f  
wendkuuni@wendkuuni-PC:~$
```

# IX. Contrôle des processus

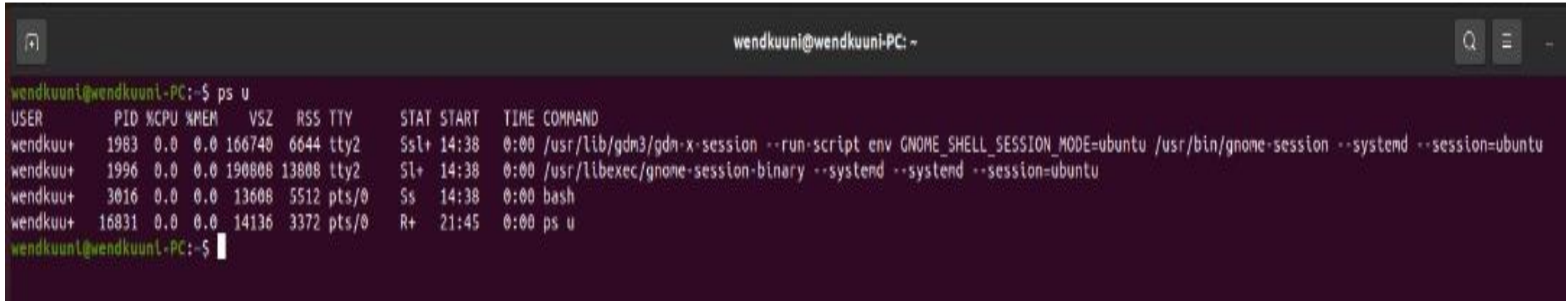
## 1. Commande « ps »

### Quelques options:

Pour plus d'option utiliser la commande « **man** » ou exécuter la commande « **ps --help** ».

- L'option « **u** »

**ps u**



```
wendkuuni@wendkuuni-PC: ~  
wendkuuni@wendkuuni-PC:~$ ps u  
USER      PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND  
wendkuu+  1983  0.0  0.0 166740 6644 tty2    Ssl+ 14:38   0:00 /usr/lib/gdm3/gdm-x-session --run-script env GNOME_SHELL_SESSION_MODE=ubuntu /usr/bin/gnome-session --systemd --session=ubuntu  
wendkuu+  1996  0.0  0.0 190808 13808 tty2    Sl+  14:38   0:00 /usr/libexec/gnome-session-binary --systemd --systemd --session=ubuntu  
wendkuu+  3016  0.0  0.0  13608  5512 pts/0    Ss   14:38   0:00 bash  
wendkuu+  16831 0.0  0.0  14136  3372 pts/0    R+   21:45   0:00 ps u  
wendkuuni@wendkuuni-PC:~$
```

# IX. Contrôle des processus

## 1. Commande « ps »

ps u

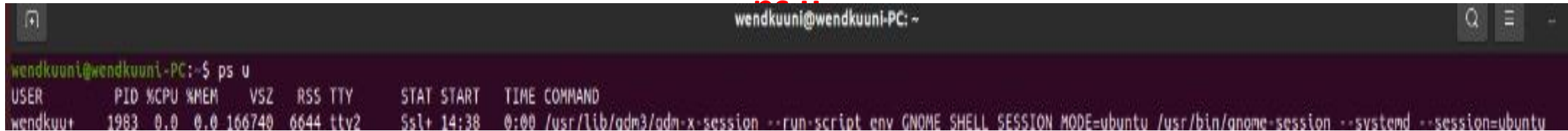
```
wendkuuni@wendkuuni-PC: ~  
wendkuuni@wendkuuni-PC:~$ ps u  
USER      PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND  
wendkuuni 1983  0.0  0.0 166740 6644 tty2    Ssl+ 14:38   0:00 /usr/lib/adm3/adm-x-session --run-script env GNOME SHELL SESSION MODE=ubuntu /usr/bin/gnome-session --systemd --session=ubuntu
```

### Signification des différentes colonnes

- **UID (User ID)** : identifiant de l'utilisateur qui a lancé le processus.
- **PID (Process ID)** : correspond au numéro du processus
- **STIME (Start TIME)** : correspond à l'heure du lancement du processus.
- **START** : idem STIME.
- **TTY** : correspond au nom du terminal depuis lequel le processus a été lancé.
- **TIME** : correspond à la durée de traitement du processus.
- **CMD** : correspond au nom du processus (la Commande exécutée).
- **%CPU** : facteur d'utilisation du cpu exprimé en %.
- **%MEM** : rapport d'utilisation de la mémoire
- **RSS** : mémoire physique (non virtuelle) utilisée par la processus, exprimée en kiloOctets .
- **VSZ** : mémoire virtuelle.
- **STAT** : état du processus

# IX. Contrôle des processus

## 1. Commande « ps »



```
wendkuuni@wendkuuni-PC: ~  
wendkuuni@wendkuuni-PC:~$ ps u  
USER      PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND  
wendkuu+  1983  0.0  0.0 166740 6644 tty2    Ssl+  14:38   0:00 /usr/lib/adm3/adm-x-session --run-script env GNOME SHELL SESSION MODE=ubuntu /usr/bin/gnome-session --systemd --session=ubuntu
```

### Signification des différentes colonnes

- **STAT** : état du processus

### Quelques différents états:

- **R (Running)**: prêt à être exécuté (il sur la file d'attente des processus).
- **S (Sleeping)**: bloqué en attente d'une action externe (par exemple le processus attend une lecture sur la clavier).
- **T** : processus arrêté, par exemple par Ctrl-Z.

# IX. Contrôle des processus

## 2. Arrêt d'un processus / signaux

Pour arrêter un processus qui s'exécute en arrière-plan, on utilise la commande « kill » qui consiste à envoyer des signaux au processus. Le signal est un moyen de communication entre les processus.

**Syntaxe :**

**kill [-num\_signal] PID [PID ...]**

Le principe de la commande consiste à envoyer un signal (une requête) au processus spécifié dans la commande par son PID.

Ce dernier intercepte le message et réagit en fonction du signal envoyé.

# IX. Contrôle des processus

## 2. Arrêt d'un processus / signaux

### Remarque:

- Seulement le propriétaire du processus ou le super-user (utilisateur "root").
- Chaque signal a un nom et un numéro qui dépendent du système utilisé.

La commande « **kill -l** » permet **d'obtenir la liste des signaux**.

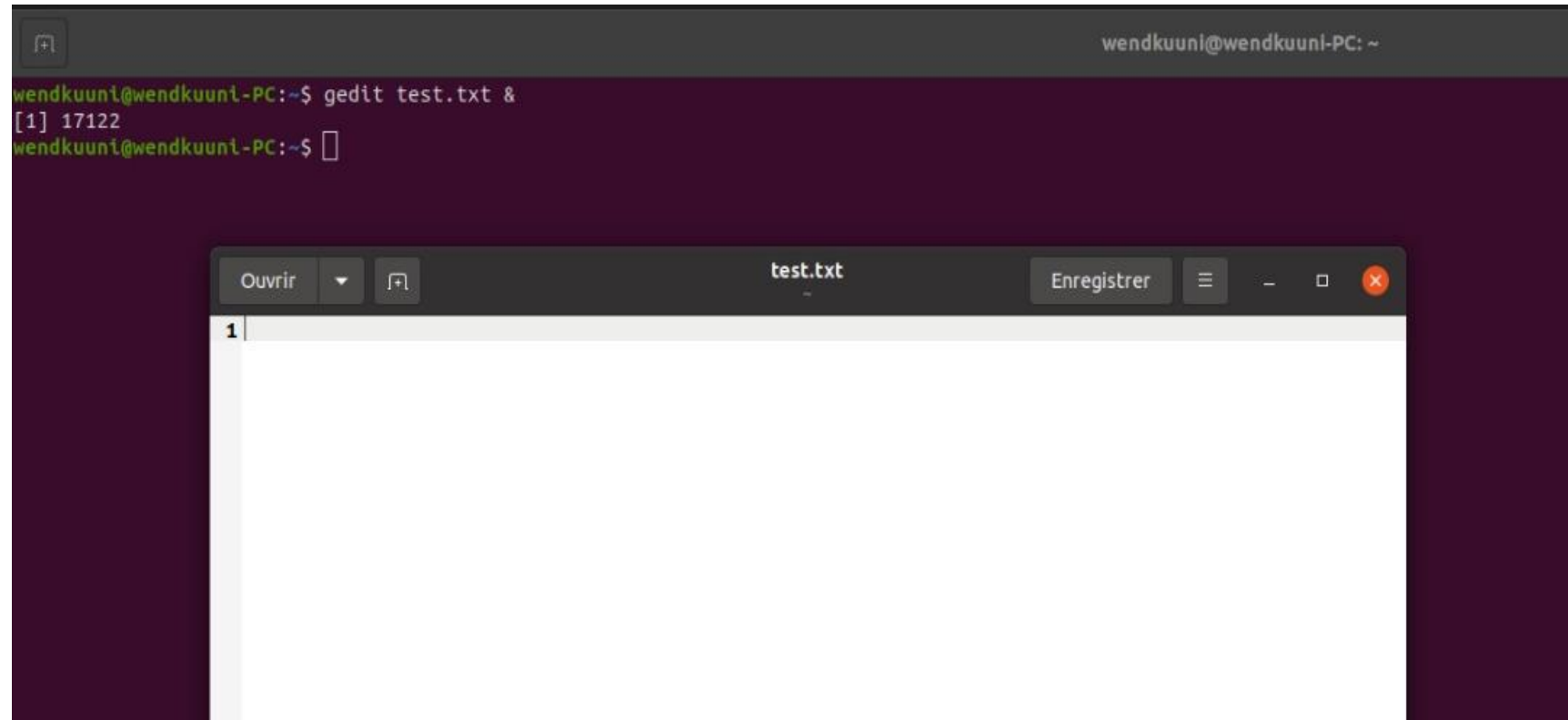
Par défaut, la commande « **kill** » envoie le signal demandant au processus spécifié de se terminer normalement.

# IX. Contrôle des processus

## 2. Arrêt d'un processus / signaux

### Exemple

**gedit test.txt &**



The screenshot shows a terminal window with a dark purple background and a gedit editor window. The terminal window has a title bar that reads "wendkuuni@wendkuuni-PC: ~". The command prompt shows the user has executed "gedit test.txt &" and the output is "[1] 17122". The gedit editor window has a title bar that reads "test.txt" and contains a single line of text "1".

```
wendkuuni@wendkuuni-PC:~$ gedit test.txt &  
[1] 17122  
wendkuuni@wendkuuni-PC:~$
```



# IX. Contrôle des processus

## 2. Arrêt d'un processus / signaux

### Exemple

**kill 17122**

A screenshot of a terminal window titled "Terminal". The terminal has a dark background with a lighter gray header bar. In the header bar, there is a window icon and a plus sign icon. The main area of the terminal shows a shell prompt "wendkuuni@wendkuuni-PC:~\$" in green. Above the prompt, the text "[1]+ Complété" is displayed in white. To the right of the prompt, the command "gedit test.txt" is shown in white. A white cursor is positioned at the end of the prompt.

```
[1]+  Complété      gedit test.txt
wendkuuni@wendkuuni-PC:~$
```

# IX. Contrôle des processus

## 2. Arrêt d'un processus / signaux

### Quelques signaux

- **Le signal numéro 1 (SIGHUP)** : Ce signal est envoyé par le père à tous ses enfants lorsqu'il se termine.
- **Le signal numéro 2 (SIGINT)** : ce signal permet l'interruption du processus demandé (Ctrl+C).
- **Le signal 9 (SIGKILL)** : ce signal permet de forcer l'arrêt du processus.

# IX. Contrôle des processus

## 2. Arrêt d'un processus / signaux

### Quelques signaux

- **Le signal 9 (SIGKILL)** : ce signal permet de forcer l'arrêt du processus.

Exemple:

**gedit test.txt &**

[1] 1234

**kill -9 1234**

[1] + Processus arrêté gedit test.c

# Chapitre X : Le terminal Linux

## X. Composition des commandes



# X. Composition des commandes

## Opérateurs « && »

**Syntaxe:**

**cmd1 && cmd2**

Exécute la commande «cmd1».

Puis, si l'exécution de la commande « cmd1 » s'est bien passée, c'est-à-dire la commande «cmd1» a un code retour (un statut de sortie) égal à 0 alors on exécute la commande «cmd2».

# X. Composition des commandes

## Opérateurs « && »

### Syntaxe:

**cmd1 && cmd2**

### Exemple:

**ls -l > fiche && wc test.txt**

11 17 118 test.txt

**ls -j > fiche && wc test.c**

ls: option invalide -'j'

Dans ce cas la commande « wc » n'a pas été exécutée.

## XI. Les partitions de disque sur Linux



# XI. Les partitions de disque sur Linux

Pour rappel, le partitionnement de disque consiste à découper le disque et formater ces derniers afin de pouvoir héberger les fichiers.

En rappel, sur Linux, le système de fichiers par défaut est **ext4**.

**Une fois les partitions de disque créées, il faut bien entendu être capable de les identifier.**

**Linux a pour cela sa propre nomenclature de partitions de disque totalement différentes de Windows.**

Les types de partitions de disques **primaires, étendues, EFI**



# XI. Les partitions de disque sur Linux

Linux permet de créer deux types de partitions :

- ❑ **Partitions primaires ou principales** : C'est le type de partition le plus utilisé et qui permet de stocker des données. en **MBR**, on peut en créer que 4 maximum par disque dur.
- ❑ **Partition étendue** : On utilise ce type de partition pour pouvoir dépasser la limite des 4 partitions principales.

Puis la partition de démarrage des disques **GPT (GUID Partition Table)** :

- ❑ **Partitions EFI** : c'est la partition de démarrage pour les disques **GPT sur les PC UEFI**. Elle est formate en FAT32 et stocke le firmware EFI.

# XI. Les partitions de disque sur Linux

Sur n'importe quelle distribution Linux comme Ubuntu ou Mint, les partitions de disque ne sont pas numérotées comme sur Windows. En effet, pour identifier un disque, Linux utilise des lettres à la place des chiffres. On obtient alors la nomenclature suivante :

- ☐ **/dev/sda = Disque 0**
- ☐ **/dev/sdb = Disque 1**
- ☐ **/dev/sdc = Disque 2**

On trouve la même logique pour les espaces de stockage de type **NVME (Nonvolatile Memory Express)**:

- ☐ **/dev/nvme0n1**
- ☐ **/dev/nvme0n2**

# XI. Les partitions de disque sur Linux

En fait selon le type de périphériques de stockages, le nom change. En effet, Linux crée et monte les périphériques dans **/dev**.

Voici la liste des principaux types de périphériques de stockage.

Type de périphériques de stockage	Nom dev dans Linux
Disque SATA ou SCSI	<b>/dev/sda</b>
Espaces de noms de stockage NVME	<b>/dev/nvme0n1</b>
Disque IDE	<b>/dev/hda</b>
Périphérique disquette	<b>/dev/fd0</b>
DVD-Rom ou CD-Rom	<b>/dev/scd0, /dev/sr0 ou /dev/cdrom</b>
Disque XT	<b>/dev/xda</b>
SD Card et carte Flash	<b>dev/mmcblkX</b>

# XI. Les partitions de disque sur Linux

**NVMe (Nonvolatile Memory Express)** est un nouveau protocole d'accès au stockage et de transport conçu pour les disques Flash et SSD nouvelle génération.

Il offre le débit le plus élevé et les délais de réponse les plus courts pour tous les types de workloads d'entreprise.

# XI. Les partitions de disque sur Linux

Au sein d'un disque, vous pouvez avoir plusieurs partitions de disques.

Elles sont alors indiquées par leurs numéros.

Ainsi pour le disque 0 en **SATA** soit donc /dev/sda, on obtient les numéros de partition suivantes : **/dev/sda1, /dev/sda2.**

Pour **un disque IDE**, ce sera /dev/hda1, /dev/hdb2, etc

En **NVME**, les numéros de partitions sont identifiés par p1, p2, ce qui donne pour le disque /dev/nvme0n1 : /dev/nvme0n1p1, /dev/nvme0n1p2

# XI. Les partitions de disque sur Linux

## XI.1. Les utilitaires de partitionnement de disque sur Linux

Linux embarque beaucoup d'utilitaires graphiques ou depuis un terminal pour gérer vos disque et partitions de disque.

Par exemple Ubuntu propose d'utiliser **gnome-disk**, un utilitaire graphique qui vous permet de visualiser les disque et partitions de disque.

Ce dernier donne aussi la possibilité de redimensionner vos partitions de disque.

# XI. Les partitions de disque sur Linux

## XI.1. Les utilitaires de partitionnement de disque sur Linux

Du côté de la ligne de commandes pour gérer les partitions, vous avez les utilitaires suivants :

Les utilitaires de disque en graphique :

- ☐ **L'outil de disque** d'Ubuntu

- ☐ **gparted** permet aussi de modifier les partitions de disques

# XI. Les partitions de disque sur Linux

## XI.1. Les utilitaires de partitionnement de disque sur Linux

Du côté de la ligne de commandes pour gérer les partitions, vous avez les utilitaires suivants :

Les utilitaires de disque en ligne de commandes :

- ☐ **fdisk** pour manipuler les partitions de disque : créer, supprimer ou redimensionner des partitions de disque. fdisk est plutôt destiné aux disques de type MBR
- ☐ **cfdisk** : Il est plus simple à prendre en main que fdisk
- ☐ **gdisk** comme pour fdisk mais à destination des disques GPT
- ☐ **parted** : un autre utilitaire en ligne de commandes pour partitionner ses disques
- ☐ **mkfs** permet de formater une partition de disque
- ☐ **tune2fs** modifier les paramètres du système de fichiers



# XI. Les partitions de disque sur Linux

## XI.2. Les commandes fdisk

Une fois que vous avez trouver **le nom du périphérique du disque**, on l'indique dans la **commande fdisk** pour travailler avec.

**fdisk /dev/sdb**

Les commandes et actions se font en saisissant une lettre. Pour afficher la liste des lettre et commandes, **appuyez sur m** et validez.

Vous établissez vos modifications et vous les **appliquez avec w**, ce qui va **écrire sur le nouveau partitionnement sur le disque**.

Il faut donc bien valider que tout est correct avant de valider les modifications car après c'est trop tard.

# XI. Les partitions de disque sur Linux

## XI.2. Les commandes fdisk

Voici la liste des commandes de fdisk les plus utiles :

Lettre	Description
<b>d</b>	<b>supprimer la partition</b>
<b>F</b>	<b>afficher l'espace libre non partitionné</b>
<b>l</b>	<b>afficher les types de partitions connues</b>
<b>n</b>	<b>ajouter une nouvelle partition</b>
<b>p</b>	<b>afficher la table de partitions</b>
<b>t</b>	<b>modifier le type d'une partition</b>
<b>v</b>	<b>vérifier la table de partitions</b>
<b>i</b>	<b>Afficher des renseignements sur la partition</b>
<b>w</b>	<b>écrire la table sur le disque et quitter</b>
<b>q</b>	<b>quitter sans enregistrer les modifications</b>

# XI. Les partitions de disque sur Linux

## XI.2. Les commandes fdisk

### ❖ Créer une partition de disque

Dans notre exemple, on souhaite ajouter une partition de disque sur le disque **/dev/sdb**.

Donc on utilise la commande **fdisk** de cette manière :

**fdisk /dev/sdb**

1) Ensuite on affiche les partitions avec **la commande p** (pour print). Elle vous indique les numéros de partitions avec **les secteurs de débuts et fin**.

2) Puis on utilise **la commande n** pour **ajouter une nouvelle partition**

fdisk nous demande ensuite le type de partition :

p pour primaire

e pour étendue

3) Puis fdisk demande d'indiquer le premier secteur de la partition. Par défaut, il propose le premier libre

# XI. Les partitions de disque sur Linux

## XI.2. Les commandes fdisk

### ❖ Créer une partition de disque

4) Une fois validé, il faut indiquer le dernier secteur, ce qui demande un calcul sur la taille que l'on souhaite allouer. Heureusement, on peut indiquer directement la taille de l'espace disque avec +XXX en Ko, Mo, Go ou To. Par exemple pour créer une partition de disque de 150 Go. Il faut saisir +150Go.

5) Quand cela est terminé, on réutilise la **commande p** pour valider que les modifications apportées aux disques sont bien les bonnes

6) Quand cela est terminé, on réutilise la commande p pour valider que les modifications apportées aux disques sont bien les bonnes

7) Puis on peut vérifier que la partition de disque est bien créé : **sudo fdisk -l /dev/sdb**

A noter que la partition de disque n'est pas formatée, ainsi si vous désirez la formater en ext4 :  
**mkfs.ext4 /dev/sdb1**

# XI. Les partitions de disque sur Linux

## XI.2. Les commandes fdisk

### ❖ Créer une partition de disque

Enfin vous pouvez monter la partition de disque nouvellement créé :

**mount /dev/sdb /mnt/monsuperpointdemontage**

# XI. Les partitions de disque sur Linux

## XI.2. Les commandes fdisk

### ❖ Supprimer une partition de disque

Pour pouvoir supprimer une partition de disque, il faut qu'aucune partition ne soit montée.

Vous pouvez utiliser la commande **df** ou **mount** pour lister les points de montage.

Ensuite pour démonter une partition, on utilise **umount** soit avec le nom du périphérique soit avec le point de montage.

**umount /dev/sdb2**

ou

**umount /media/sdb2**

# XI. Les partitions de disque sur Linux

## XI.2. Les commandes fdisk

### ❖ Supprimer une partition de disque

Ensuite on peut supprimer la partition avec disque.

- 1) **Affichez la liste des partitions** avec la **commande p** afin d'avoir les numéros
- 2) Ensuite on utilise la **commande d**
- 3) Puis on doit indiquer **le numéro de partition à supprimer**
- 4) A nouveau lister les partitions avec **p**
- 5) Enfin validez les modifications avec la **commande w**

TAF :

- Créer une partition de disque avec fdisk
- Redimensionner une partition de disque avec fdisk
- Supprimer une partition de disque avec fdisk

# XI. Les partitions de disque sur Linux

## XI.2. Les commandes fdisk

### ❖ Monter des partitions de disques

Sur Linux, lorsque l'on ajoute un nouveau disque local ou après avoir créé une partition de disque, vous devez **monter la partition de disque** dans un emplacement que l'on ne nomme point **de montage**.

La commande **mount** peut monter n'importe **quel type de périphériques** supportés par le noyau Linux (**ext3, ext4, nfs, hpfs, ntfs, vfat, cifs, jfs, ReiserFS, XFS, Btrfs, ...**).

Pour cela, on utilise deux éléments :

- ✓ **mount** : la commande pour monter un disque local ou réseau sur un point de montage
- ✓ **umount** : démonter la partition de disque

**/etc/fstab** : le fichier de configuration des points de montage automatique au démarrage de Linux



# XI. Les partitions de disque sur Linux

## XI.3. La commande mount

❖ Monter des partitions de disques

L'utilisation générique est :

**mount -t TYPE NOM\_PERIPHERIQUE REPERTOIRE**

Si vous ne spécifiez pas **l'option -t**, **mount** tente de déduire le système de fichiers du périphérique pour le monter.

# XI. Les partitions de disque sur Linux

## XI.3. La commande mount

### ❖ Monter des partitions de disques

Les options	Description
<b>-a</b>	Monte tous les périphériques décrits dans <code>/etc/fstab</code>
<b>-l</b>	Répertorie tous les systèmes de fichiers déjà montés
<b>-o, -options</b>	Utilisez les options de montage spécifiées. L'argument <code>opts</code> est une liste séparée par des virgules. Par exemple: <code>mount LABEL = mydisk -o noatime, nodev, nosuid</code>
<b>-r</b>	Mode lecture seule monté
<b>-t</b>	Type de périphérique de système de fichiers utilisé
<b>-T</b>	Décrit un fichier <code>fstab</code> alternatif
<b>-h</b>	Affiche les options de commande
<b>-V</b>	Affiche les informations de version

# XI. Les partitions de disque sur Linux

## XI.3. La commande mount

### ❖ Les options de la commande mount

Par exemple pour **monter une partition NTFS Windows** se trouvant dans `/dev/sda1` sur Linux :

```
mount -t nfs /dev/sda1 /media/sda1/
```

**Le paramètre -o de mount** est utile pour spécifier des options additionnelles.

Il existe des options communes (comme `rw`, `ro`, etc) communes à tous les types de périphériques et des options spécifiques aux types de périphériques.

# XI. Les partitions de disque sur Linux

## XI.3. La commande mount

### ❖ Monter des partitions de disques

Rien de vraiment particulier pour monter des partitions de disque, on utilise la syntaxe classique de mount.

Par exemple pour **monter le disque /dev/sdb2 sur /data** :

**mount /dev/sdb2 /media/usb**

Pour monter une partition de disque en FAT32 :

**mount -t vfat /dev/sda1 /media/usb**

Pour la monter en lecture seule :

**mount -t vfat -o ro,noexec /dev/sda1 /media/usb**

Mais aussi pour **monter une clé USB** :

**mount /dev/sdd1 /media/usb**

## XI.3. La commande mount

Les options -o communes	Description
async	Les modifications des données des fichiers sont initialement effectuées en mémoire. Cela accélère les performances mais est plus susceptible d'entraîner une perte de données. Sur certaines distributions, c'est la valeur par défaut.
defaults	Utilisez les options par défaut: rw, suid, dev, exec, auto, nouser et async
rw ro	Monte la partition en lecture/écriture (read-write) ou en lecture seule (read-only).
group users	Permettez à un utilisateur ou groupe utilisateur de monter le système de fichiers.  Cette option implique les options noexec, nosuid et nodev (sauf si elles sont remplacées par les options suivantes, comme dans la ligne d'option user, exec, dev, suid).
exec noexec	Permet ou ne permet pas d'exécuter des fichiers (programmes ou script) situés sur la partition.
noatime	Ne pas mettre à jour les temps d'accès aux inodes sur ce système de fichiers.  Cela fonctionne pour tous les types d'inœuds (répertoires aussi), donc cela implique nodiratime.
owner	Autorise un utilisateur ordinaire à monter le système de fichiers si cet utilisateur est le propriétaire du périphérique.  Cette option implique les options nosuid et nodev (sauf si elles sont remplacées par les options suivantes, comme dans la ligne d'options propriétaire, dev, suid).
relatime	L'inode de répertoire n'est pas mis à jour sur le système de fichiers lors de l'accès. Cela peut améliorer les performances de la même manière que ne pas mettre à jour le temps d'accès aux fichiers.
remount	Essaye de remonter un système de fichiers déjà monté. Ceci est couramment utilisé pour modifier les indicateurs de montage d'un système de fichiers, en particulier pour rendre un système de fichiers en lecture seule accessible en écriture. Cela ne change pas le périphérique ou le point de montage.
sync	Les modifications des données de fichier sont effectuées sur le disque immédiatement, ce qui a un impact sur les performances, mais est moins susceptible d'entraîner une perte de données. Sur certaines distributions, c'est la valeur par défaut.

# CONCLUSION

**MERCI DE VOTRE  
AIMABLE ATTENTION!!!**

