

Département Mathématique et Informatique  
Décisionnelle  
Niveau : 1-IDSD

---

Cours  
Commande et  
administration Linux  
(Préparation certif LPIC 1)

---



Préparé par

Sonda Ammar

Maître Assistant à l'Ecole Nationale d'Electronique et des Télécommunications de Sfax

Email : [sonda.ammar@enetcom.usf.tn](mailto:sonda.ammar@enetcom.usf.tn)





Préparation à l'examen 101 pour la certification de l'Institut professionnel de Linux, (LPIC-1)

SONDA AMMAR (1<sup>er</sup> Edition)

September 2023



## Table des matières

<b>1</b>	<b>Introduction au monde UNIX</b>	<b>6</b>
1.1	Qu'est-ce qu'un système d'exploitation? . . . . .	6
1.1.1	SE comme une extension de la machine . . . . .	6
1.1.2	SE comme un gestionnaire de ressources . . . . .	7
1.2	Les grands systèmes d'exploitation . . . . .	8
1.2.1	Le système UNIX . . . . .	9
1.2.2	Historique du système Unix . . . . .	9
1.2.3	Architecture logique d'un système UNIX . . . . .	10
1.3	Le projet GNU, la Free Software Foundation (FSF) et la licence GPL . . . . .	11
1.4	GNU/Linux . . . . .	12
1.4.1	Historique . . . . .	12
1.4.2	Bureau KDE . . . . .	12
1.4.3	Bureau GNOME . . . . .	13
1.5	Distributions Linux . . . . .	14
1.5.1	Ubuntu . . . . .	15
1.5.2	Installer Ubuntu . . . . .	15
1.6	Premiers pas sous Ubuntu . . . . .	16
1.6.1	Le terminal . . . . .	16
1.6.2	Syntaxe d'une commande . . . . .	17
1.6.3	Obtenir de l'aide . . . . .	17
<b>2</b>	<b>Gestion des fichiers</b>	<b>19</b>
2.1	Introduction . . . . .	19
2.2	Travailler en ligne de commandes . . . . .	20
2.2.1	Séquence de commandes . . . . .	21
2.2.2	Commandes utiles . . . . .	21
2.3	Gestion de base de fichiers . . . . .	21
2.3.1	L'arborescence LINUX . . . . .	22
2.3.2	Chemin d'un répertoire . . . . .	22
2.3.3	La commande ls . . . . .	23

2.3.4	Manipulation des répertoires et des fichiers . . . . .	24
2.3.5	Copier et déplacer les fichiers . . . . .	25
2.3.6	Les liens . . . . .	27
2.3.7	Les méta-caractères . . . . .	29
2.3.8	Les expressions régulières . . . . .	29
2.4	Les processus . . . . .	32
2.4.1	Arborescence des processus . . . . .	33
2.4.2	Cycle de vie du processus . . . . .	33
2.4.3	Gestion de processus . . . . .	34
2.4.4	Envoyer un signal à un processus . . . . .	37
2.4.5	Avant-plan et arrière-plan . . . . .	38
2.4.6	Modifier les priorités d'exécution de processus . . . . .	39
2.4.7	Les tubes et les redirections . . . . .	40
2.5	Éditeur VI . . . . .	43
2.5.1	La saisie . . . . .	43
2.5.2	Quitter et sauver . . . . .	43
2.5.3	Correction . . . . .	44
2.5.4	Recherche . . . . .	45
2.5.5	Copier-Coller . . . . .	45
<b>3</b>	<b>Gestion des droits d'accès</b>	<b>47</b>
3.1	Introduction . . . . .	47
3.2	Organisation des utilisateurs sous Linux . . . . .	48
3.3	Utilisateurs et permissions . . . . .	48
3.3.1	Sémantique des permissions . . . . .	49
3.3.2	Chmod : change mode . . . . .	49
3.3.3	Chown : change owner . . . . .	51
3.3.4	Chgrp : change group . . . . .	51
3.3.5	umask : user mask . . . . .	52
3.4	Gestion des utilisateurs . . . . .	53
3.5	Gestion des groupes . . . . .	55
3.6	Les droits spéciaux et utilisation . . . . .	55
3.6.1	Le sticky bit . . . . .	56
3.6.2	SUID . . . . .	56
3.6.3	GUID . . . . .	57
<b>Webography</b>		<b>59</b>
<b>Bibliography</b>		<b>60</b>

1.1	Historique du système Unix . . . . .	9
1.2	Différentes versions de KDE . . . . .	13
1.3	Différentes versions de GNOME . . . . .	14
1.4	Exemples de distributions Linux . . . . .	14
1.5	Interface Ubuntu . . . . .	15
2.1	Le shell par rapport au noyau . . . . .	20
2.2	Arborescence du système LINUX . . . . .	23
2.3	Exemple de chemin relatif et de chemin absolu . . . . .	24
2.4	Exemples d'utilisation : déplacer et renommer un répertoire . . . . .	26
2.5	Arborescence de répertoires . . . . .	27
2.6	Les liens symbolique et physique sous Linux . . . . .	28
2.7	Cycle de vie d'un processus . . . . .	34
2.8	Exemple de résultat de la commande ps . . . . .	35
2.9	Exemple de résultat de la commande pstree . . . . .	36
2.10	Les canaux standards de lecture et d'écriture . . . . .	41
2.11	Le mécanisme des tubes . . . . .	42
3.1	Organisation des utilisateurs sous Linux . . . . .	48

## Table des figures

## Liste des tableaux

2.1	Liste des méta-caractères . . . . .	29
2.2	Les commandes de saisie . . . . .	44
2.3	Les commandes pour sauvegarder et quitter . . . . .	44
2.4	Les commandes de correction . . . . .	44
3.1	Les permissions sur les fichiers et les répertoires . . . . .	49
3.2	Les permissions sur les fichiers et les répertoires . . . . .	49
3.3	Liste des permissions pour le répertoire Ex_droit . . . . .	53
3.4	Les droits spéciaux . . . . .	56

## Chapitre 1

### Introduction au monde UNIX

#### Contenu du chapitre 1

- Qu'est-ce qu'un système d'exploitation ?
- Historique du système Unix
- Logiciel Libre
- Quel matériel pour Linux ?
- Distributions Linux
- La distribution Ubuntu
- Obtenir de l'aide

#### 1.1 Qu'est-ce qu'un système d'exploitation ?

Les Systèmes d'Exploitation (SE) constituent le cœur de toute expérience informatique moderne. En effet, ces logiciels agissent en tant qu'interface entre l'utilisateur, les applications et le matériel informatique, coordonnant efficacement les ressources et facilitant l'exécution des tâches informatiques. Ainsi, un système d'exploitation est une couche logicielle qui permet de gérer les périphériques et de fournir aux programmes utilisateur une interface simplifiée avec le matériel. Alors, le SE remplit deux tâches : i) une extension de la machine et ii) un gestionnaire de ressources.

##### 1.1.1 SE comme une extension de la machine

Le rôle d'un SE est de masquer la complexité inhérente des machines informatiques et les détails complexes du fonctionnement du matériel en fournissant à l'utilisateur une interface simplifiée et conviviale. Selon Coy, cette abstraction peut être comprise en termes de différentes couches de complexité,

chacune ajoute une certaine forme d'abstraction par rapport à la précédente. Pour illustrer ce concept, Coy divise la notion de "machine" en trois niveaux distincts : la machine réelle, la machine abstraite et la machine utilisable.

Une machine réelle est constituée de composants matériels, à savoir l'unité centrale et une variété de périphériques tels que le processeur, la mémoire, le stockage et les périphériques d'entrée/sortie. Chacun de ces composants possède ses propres caractéristiques, interfaces et modes de fonctionnement spécifiques, ce qui rend l'interaction directe avec ces éléments peu pratique pour les utilisateurs. C'est là que le système d'exploitation entre en jeu, en créant une couche intermédiaire entre le matériel et les logiciels applicatifs. Cette abstraction se matérialise sous la forme d'une machine abstraite, qui combine la machine réelle avec le système d'exploitation. Ainsi, le SE offre une interface unifiée et simplifiée aux utilisateurs, leur permettant d'interagir avec la machine de manière cohérente et intuitive. Pour que cette machine soit exploitée par les utilisateurs, un ensemble d'application est nécessaire, donnant lieu à une machine utilisable.

En fournissant cette interface utilisateur, le système d'exploitation cache efficacement la complexité du matériel, permettant aux utilisateurs de se concentrer sur leurs tâches et leurs objectifs sans être encombrés par les détails techniques. En fournissant cette interface utilisateur conviviale, le système d'exploitation cache efficacement la complexité du matériel, permettant aux utilisateurs de se concentrer sur leurs tâches et leurs objectifs sans être encombrés par les détails techniques. Cette approche rend l'informatique plus accessible et utilisable pour un large éventail d'utilisateurs, qu'ils soient débutants ou expérimentés dans le domaine de la technologie.

### 1.1.2 SE comme un gestionnaire de ressources

Un système d'exploitation agit comme un gestionnaire de ressources en orchestrant efficacement l'utilisation des ressources matérielles disponibles dans un ordinateur. Il gère principalement quatre types de ressources : le processeur, les fichiers, la mémoire et les entrées/sorties. Voici un bref aperçu de la manière dont un SE exerce cette gestion dans différentes ressources :

- **Gestion des processus** : Le SE gère les ressources du processeur entre les différents processus en cours d'exécution. Il planifie l'ordonnancement, la commutation et la synchronisation des processus pour optimiser l'utilisation du processeur.
- **Gestion des fichiers** : Le SE fournit des mécanismes pour créer, modifier, déplacer, copier et supprimer des fichiers de manière cohérente et sécurisée. Il gère également les permissions d'accès aux fichiers.

- **Gestion de la mémoire** : Le SE contrôle l'allocation et la libération de la mémoire système entre les différents processus en cours d'exécution.
- **Gestion des entrées/sorties (E/S)** : Le SE coordonne les opérations d'entrée/sortie entre les périphériques d'entrée/sortie (clavier, souris, disques, etc.) et les processus. Il fournit des interfaces normalisées et des pilotes de périphériques pour permettre aux processus d'accéder aux périphériques de manière uniforme.

## 1.2 Les grands systèmes d'exploitation

Dans cette partie, nous explorerons certains des grands systèmes d'exploitation largement utilisés à travers le monde.

- **Windows** : Windows, développé par Microsoft, est l'un des systèmes d'exploitation les plus répandus. Depuis son lancement avec Windows 1.0 en 1985, il a connu plusieurs évolutions majeures, offrant des versions telles que Windows 3.1, Windows 95, Windows XP, jusqu'à sa dernière version Windows 11. Windows est connu par son interface graphique conviviale, sa compatibilité logicielle étendue et son support matériel varié. Il demeure ainsi un choix populaire pour les utilisateurs du monde entier. Pour une documentation détaillée sur les différentes versions de Windows, vous pouvez consulter [cette source<sup>1</sup>](#).
- **macOS** : macOS est un système d'exploitation lancé par Apple, pour la première fois le 24 mars 2001. MacOS est connu pour son design élégant et son intégration étroite avec les produits Macintosh. Il a vu plusieurs itérations au cours de ces deux décennies. La dernière version de macOS est Ventura, arrivée le lundi 24 octobre 2022. Pour plus d'informations sur les différentes versions et mises à jour de macOS, vous pouvez consulter [cette source<sup>2</sup>](#).
- **Linux** : Linux est un système d'exploitation open-source largement utilisé, offrant une variété de distributions telles qu'Ubuntu, Fedora et CentOS. Il est couramment utilisé dans les environnements professionnels, les serveurs, le cloud computing, l'embarqué et les supercalculateurs. Pour une documentation approfondie sur Linux, vous pouvez visiter [cette source<sup>3</sup>](#).

1. <https://www.malekal.com/historique-versions-windows/Historique>

2. <https://frenchmac.com/actualite/mac-os-x-macos-versions-publiees-present/>

3. <https://linux.goffinet.org/administration/introduction-a-linux/evolution-de-linux/>

De plus, pour une comparaison de la popularité des systèmes d'exploitation entre 2004 et 2019, vous pouvez visionner [cette vidéo](#)<sup>4</sup>.

### 1.2.1 Le système UNIX

### 1.2.2 Historique du système Unix

Le système Unix date depuis les années 1960, qui représente ses modestes débuts dans les laboratoires de recherche. Unix a joué un rôle majeur dans le façonnement de l'industrie technologique. Il possède une présence prédominante dans les systèmes informatiques actuels. Cette section examinera les origines, le développement et les jalons importants de cette histoire, en soulignant les contributions et les avancées qui ont forgé l'identité actuelle d'Unix.

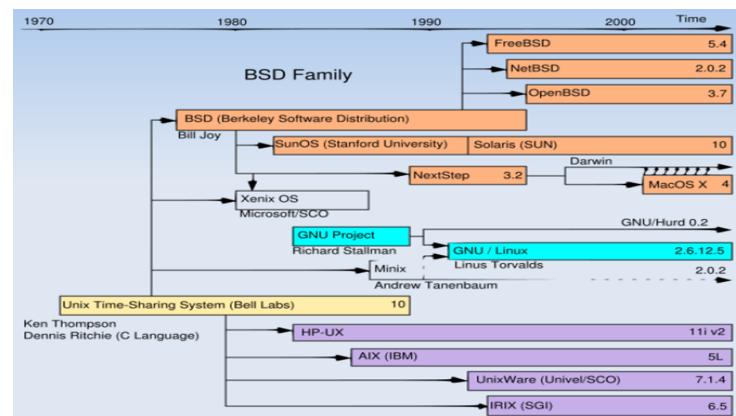


FIGURE 1.1 – Historique du système Unix

- En 1964, le laboratoire Bell Labs et la General Electric ont lancé le projet MULTICS (Multiplexed Information and Computing Service) pour répondre à divers besoins, notamment :
  - la possibilité d'être utilisé simultanément par plusieurs utilisateurs
  - l'exécution de tâches en arrière-plan et
  - une meilleure gestion de la sécurité

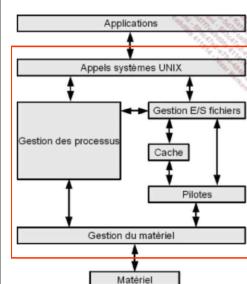
4. <https://www.youtube.com/Les systèmes d'exploitations les plus populaires 2004-2019>

- En 1973, Unix a été réécrit en langage C afin d'assurer sa portabilité sur différents types de matériel.
- Plusieurs sociétés telles que IBM, Sun et HP, ont manifesté un intérêt pour Unix et ont développé leurs propres versions, à savoir Solaris de Sun, AIX d'IBM, HP-UX de HP et FreeBSD de l'Université Berkeley.
- En 1987, une première version de Minix, une variante Unix conçue pour les PC, a été publiée.

### 1.2.3 Architecture logique d'un système UNIX

L'architecture logique d'un système UNIX représente la structure qui définit la manière dont les différents composants et fonctionnalités du système sont organisés et interagissent entre eux. Dans la suite, nous explorerons les différentes couches et composants essentiels de cette architecture, mettant en évidence leur fonction et leurs interactions.

- Les appels système servent de moyen de communication entre les programmes et le système d'exploitation.
- La gestion des processus assure la communication des tâches et leur priorisation (multitâche).
- La gestion des E/S des fichiers gère la lecture et l'écriture de données sur le disque dur ainsi que sur les périphériques externes tels que la carte son ou l'imprimante.
- La mémoire cache permet à UNIX d'écrire les données dans une zone mémoire intermédiaire plutôt que directement sur le disque dur, ce qui améliore les performances.
- Les pilotes assurent la gestion du matériel au niveau le plus bas de la machine.



### 1.3 Le projet GNU, la Free Software Foundation (FSF) et la licence GPL

À la fin de l'année 1983, [Richard Stallman](#)<sup>5</sup> a lancé le projet GNU en réponse à la pratique de garder le code source des logiciels secret et d'imposer le paiement de licences pour leur utilisation. L'objectif du projet était de développer un système d'exploitation complet, similaire à Unix, mais qui serait un logiciel libre. Ainsi est né le système GNU, dont le nom est un acronyme récursif signifiant " GNU is Not Unix ". Pour soutenir le développement de ces logiciels libres, Stallman a également fondé la FSF (Free Software Foundation). Cette organisation a été chargée de collecter des fonds pour financer le développement de nombreux logiciels, tels que l'éditeur Emacs et le compilateur GCC. De plus, la FSF a travaillé avec des avocats pour rédiger une licence spéciale pour ces logiciels, connue sous le nom de GPL (General Public License).

D'après les créateurs de la licence GNU GPL, cette licence permet de garantir à l'utilisateur divers droits (appelés libertés) sur un programme informatique. Un logiciel libre est à prendre dans le sens de "liberté" et pas de gratuité. En fait, la gratuité est un effet de la liberté. La liberté au sens GNU est définie selon quatre principes :

1. **Liberté 0** : La liberté d'utiliser un logiciel.
2. **Liberté 1** : La liberté d'étudier le fonctionnement du programme et de l'adapter à votre besoin.
3. **Liberté 2** : La liberté de redistribuer des copies.
4. **Liberté 3** : La liberté d'améliorer le programme et de diffuser les améliorations au public afin d'en faire bénéficier l'ensemble de la communauté.

Un grand nombre de logiciels libres tels que Tex-LaTeX, Emacs, PHP, MySQL, etc sont publiés sous la licence GNU GPL.

Pour une documentation approfondie sur le projet GNU et la licence GPL, vous pouvez visiter [cette source](#)<sup>6</sup>.

---

5. <https://stallman.org/>

6. <https://www.gnu.org/gnu/thegnuproject.fr.html>

### 1.4 GNU/Linux

Linux, également connu sous le nom de GNU/Linux, est un système d'exploitation de type Unix, fondé sur le noyau Linux. Ce noyau Linux est le cœur, la partie centrale, du système, chargée de la gestion des ressources matérielles et de la fourniture d'interfaces pour les logiciels. Le système d'exploitation Linux est un système d'exploitation open source, distribué sous la licence GPL du projet GNU, ce qui permet à quiconque d'accéder à son code source, le modifier et le redistribuer librement. Cette caractéristique a conduit à la création de multitude de distributions Linux complètes. Le système d'exploitation Linux se caractérise principalement par sa capacité à gérer de manière efficace le multitâche et à prendre en charge plusieurs utilisateurs simultanément, multi-utilisateur. C'est pourquoi Linux est largement utilisé dans divers domaines, notamment les serveurs web, les supercalculateurs, les appareils embarqués, les ordinateurs personnels, et bien d'autres. Il est recommandé pour sa stabilité, fiabilité, sécurité et flexibilité, ainsi que pour la variété des distributions disponibles, qui peuvent être adaptées à des besoins spécifiques.

#### 1.4.1 Historique

En 1991, Linus Torvalds, un étudiant finlandais, a lancé le développement du noyau Linux en s'inspirant du système Minix. De 1994 à 1997, plusieurs distributions majeures de Linux ont été créées, telles que Red Hat, Debian, SUSE et Slackware. Pendant cette période, de nombreuses améliorations ont été apportées au système, ce qui l'a rendu plus adapté à une utilisation sur poste de travail. Cela a également coïncidé avec le début du développement des environnements de bureau GNOME et KDE, visant à fournir des interfaces graphiques conviviales pour les utilisateurs de Linux.

#### 1.4.2 Bureau KDE

En 1996, la première version 1.0 du bureau KDE, "K Desktop Environment", développée par Matthias Ettrich, apparaît. Cette première version du KDE présentait une interface similaire à Windows 95. Son interface comporte, en bas de l'écran, une barre des tâches intégrant divers raccourcis vers les applications. Cette interface comporte également, en haut de l'écran, des boutons pour accéder aux applications en cours d'exécution et aux bureaux virtuels. La bibliothèque d'outils Qt, utilisée pour le développement de KDE, était disponible gratuitement pour les projets de logiciels libres, mais

son utilisation dans des applications commerciales nécessitait des frais de licence. Cette politique a soulevé des préoccupations au sein de nombreuses distributions Linux, qui ont donc hésité à intégrer ce nouvel environnement dans leurs distributions. En 2000, la version KDE 2.0 a été lancée, marquant une réécriture presque complète du bureau KDE, suivi de nombreuses autres versions. La figure 1.2 illustre les premières versions de KDE ainsi que leurs évolutions jusqu'à la version apparue en 2024.



FIGURE 1.2 – Différentes versions de KDE

Pour découvrir le comité de KDE, son historique, ainsi que ses différentes versions, vous pouvez visiter [cette source](https://25years.kde.org/fr/)<sup>7</sup>. Vous pouvez visiter aussi le site officiel de KDE sur [cette source](https://kde.org/fr/announcements/)<sup>8</sup>.

#### 1.4.3 Bureau GNOME

En 1997, et dans le but de remédier au problème rencontré par KDE, Miguel de Icaza et Federico Mena ont initié le développement d'un nouvel environnement de bureau pour Linux, nommé GNOME, qui signifie GNU Network Object Modeling Environment. Pour éviter les problèmes rencontrés par KDE, GNOME a opté pour l'utilisation d'une bibliothèque différente appelée GTK, développée à l'origine pour l'éditeur d'images Gimp. GNOME a été officiellement publié en 1999. En raison de sa liberté, l'environnement GNOME a été inclus dans la plupart des distributions GNU/Linux, notamment, Debian, Red Hat/Fedora, openSUSE, Mageia. La figure 1.3 illustre les premières versions de GNOME ainsi que la version apparue en 2024.

7. <https://25years.kde.org/fr/>

8. <https://kde.org/fr/announcements/>



FIGURE 1.3 – Différentes versions de GNOME

Vous pouvez visiter le site officiel de GNOME sur [cette source](https://www.gnome.org/)<sup>9</sup>.

## 1.5 Distributions Linux

Les Distributions Linux représentent une large gamme de systèmes d'exploitation open-source qui jouent un rôle essentiel dans l'écosystème Linux. Elles proposent une variété de logiciels, de pilotes et d'outils dans une interface utilisateur intuitive, partageant toutes le noyau Linux. Chaque distribution est unique, offrant des fonctionnalités spécifiques adaptées à différents cas d'utilisation, aux préférences des utilisateurs et aux exigences professionnelles. La figure 1.4 cite plusieurs distributions Linux.



FIGURE 1.4 – Exemples de distributions Linux

Face aux centaines de distributions Linux proposées sur le marché, plus de 350 distributions, il est devenu difficile de faire un choix. Plusieurs ressources facilitent notre tâche. Vous pouvez ainsi consulter ces ressources, [source 1](https://www.youtube.com/watch?v=qd6FibkktRM)<sup>10</sup>,

9. <https://www.gnome.org/>

10. <https://www.youtube.com/watch?v=qd6FibkktRM>

source 2<sup>11</sup> et source 3<sup>12</sup>.

En tant que débutant dans le monde de Linux, Ubuntu est un choix idéal pour nous. Il est également préféré par ceux qui cherchent une distribution Linux stable et bien supportée.

### 1.5.1 Ubuntu

Ubuntu, figure 1.5, est un système d'exploitation GNU/Linux, basé sur la distribution Linux Debian. Il est caractérisé par sa liberté, sa gratuité et sa simplicité d'utilisation. Le système d'exploitation Ubuntu figure parmi les distributions Linux les plus populaires. Il intègre le bureau GNOME avec une interface utilisateur intuitive et offrant un large choix de logiciels. Une vaste communauté d'utilisateurs et de développeurs s'intéresse à Ubuntu et qui est très active et engagée.



FIGURE 1.5 – Interface Ubuntu

Le site officiel d'Ubuntu est accessible via ce lien<sup>13</sup>.

### 1.5.2 Installer Ubuntu

Pour utiliser la distribution Ubuntu, plusieurs options s'offrent à vous, chacune adaptée aux besoins et configurations matérielles :

— **Installation sur une partition dédiée** : Cette solution consiste à créer une partition dédiée sur le disque dur et y installer Ubuntu

11. <https://www.clubic.com/telecharger/actus-logiciels/article-842846-1-10-distributions-gnu-linux-preferees-dire-adieu-windows-10.html>

12. <https://www.justgeek.fr/les-meilleures-distributions-linux-107294/>

13. <https://www.ubuntu-fr.org/>

comme système d'exploitation principal. Cela vous permettra de bénéficier d'une installation complète et permanente d'Ubuntu sur votre ordinateur.

- **Utilisation d'Ubuntu en version live (Bootable USB)** : Une autre option consiste à créer une clé USB bootable contenant une version live d'Ubuntu. Cela vous permet de lancer votre ordinateur à partir de la clé USB sans installer Ubuntu sur votre disque dur. Vous pouvez suivre en ligne via ce lien<sup>14</sup> pour créer une clé USB bootable avec Ubuntu.
- **Utilisation de Windows Subsystem for Linux (WSL 2)** : Utilisant Windows 10 ou plus, vous pouvez également installer Ubuntu en tant que sous-système Linux grâce à Windows Subsystem for Linux (WSL 2). Cela vous permet d'exécuter des applications Linux directement sur votre système Windows sans avoir besoin de créer de partition supplémentaire ou de démarrer à partir d'une clé USB. Une documentation des étapes à suivre se trouve sur ce lien<sup>15</sup>.
- **Utilisation d'un émulateur** : Vous pouvez également installer Ubuntu dans une machine virtuelle en utilisant un émulateur tel que VMware ou VirtualBox. Ces émulateurs vous permettent de créer et d'exécuter des machines virtuelles Ubuntu sur votre système d'exploitation principal, qu'il s'agisse de Windows, macOS ou Linux. Vous pouvez télécharger l'émulateur VirtualBox depuis ce lien<sup>16</sup> et l'image ISO d'Ubuntu depuis le site officiel<sup>17</sup> et suivre ce lien<sup>18</sup> pour installer Ubuntu dans votre machine virtuelle.

Quelle que soit l'option que vous choisissez, assurez-vous de suivre les instructions spécifiques à votre configuration et à vos besoins pour une expérience optimale avec Ubuntu.

## 1.6 Premiers pas sous Ubuntu

### 1.6.1 Le terminal

Le terminal, la ligne de commande ou l'invite de commande, est un utilitaire robuste qui permet aux utilisateurs de communiquer avec le système d'exploitation Ubuntu à l'aide de commandes textuelles. Alors que l'interface

14. <https://tutorials.ubuntu.com/tutorial/tutorial-create-a-usb-stick-on-windows#0>

15. <https://docs.microsoft.com/en-us/windows/wsl/install-win10>

16. <https://www.virtualbox.org/wiki/Downloads>

17. <https://www.ubuntu-fr.org/>

18. [https://www.youtube.com/watch?v=PlJx0XpyILU&ab.channel=UNEHEURELINUX](https://www.youtube.com/watch?v=PlJx0XpyILU&ab_channel=UNEHEURELINUX)

graphique d'Ubuntu offre une expérience conviviale et intuitive, le terminal représente une solution efficace pour accomplir des tâches avancées, automatiser des processus et résoudre des problèmes système.

toto@ubuntu : ~ \$

- toto** : C'est le pseudo sous lequel vous vous êtes connectés.
- @ubuntu** : le nom de l'ordinateur sur lequel vous travaillez (il est attribué lors de l'installation d'Ubuntu)
- ~** : C'est le dossier personnel, le "Home" sous Linux ("Mes documents" sous Windows).
- \$** : indiquant le type d'utilisateur connecté :
- \$** : indique qu'il s'agit d'un simple utilisateur
- #** : indique qu'il s'agit de l'administrateur ("root")

## 1.6.2 Syntaxe d'une commande

cmd - : [option(s)] paramètre(s)

**cmd** : le nom de la commande en minuscules  
Un ou plusieurs espaces (ou <TAB>) sont introduits comme séparateurs entre la commande (cmd), les options et le(s) paramètre(s). Le caractère "-" introduit les options de la commande. Généralement, les options peuvent être regroupées.

**Exemple :** "ls -all" est équivalent à "ls -a -l -L"

## 1.6.3 Obtenir de l'aide

Aide propre aux commandes : - -help

Syntaxe : help - -help

Aide interne au shell : Les commandes internes n'acceptent pas de paramètre -help, il faut alors utiliser la commande help

Syntaxe : date pwd

Syntaxe : man commande

Afficher le manuel de la commande  
Afficher une information succincte sur la commande

Syntaxe : whatis commande

# Chapitre 2

## Gestion des fichiers

### Contenu du chapitre 2

- Travailler en ligne de commande
- Effectuer la gestion de base des fichiers
- Contrôler des flux de texte à l'aide des filtres
- Recherche sur des fichiers texte avec des expressions régulières
- Création, surveillance et destruction de processus
- Modifier la priorité d'exécution d'un processus
- Utilisation des flux, des tubes (pipes) et des redirections
- Édition de fichiers texte avec "vi"

### 2.1 Introduction

Un fichier est, au sens commun, une collection logique d'information réunie sous un même nom. Un système de fichiers est une manière de stocker les informations et de les organiser dans des fichiers.

Une des fonctions d'un système d'exploitation est de proposer aux utilisateurs en général et aux programmeurs en particulier un modèle de manipulation des fichiers simple, facile, agréable et indépendant du matériel utilisé. Ceci est offert en masquant les spécificités et ignorant les caractéristiques de l'ensemble du matériel, tels que les disques et les autres périphériques d'entrées/sorties. Les appels système permettent de créer, de modifier et de supprimer des fichiers et aussi de lire et d'écrire dans ces fichiers. Les fichiers sont regroupés en répertoires constituant une arborescence. Le déplacement dans cette arborescence pour accéder à un fichier est en énonçant leur chemin d'accès, chemin d'accès absolu ou chemin d'accès relatif. Le système d'exploitation gère également la protection des fichiers.

### 2.2 Travailler en ligne de commandes

Objectifs | Se familiariser avec la Ligne de Commande et les commandes les plus courantes

Mots clés | Shell, Bash, ligne de commande

Le Shell (coquille par opposition au noyau, tel qu'il est illustré par la Figure 2.1) est un interpréteur de commandes permettant la communication avec le système d'exploitation par l'intermédiaire d'un langage de commandes. Utilisant le Shell, l'utilisateur peut piloter les périphériques tout en ignorant la gestion des adresses physiques, les caractéristiques du matériel, etc. Ainsi, la fonction principale de l'interpréteur de commandes est d'obtenir et d'exécuter une succession de commandes spécifiées par l'utilisateur. Pour exécuter une commande, l'interpréteur de commandes vérifie sa validité, sinon il génère une erreur.

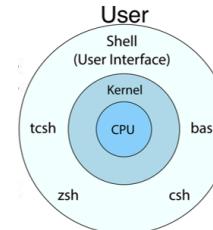


FIGURE 2.1 – Le shell par rapport au noyau

#### - Exemple sur Unix/Linux : csh, tcsh, ksh :

- Le Bourne shell (sh) est l'interpréteur original de l'environnement UNIX. Sa grande originalité était l'utilisation de tubes (caractère "|"), qui permettent de connecter la sortie d'une commande à l'entrée d'une autre. On peut ainsi écrire des commandes complexes à partir de commandes simples.
- Bash : Bourneagain shell est l'interpréteur de commande par défaut d'Ubuntu. Il est compatible avec des fonctionnalités de ksh, csh

### 2.2.1 Séquence de commandes

La ligne de commande accepte l'insertion d'une seule commande, comme elle favorise l'insertion de plusieurs commandes. Dans ce cas et selon la nature du séparateur, le processus d'exécution des commandes se différencie :

- Exécuter séquentiellement des commandes l'une après l'autre :  
**cmd1 ; cmd2**
  - Exécuter cmd2 si et seulement si cmd1 s'est exécutée sans erreur :  
**cmd1 && cmd2**
  - Exécuter cmd2 si et seulement si cmd1 a renvoyé une erreur :  
**cmd1 || cmd2**
  - & en fin de commande permet de lancer cette commande en tâche de fond (background) :  
**cmd&**
- Exemple :** ./firefox&

### 2.2.2 Commandes utiles

Une liste non exhaustive de commandes utiles sera découverte tout au long de ce cours. Dans cette partie, nous exposons quelques commandes très utiles pour commencer :

- Afficher la durée d'exécution d'une commande : **time pwd**
- Afficher l'historique des commandes : **history**
- Supprimer l'historique des commandes : **history -c**
- Changer le mode du clavier de "qerty" en "azerty" : **setxkbmap fr**
- Nettoyer le shell : **clear**
- Afficher le calendrier : **cal**

## 2.3 Gestion de base de fichiers

### Objectifs

- Utiliser les commandes Linux de base pour gérer les fichiers et les répertoires.
- Copier, déplacer et supprimer des fichiers et des répertoires individuellement
- Copier plusieurs fichiers et répertoires de manière récursive
- Supprimer les fichiers et les répertoires de manière récursive
- Utilisez des spécifications génériques simples et avancées dans les commandes

### Mots clés

mkdir, touch,

### 2.3.1 L'arborescence LINUX

Sous le système Linux, la racine du système est représentée par " / ". Les répertoires de base de l'arborescence standard de fichiers sous Linux sont les suivants :

- **/boot** : contient principalement le fichier binaire du noyau et les ressources nécessaires à son lancement lors du démarrage ;
- **/dev** : contient les fichiers des périphériques de la machine ainsi que des fichiers spéciaux ;
- **/etc** : répertoire très important où sont stockés tous les fichiers de configuration du système en général et des différentes variables (démone) en particulier. Il s'agit du répertoire à sauvegarder pour pouvoir restaurer la configuration d'une machine ;
- **/home** : contient les répertoires des utilisateurs du système. Elle représente le répertoire de stockage par défaut ;
- **/proc** : contient les informations nécessaires au noyau. C'est une arborescence virtuelle généralement en lecture seule sauf proc/sys ;
- **/root** : répertoire home du super-utilisateur, appelé le "root" ;
- **/tmp** : permet aux applications et aux utilisateurs d'avoir un espace d'échange où ils peuvent stocker leurs fichiers temporaires. Il est effacé à chaque redémarrage, "reboot", de la machine ;
- **/usr** : contient les fichiers nécessaires aux applications, la documentation, les manuels, les fichiers sources et les bibliothèques qui sont généralement statiques et générées à l'installation des logiciels standards de la distribution ;
- **/usr/local** : arborescence qui sert à installer des logiciels supplémentaires ;
- **/var** : contient les fichiers journaux des différentes variables ainsi que les spools de mail, d'impression, etc.

La figure 2.2 présente les répertoires de base de l'arborescence standard de fichiers sous Linux.

### 2.3.2 Chemin d'un répertoire

On distingue deux types de chemins :

1. **Chemin absolu** : Un chemin absolu commence par le répertoire racine et suit l'arborescence, branche par branche, jusqu'à atteindre le répertoire ou le fichier souhaité. Les chemins absolus commencent toujours par **/**.
2. **Chemin relatif** : Un chemin relatif commence à partir du répertoire de travail actuel. Les chemins relatifs ne commencent jamais par **/**.

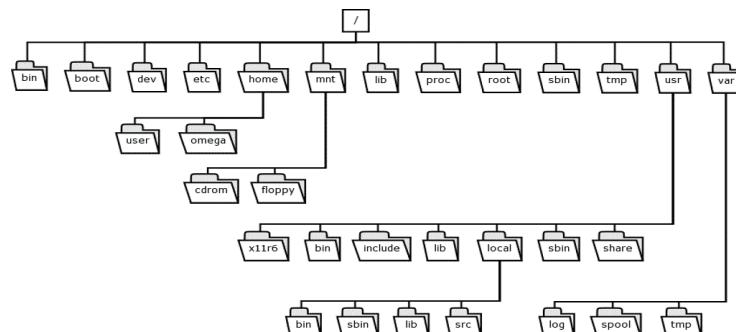


FIGURE 2.2 – Arborescence du système LINUX

Plusieurs barres obliques ” / ” entre les répertoires et les fichiers sont autorisées. Mais toutes ces barres, sauf une entre les éléments du chemin, sont ignorées par le système. Par exemple, les deux chemins, `///usr` et `//bin`, sont valides, mais considérés, respectivement, comme `/usr` et `/bin` par le système.

Il est à noter que :

- Le ” . ” signifie dossier courant
- Le ” .. ” signifie dossier parent
- Le ” ~ ” : signifie dossier personnel

La figure 2.3 présente un exemple de chemin relatif (marqué en bleu ciel) et un exemple de chemin absolu (marqué par un bleu foncé).

Pour se déplacer dans le système de fichier, d'un répertoire à un autre, la commande `cd` doit être utilisée. Utilisant cette commande, nous pouvons utiliser soit un chemin relatif, soit un chemin absolu.

#### **Exemple :**

- Chemin relatif : `cd chemin`
- Chemin absolu : `cd /chemin`

### 2.3.3 La commande ls

Syntaxe : `ls [option] [paramètre]`

La commande `ls` permet de lister les fichiers/répertoires correspondant

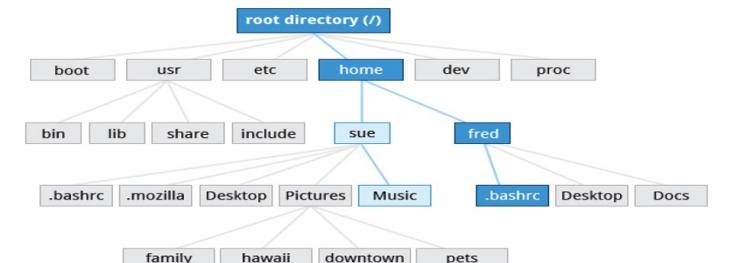


FIGURE 2.3 – Exemple de chemin relatif et de chemin absolu

au paramètre. Si aucun paramètre n'est spécifié, la commande `ls` affiche le contenu du répertoire courant.

Elle possède plusieurs options.

#### **- Exemples d'options de la commande ls :**

- `-a` : afficher tous les fichiers même ceux qui sont cachés (commençant par un point ‘.’)
- `-l` : listing étendu (format long)
- `-R` : récursif
- `-S` : tri par taille
- `-t` : tri par date de modification
- `-1` : affichage sur une seule colonne

### 2.3.4 Manipulation des répertoires et des fichiers

Pour créer des répertoires, nous utilisons la commande `mkdir`.

Syntaxe : `mkdir [option] paramètre(s)`

L'option `-p`, couramment utilisée avec la commande `mkdir`, permet de créer les répertoires parents s'ils n'existent pas.

#### **- Exemple : `mkdir Rep_1 Rep_2`**

Cet exemple permet la création des deux répertoires, Rep\_1 et Rep\_2, sous le même répertoire actuel.

Pour supprimer un répertoire vide, nous utilisons la commande **rmdir**.

Syntaxe : **rmdir paramètre(s)**

- Exemple : **rmdir Rep\_1 Rep\_2**

Nous pouvons utiliser les chemins relatif et absolu pour spécifier le répertoire à supprimer, sans prendre la peine et se déplacer dans son répertoire parent.

- Exemple :

```
rmdir /home/etudiant/rep  
rmdir toto titi
```

Pour créer des fichiers, nous utilisons la commande **touch**.

Syntaxe : **touch paramètre(s)**

Pour supprimer des fichiers ou des répertoires non vides, nous utilisons la commande **rm**.

Syntaxe : **touch [option] paramètre(s)**

**Exemples d'utilisation de la commande rm avec des options différentes :**

```
rm -r /home/etudiant/rep : supprimer un répertoire non vide  
rm -i fichier : supprimer interactivement, avec demande de confirmation  
rm -f fichier : avec force, sans demande de confirmation  
rm -rf dossier : supprime le répertoire et tout son contenu, sans confirmation
```

### 2.3.5 Copier et déplacer les fichiers

- La commande **cp** est utilisée pour copier un fichier ou un répertoire vers une destination

**Exemples d'utilisation de la commande cp avec des options différentes :**

```
cp test.txt test2.txt : copier le fichier test.txt dans le même répertoire.  
La copie doit être de nom différent, dans ce cas test2.txt.  
cp test.txt ../Bureau/test2.txt : copier le fichier test.txt dans un autre emplacement, dans cet exemple sous le répertoire /Bureau. Dans ce cas, la copie peut avoir le même nom ou un nom différent.
```

Syntaxe : **cp fichier fichier\_copie**

**cp test.txt ../Bureau/** : copier le fichier test.txt dans un autre emplacement, dans cet exemple sous le répertoire /Bureau. Dans ce cas, la copie prend le même nom que le fichier original puisque le nouveau nom n'est pas mentionné.

**cp -R** : l'option -R de la commande cp permet de lancer le processus d'une manière récursive, c'est-à-dire copier un répertoire et tout ce qu'il contient

**cp -i** : l'option -i de la commande cp permet de lancer le processus d'une manière interactive avant d'écraser chaque fichier, c'est-à-dire demander une confirmation avant la suppression.

- La commande **mv** est utilisée pour déplacer ou renommer un fichier ou un répertoire.

Syntaxe : **mv ancien\_chemin nouveau\_chemin**

**Exemples d'utilisation de la commande mv :**

```
mv test.txt test3.txt : Le fichier test.txt sera renommé en test3.txt  
mv test.txt Documents : Le fichier test.txt sera déplacé sous le répertoire Documents
```

La figure 2.4 présente un exemple d'utilisation de la commande mv pour déplacer et renommer un répertoire.

**EXEMPLE D'UTILISATION: DÉPLACER UN RÉPERTOIRE**



**EXEMPLE D'UTILISATION: RENOMMER UN RÉPERTOIRE**



FIGURE 2.4 – Exemples d'utilisation : déplacer et renommer un répertoire

### Exercice : Gestion des fichiers

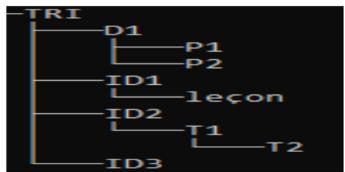


FIGURE 2.5 – Arborescence de répertoires

1. Afficher votre répertoire de travail : le résultat sera `/home/user1/TRI`
2. Créer l'arborescence mentionnée par la figure 2.5 (il est à noter qu'ils sont tous des répertoires.)
3. Dans le répertoire T2, créer en une seule commande deux fichiers nommés test1 et test2
4. Copier ces deux fichiers vers le répertoire P1
5. Déplacer le répertoire T2 vers ID1
6. Copier le répertoire P1 vers ID2.
7. Afficher le contenu du répertoire ID1 d'une façon détaillée, y compris les fichiers cachés.

### 2.3.6 Les liens

#### Rappel sur les fichiers :

- Un fichier correspond à une entrée dans le répertoire, où il se trouve.
- Un fichier est caractérisé par un nom et un numéro d'inode.
- L'inode contient les métadonnées et des pointeurs vers les blocs de contenus.

- Les permissions du fichier sont stockées dans l'inode.

Il existe deux types de liens qui permettent de rediriger un fichier vers un autre :

- **Lien physique** : permet de donner plusieurs noms/chemin d'accès, à un même fichier en pointant sur un numéro de fichier.
- **Lien symbolique** : permet d'attribuer un autre chemin d'accès à un fichier en pointant sur un nom de fichier.

La figure 2.6 illustre le principe de chacun de ces deux types de liens.

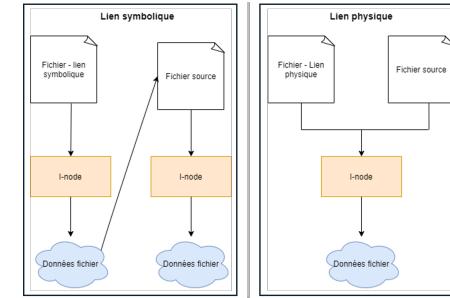


FIGURE 2.6 – Les liens symbolique et physique sous Linux

### Exercice : Les liens

1. Création du fichier sources et des liens :
  - (a) Créer un fichier source `file_1`
  - (b) Créer un lien physique `file_2` à partir de `file_1`
  - (c) Créer un lien symbolique `file_3` à partir de `file_1`
2. Modification :
  - (a) Modifier le contenu du `file_2` (lien physique)
  - (b) Observer le contenu du `file_1` et `file_3`
  - (c) Modifier le contenu du `file_3` (lien symbolique)
  - (d) Observer le contenu du `file_1`, `file_2` et `file_3`
3. Utilisation de la commande `mv`
  - (a) Créer un nouveau fichier `file4`
  - (b)Modifier le contenu du `file4`
  - (c) Renommer `file4` à `file_1`
  - (d) Visualiser le contenu des fichiers `file_1`, `file_2` et `file_3`
4. Suppression :
  - (a) Supprimer le fichier source `file_1`
  - (b) Visualiser le contenu des fichiers `file_2` et `file_3`

### 2.3.7 Les métacaractères

TABLE 2.1 – Liste des métacaractères

Métacaractère	désignation	Exemple
*	Désigne 0 ou plusieurs caractères	x*
?	Désigne un seul caractère	x? x??
[caractères]	Désigne un seul caractère de la liste des caractères	x[yz]
[!caractères]	Désigne un seul caractère en dehors de la liste des caractères	x[!yz]
[a-z]	Désigne un seul caractère appartenant à l'intervalle de caractères défini entre []	x[0-9] x[a-z]
[!a-z]	Désigne un seul caractère n'appartenant pas à l'intervalle de caractères défini entre []	x[!0-9] x[!a-z]
{a,b,c}	Désigne un seul caractère appartenant au groupe	file_{one,two,three}

#### Exemples d'utilisation des Métacaractères :

- [ ^ chars ] : remplace une classe de caractères refusée telle que liste, suite alphabétique ou suite numérique. On peut aussi utiliser [ !chars ].
- **ls file[^13579]\*[A-Z]** : lister les fichiers contenant un nombre pair et se terminant pas par une majuscule.
- {e1,e2,e3} : remplace les arguments par les éléments de la liste
- **mv file{1,2,4} /home** : les fichiers nommés : file1, file2 et file4
- {e1,e2} {e3,e4} : remplace les arguments par le produit cartésien des deux listes.
- **cp {photo,image} {.jpg,.png} /etc** : les fichiers nommés 'photo.jpg', 'photo.png', 'image.jpg' ou 'image.png'

### 2.3.8 Les expressions régulières

Les expressions régulières, appelées aussi regex, sont des motifs de texte permettant de rechercher, de correspondre ou de manipuler des chaînes de caractères. Les expressions régulières sont couramment utilisées dans diverses commandes, telles que grep, sed, egrep.

#### Recherche dans un fichier : grep

La commande grep permet de lancer une recherche de toutes les occurrences d'un terme ou d'une expression.

Syntaxe : **grep [options] expreg [fichiers]**

#### Exemples d'utilisation de la commande grep :

**grep root /etc/passwd** : cette commande permet de chercher le mot root dans le fichier passwd.

La commande grep possède plusieurs options. Dans la suite, une liste de quelques options :

- **-v** effectue la recherche inverse : toutes les lignes ne correspondant pas aux critères sont affichées.
- **-h** Ne pas afficher le nom des fichiers dans les résultats lorsque plusieurs fichiers sont parcourus.
- **-c** ne retourne que le nombre de lignes trouvées sans les afficher.
- **-i** ne différencie pas les majuscules et les minuscules.
- **-n** indique le numéro de ligne pour chaque ligne trouvée.
- **-E** Interpréter regex comme une expression régulière étendu. egrep

#### Expressions reconnues :

- . signifie un caractère quelconque
- \* répétition du caractère situé devant
- ^ début de ligne
- \$ fin d'une ligne (donc "e\$" mots se terminant par e)
- [...] contient une liste ou un intervalle de caractères cherchés
- [...] contient une liste ou un intervalle de caractères interdits.

#### Description des expressions reconnues :

##### 1. Ancrages

La commande egrep, grep -E, prend en charge des ancrages pour rechercher des motifs au début ou à la fin des lignes. Les symboles ^ et \$ indiquent respectivement le début et la fin d'une chaîne. Ils permettent donc de la délimiter.

##### Exemples d'utilisation des deux symboles ^ et \$ :

- "**^ debut**" : chaîne qui commence par "debut"
- "**fin\$**" : chaîne qui se termine par "fin"
- "**^ chaîne\$**" : chaîne qui commence et se termine par "chaîne"
- "**abc**" : chaîne contenant la chaîne "abc"

##### 2. Quantificateurs

**2.1** Les symboles \*, + et ?, désigne respectivement "zero ou plusieurs", "un ou plusieurs", "un ou aucun" caractère. Ces symboles permettent ainsi de donner une notion de nombre.

##### Exemples d'utilisation des deux symboles \*, + et ? :

- "abc+" : chaîne qui contient "ab" suivie de un ou plusieurs "c" ("abc", "abcc", ...)
- "abc\*+" : chaîne qui contient "ab" suivie de zero ou plusieurs "c" ("ab", "abc", ...)
- "abc?" : chaîne qui contient "ab" suivie de zero ou un "c" ("ab" ou "abc")
- " ^ abc+" : chaîne commençant par "ab" suivie de un ou plusieurs "c" ("abc", "abcc", ...)

**Exemple :** grep -E "abc\*" file1

**2.2** Les accolades {X,Y} permettent de donner des limites de nombre.

#### **Exemples d'utilisation :**

- "abc{2}" : chaîne qui contient "ab" suivie de deux "c" ("abcc")
- "abc{2,}" : chaîne qui contient "ab" suivie de deux "c" ou plus ("abcc" etc..)
- "abc{2,4}" : chaîne qui contient "ab" suivie 2, 3 ou 4 "c" ("abcc" .. "abcccc")

**Exemple :** grep -E "abc{2}" file1 ou bien grep "abc{2}" file1

**2.3** Le point . indique n'importe quel caractère (une fois)

#### **Exemple d'utilisation :**

- ".{3}\$" : chaîne qui contient 3 caractères

**Exemple :** grep -E ".{3\$}" file1

### **3. Utilisation de parenthèses pour grouper les motifs**

Les parenthèses () permettent de représenter une séquence de caractères.

#### **Exemple d'utilisation :**

- "a(bc)\*" : chaîne qui contient "a" suivie de zero "bc" ou plus

**Exemple :** grep -E "a(bc)\*" file1

### **4. Rechercher plusieurs motifs avec l'opérateur | :**

La barre verticale | se comporte en tant qu'opérateur OU.

#### **Exemples d'utilisation :**

- "un | le" : chaîne qui contient "un" ou "le"
- "(un | le) chien" : chaîne qui contient "un chien" ou "le chien"
- "(a | b)\*" : chaîne qui contient une suite de "a" ou de "b"

**Exemple :** grep -E "(un | le)" file1

### **5. Utilisation de classes de caractères**

Les crochets [] définissent une classe de caractères autorisés (ou interdits).

Le signe – permet quand à lui de définir un intervalle. Le caractère ^ après le premier crochet indique quand à lui une interdiction.

#### **Exemples d'utilisation :**

- "[abc]" : chaîne qui contient un "a", un "b", ou un "c"
- "[a-z]" : chaîne qui contient un caractère compris entre "a" et "z"
- "^ [a-zA-Z]" : chaîne qui commence par une lettre
- "^ [^ a-zA-Z]" : chaîne qui ne commence pas par une lettre

#### **Exercice : les expressions régulières**

Chercher les lignes répondant aux critères suivants :

1. toutes les lignes commençant par "a" ou "A".
2. toutes les lignes finissant par "rs".
3. toutes les lignes contenant au moins un chiffre.
4. toutes les lignes commençant par une majuscule.
5. toutes les lignes commençant par "B", "M" ou "Q".
6. toutes les lignes finissant par un point d'exclamation.
7. toutes les lignes ne finissant pas par un signe de ponctuation (point, virgule, point-virgule, deux-points, point d'interrogation, point d'exclamation).
8. tous les mots contenant un "r" précédé de n'importe quelle lettre majuscule ou minuscule.
9. tous les mots dont la seconde lettre est un "r".
10. Toutes les lignes contenant "nous" ou "avons"
11. Toutes les lignes contenant "nous avons"
12. Nombre de lignes contenant MICHEL ou michel

## **2.4 Les processus**

Un processus est un programme en cours d'exécution qui utilise les ressources de la mémoire et du processeur. Chaque tâche ou application, exécutée sur un système Linux, est gérée en tant que processus. Les processus jouent un rôle central dans le fonctionnement du système d'exploitation Linux, car ils permettent l'exécution simultanée de multiples tâches.

Chaque processus possède un ensemble d'informations :

- PID : Process ID
- PPID : Parent Process ID

## Objectifs

- Comprendre le fonctionnement des processus
- Savoir gérer, créer, exécuter et contrôler les processus
- Comprendre la communication inter-processus
- ajuster et réajuster la priorité des processus

## Mots clés

- ps, pstree, PID, kill, nice, renice, pipe

- User ID (UID) et Group ID (GID) : Ayant lancé le processus
- Temps CPU
- Tables de référence des fichiers ouverts

### 2.4.1 Arborescence des processus

Tous les processus sont créés par un autre processus. Le seul qui ne suit pas cette règle est le premier processus lancé : le processus **init** qui n'a pas de père et qui a pour **PID = 1**.

Une fois que le processus père lance un processus fils, il n'y aura plus de lien bien défini entre les deux.

Le processus fils devient automatiquement indépendant, s'exécute dans sa propre zone mémoire, obtient sa propre priorité et peut exister sans son parent.

Le fait d'arrêter le processus père n'arrête pas nécessairement le processus fils.

### 2.4.2 Cycle de vie du processus

Un processus passe généralement par plusieurs étapes de son cycle de vie, notamment la création, l'exécution, la suspension (mise en attente), la reprise (réactivation), la terminaison et la suppression.

- Un processus **élu** s'il est en cours d'exécution sur le processeur. Dans le cas d'une machine multiprocesseurs, plusieurs processus peuvent être élus en même temps.
- Un processus **prêt** s'il est suspendu en faveur d'un autre processus. Un processus est prêt s'il ne lui manque que la ressource processeur pour s'exécuter.
- Un processus **bloqué** s'il est en attente d'un événement externe, une ressource autre que le processeur, par exemple : bloc de disque, frappe

clavier, etc, ...

La Figure 2.7 illustre les états d'un processus dès son arrivé jusqu'à sa fin.

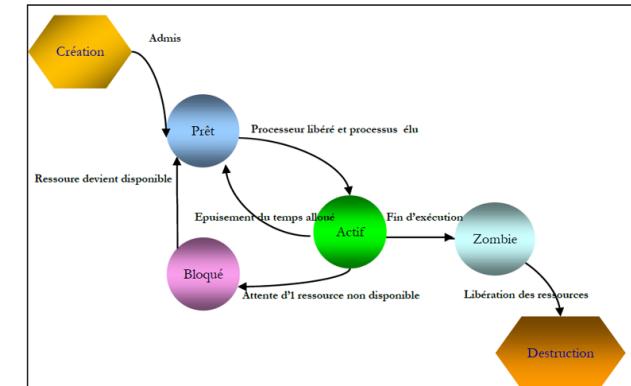


FIGURE 2.7 – Cycle de vie d'un processus

### 2.4.3 Gestion de processus

Il est possible de gérer les processus à l'aide de commandes en ligne de commande telles que **ps** (pour afficher des informations sur les processus), **kill** (pour terminer un processus), **top** (pour surveiller l'utilisation des ressources système) et d'autres outils de gestion des processus, **nice**, **renice**.

#### La commande ps

La commande **ps** permet de visualiser la liste des processus en cours d'exécution (vue statique).

- Elle peut être accompagnée de plusieurs options.
- **-u** : Affiche les processus de l'utilisateur qui exécute la commande
  - **-au** : Affiche les processus de tous les utilisateurs
  - **-aux** : Affiche l'intégralité des processus du système. Équivalent à **ps -A**

Syntaxe : **ps [options] [paramètres]**

### Exemples :

**ps -u toto** : Tous les processus de l'utilisateur toto  
**ps -u** : Tous les processus de l'utilisateur courant  
**ps -f -C vi** : affiche avec détail (option -f) le processus de la commande vi (option -C).  
**ps -ef** : affiche tous les processus, équivalente à **ps -A**.

```
ubuntu@ubuntu:~$ ps -ef
UID      PID  PPID  C STIME TTY      TIME CMD
root      1      0  0 Oct08 ?        00:00:05 /sbin/init
root      2      0  0 Oct08 ?        00:00:00 [kthreadd]
root      3      2  0 Oct08 ?        00:02:10 [ksoftirqd/0]
root      5      2  0 Oct08 ?        00:00:00 [kworker/0:0H]
root      7      2  0 Oct08 ?        00:01:07 [rcu_sched]
root      8      2  0 Oct08 ?        00:00:00 [rcu_bh]
root      9      2  0 Oct08 ?        00:00:46 [migration/0]
root     10      2  0 Oct08 ?        00:00:05 [watchdog/0]
root     11      2  0 Oct08 ?        00:00:07 [watchdog/1]
root     12      2  0 Oct08 ?        00:00:46 [migration/1]
root     13      2  0 Oct08 ?        00:00:21 [ksoftirqd/1]
root     15      2  0 Oct08 ?        00:00:00 [kworker/1:0H]
root     16      2  0 Oct08 ?        00:00:00 [khelper]
root     17      2  0 Oct08 ?        00:00:00 [kdevtmpfs]
root     18      2  0 Oct08 ?        00:00:00 [netns]
root     19      2  0 Oct08 ?        00:00:00 [khungtaskd]
root     20      2  0 Oct08 ?        00:00:00 [writeback]
root     21      2  0 Oct08 ?        00:00:00 [ksmd]
root     22      2  0 Oct08 ?        00:00:08 [khugepaged]
```

FIGURE 2.8 – Exemple de résultat de la commande ps

La Figure 2.8 présente le résultat d'exécution de la commande ps avec les options **-e** et **-f**. Un ensemble de champs est affiché. Dans la suite, nous définissons la signification de ces champs.

- **UID** nom de l'utilisateur qui a lancé le processus.
- **PID** correspond au numéro du processus.
- **PPID** correspond au numéro du processus parent.
- **C** au facteur de priorité
- **STIME** correspond à l'heure de lancement du processus
- **TTY** correspond au nom du terminal
- **TIME** correspond à la durée de traitement du processus
- **CMD** correspond au nom du processus.

### La commande pstree

La commande **pstree** permet d'afficher l'arborescence des processus. Elle est très pratique et fréquemment utilisée pour connaître de quel processus dépend un PID.

Elle peut être accompagnée de plusieurs options.  
— **-u** : afficher l'arbre des processus d'un utilisateur.  
— **-p** : d'afficher l'arbre de processus depuis un PID.

### Exemples :

**pstree -u toto** : affiche l'arbre des processus de l'utilisateur toto  
**ps -p 3635** : affiche l'arbre de processus depuis le processus d'ID=3635

```
ubuntu@ubuntu:~$ pstree
init---ModemManager---2*[{ModemManager}]
|   +--NetworkManager---dhclient
|   |   +--dnsmasq(nobody)
|   |   +--3*[{NetworkManager}]
|   +--accounts-daemon---2*[{accounts-daemon}]
|       +--acpid
|       +--avahi-daemon(avahi)---avahi-daemon
|       +--bluetoothd
|       +--colord(colord)---2*[{colord}]
|       +--cron
|       +--cupsd
|       +--cupsd
|       +--dbus-daemon(messagebus)
|       +--*[{getty}]
|       +--gnome-keyring-d(ubuntu)---6*[{gnome-keyring-d}]
|       +--irqbalance
|       +--kerneloops(kernoops)
|       +--lightdm---Xorg
|           +--lightdm---init(ubuntu)---at-spi-bus-laun---dbus-daemon
|               +--3*[{at-spl-bus-laun}]
|
|       +--at-spi2-registr---{at-spi2-registr}
|           +--bamfdaemon---3*[{bamfdaemon}]
```

FIGURE 2.9 – Exemple de résultat de la commande pstree

La Figure 2.9 présente le résultat d'exécution de la commande pstree sans aucune option. L'arborescence de l'ensemble des processus du système est affichée.

### La commande top

La commande **top** permet d'afficher des informations sur l'activité du système en temps réel.

Quelques options interactives :  
— **h** : help  
— **n** : nombre de processus à afficher

- q : quitter
- f : choisir les champs à afficher
- r : (renice) changer la priorité d'un processus

Les états affichés avec la commande top (colonne S) :

- **Exécution** (R pour Running) : le processus est en cours d'exécution ;
- **Sommeil** (S pour Sleeping) : quand il est interrompu au bout d'un quantum de temps ;
- **Arrêt** (T pour sTopped) : le processus a été temporairement arrêté par un signal (suspension du processus). Il ne s'exécute plus ;
- **Zombie** (Z pour Zombie) : le processus s'est terminé, mais son père n'a pas encore lu son code de retour.

#### 2.4.4 Envoyer un signal à un processus

L'envoi d'un signal à un processus en cours d'exécution vise à contrôler son comportement. Un signal représente un message dirigé vers un processus afin de lui spécifier l'action à prendre. Diverses actions peuvent être initiées dès l'envoi de signaux à un processus :

- **Terminer un processus** : L'émission du signal SIGTERM (signal de terminaison de numéro 15) à un processus demande à celui-ci de se terminer proprement, de manière ordonnée, permettant ainsi la libération des ressources utilisées.
- **Forcer la terminaison d'un processus** : En cas d'urgence nécessitant la cessation immédiate d'un processus qui ne répond pas aux signaux normaux de terminaison, le signal SIGKILL (numéro 9) peut être utilisé pour forcer la fin du processus, sans possibilité de fermeture propre.
- **Redémarrer un processus** : Certains processus sont configurés pour réagir au signal SIGHUP (signal de terminaison de session de numéro 1) en rechargeant ou en redémarrant leur configuration. Cette action est souvent considérée pour redémarrer des services système sans les stopper complètement.
- **Interruption d'un processus en cours d'exécution** : Le signal SIGINT est un signal de type interrupt (numéro 2). Il est généralement déclenché lors de l'appui sur les touches Ctrl+C dans le terminal. Lorsqu'un processus reçoit le signal SIGINT, il est demandé de terminer son exécution de manière propre et à libérer les ressources qu'il utilise.

La commande kill est utilisée pour envoyer un signal à un utilisateur.

Pour afficher la liste complète des noms de tous les signaux, l'option -l est utilisée avec la commande kill.

Syntaxe : `kill [numéro_du_signal] PID`

**Exemples 1 : Envoyer SIGTERM aux processus (PIDs 1000 et 1001)**

```
kill 1000 1001
kill -15 1000 1001
kill -SIGTERM 1000 1001
```

**Exemples 2 : Arrêt forcé**

```
kill -9 1000 1001
kill -KILL 1000 1001
```

#### 2.4.5 Avant-plan et arrière-plan

Lorsque vous exécutez une commande dans un terminal, celle-ci peut être lancée soit en avant-plan, soit en arrière-plan.

##### Processus en avant plan (foreground)

Un processus en avant-plan est celui pour lequel le terminal doit attendre la fin de son exécution avant de retrouver le prompt de nouveau pour entrer de nouvelles commandes. Cela signifie que le terminal est bloqué jusqu'à ce que le processus en cours soit terminé. Par exemple, si vous lancez une commande longue comme la compilation d'un programme, le terminal restera occupé jusqu'à ce que la compilation soit terminée.

##### Processus en arrière plan (background)

En ajoutant le symbole & à la fin d'une commande, le processus se lance en arrière-plan. Cela signifie que le terminal n'attend plus la fin de l'exécution du processus en cours pour retrouver le prompt de nouveau et entrer de nouvelles commandes. La commande, lancée en arrière-plan, sera exécutée tout en continuant le travail avec le terminal. Par exemple, si vous lancez une commande comme la compilation d'un programme qui prend du temps, vous pouvez la mettre en arrière-plan et continuer à utiliser le terminal pour d'autres tâches. Lorsqu'une commande est lancée en arrière-plan, le terminal affiche entre crochets le numéro de tâche (job) et le PID du processus. Le premier processus lancé en arrière-plan aura un numéro de tâche égale à 1. Cela vous permet de suivre les processus en arrière-plan et de les gérer si nécessaire. Alors, pour afficher les processus qui s'exécutent en arrière-plan, en utilise la commande jobs.

**Exemples 1 : Lancer un processus en arrière-plan et contrôler son exécution**

```
# ./firefox &
[1] 5788
# jobs
[1]+ Running ./firefox &
```

**Exemples 2 : Lancer firefox en avant-plan**

```
# ./firefox
```

Supposons que vous voulez l'envoyer en arrière-plan sans le terminer et le relancer de nouveau.

(vous appuyez **ctrl z**), le message, [1]+ Stopped ./firefox , s'affiche, indiquant que le processus est en pause

Saisir la commande **bg** pour reprendre l'exécution du processus, mais cette fois en arrière-plan.

Le message, [1]+ ./firefox &, s'affiche, indiquant que le processus est en cours d'exécution en arrière-plan.

#### 2.4.6 Modifier les priorités d'exécution de processus

Dans un système d'exploitation, chaque processus possède un degré de priorité d'exécution, indiquant son importance par rapport à d'autres processus en cours d'exécution sur le même système. La priorité par défaut attribuée à un processus lors de sa création par le système d'exploitation est égale à 0. Cette priorité peut être modifiée pour ajuster la manière dont le système d'exploitation attribue des ressources de traitement à différents processus. Pour ajuster en augmentant ou en diminuant la priorité d'un processus par rapport à la valeur par défaut, la commande "nice" peut être utilisée. La modification de la priorité d'exécution d'un processus déjà en cours d'exécution est aussi possible en utilisant la commande "renice". Cette commande permet d'ajuster dynamiquement les priorités des processus en fonction des besoins changeants du système.

Le niveau de priorité varie entre -20 et 19. Les valeurs positives correspondent aux processus moins prioritaires, tandis que les valeurs négatives correspondent aux processus les plus prioritaires. Les valeurs négatives sont réservées uniquement au super utilisateur (ROOT).

Syntaxe : nice < Priorité > commande

**Exemples 1 : Un utilisateur lance une commande 'cmd' avec le niveau de priorité +16**

```
$ nice -16 cmd
```

```
$ nice -n 16 cmd
```

**Exemples 2 : Le root lance la commande 'vi' avec le niveau de priorité -10**

```
# nice - -10 vi /etc/hosts.deny
# nice -n -10 vi /etc/hosts.deny
```

Syntaxe : renice < Priorité > {-p <PID> | -g <GID> | -u <User>}

**Exemples 3 :Modifier la priorité d'un processus lorsque le processus est déjà en cours d'exécution**

```
#renice -20 501
#renice -10 -u salah -p 501
```

#### 2.4.7 Les tubes et les redirections

Un processus fait recours à trois canaux, Figure 2.10, pour communiquer avec l'utilisateur et pour gérer efficacement les données d'entrée et de sortie. Ces canaux sont connus sous les noms de "stdin", "stdout" et "stderr".

##### L'entrée standard : stdin

Le stdin (Standard Input), identifiée par le descripteur 0, est le canal à partir duquel un processus récupère des données en entrée. Le périphérique d'entrée standard est le clavier. Il est utilisé par l'utilisateur pour saisir des données dans le terminal. En revanche, le processus peut lire les données à partir d'un fichier ou un autre processus, ce qui nécessite une redirection du canal d'entrée.

##### La sortie standard : stdout

Le stdout (Standard Output), identifiée par le descripteur 1, est le canal vers lequel un processus envoie ses données. Le périphérique de sortie standard est l'écran du terminal, c'est pourquoi par défaut, les données envoyées à stdout sont affichées à l'utilisateur dans le terminal. Les programmes font recours au canal stdout pour afficher leurs résultats, des messages et toutes autres informations utiles à l'utilisateur. Ce canal de sortie peut être redirigé vers un fichier pour sauvegarder les résultats plutôt de les afficher au terminal.

## La sortie standard : stderr

Le stderr (Standard Error), identifiée par le descripteur 2, est le canal utilisé par un processus pour envoyer des messages d'erreur, des avertissements et des informations de diagnostic signalant des problèmes. Tout comme stdout, stderr est par défaut associé à l'écran du terminal, mais il peut être redirigé vers un fichier, comme les fichiers log.

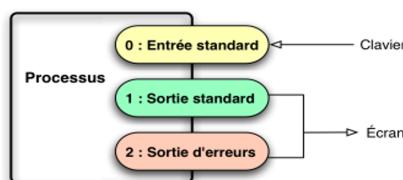


FIGURE 2.10 – Les canaux standards de lecture et d'écriture

## Redirections des entrées / sorties

- **cmd < fichier** : redirection de l'entrée standard du fichier. Si le fichier n'existe pas, le shell renvoie une erreur.  
**Exemple :** tr a-z A-Z < fichier.txt
- **cmd > fichier** : redirection de la sortie standard de la commande vers le fichier. Si le fichier existe, il sera écrasé sinon créé.  
**Exemple :** cat bonjour > Toto.txt
- **cmd >> fichier** : redirection de la sortie standard de la commande vers la fin du fichier sans écraser son ancien contenu.
- **cmd >& fichier** : rediriger la sortie standard et la sortie standard d'erreur vers un fichier.
- **cmd 2> fichier** : rediriger seulement la sortie standard d'erreur vers le fichier.

## Les tubes (pipes)

Un pipe, "tube" en français, est un mécanisme de communication inter-processus, qui permet de connecter la sortie standard (stdout) d'une commande à l'entrée standard (stdin) de la commande suivante. Ce mécanisme facilite l'enchaînement de plusieurs commandes ensemble, de sorte que la

sortie d'une commande devient l'entrée de la suivante, facilitant ainsi le traitement de données dans un flux continu. Ce mécanisme est illustré par la Figure

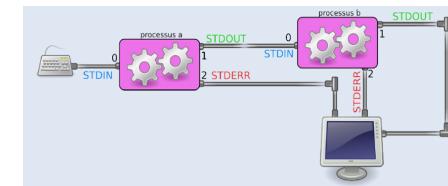


FIGURE 2.11 – Le mécanisme des tubes

### Exemples 1 : ls -al | wc -l

**Etape 1 :** la commande **ls** crée une liste de fichiers qui est envoyée vers la sortie standard.

**Etape 2 :** Cette sortie est dirigée vers la commande **wc** qui compte le nombre de lignes dans cette liste.

**Etape 3 :** Le résultat est transmis vers la sortie standard (écran).

### Exemples 2

ls | grep win | wc -l  
who | tee fichier | wc -l

### Exercice : les tubes

Créer le répertoire 'essai-grep', ensuite les fichiers 'tomate' 'poire' 'pomme' 'cerise' 'Fraise' 'fraise' 'courgette' 'POMME3' et 'afraise'.

1. Afficher les fichiers dont le nom commence par F ou f
2. Afficher les fichiers dont le nom se termine par se
3. Afficher les fichiers dont le nom contient ai
4. Afficher les fichiers dont le nom contient un chiffre numérique
5. Recherchez dans /etc/passwd la ligne correspondant à votre utilisateur

## 2.5 Éditeur VI

Objectifs	- effectuer des manipulations simples du contenu d'un fichier texte en utilisant vi - naviguer dans un fichier utiliser les modes vi de base - Insérer, éditer, copier et supprimer du texte dans un fichier - trouver du texte dans un fichier
-----------	--

Mots clés	vi
-----------	----

L'éditeur "vi" est un éditeur de texte conçu par Bill Joy en 1976 depuis les toutes premières versions d'Unix et Linux. Il offre une gamme étendue de fonctionnalités et de commandes pour l'édition de texte. Il propose trois modes de fonctionnement :

1. **Mode commande** : les saisies représentent des commandes. On y accède en appuyant sur [Echap]. Chaque touche ou combinaison de touches déclenche une action (suppression de lignes, insertions, déplacement, copier, coller, etc.).
2. **Mode saisie** : c'est la saisie de texte classique.
3. **Mode ligne de commande** : une ligne en bas d'écran permet de saisir des commandes spéciales, validée avec la touche [Entrée]. On y accède en appuyant, en mode commande, sur la touche ":".

### 2.5.1 La saisie

Les actions suivantes sont à effectuer en mode commande. Elles doivent être précédées d'un appui sur [Echap], par exemple [Echap] a, [Echap] i, etc. Le Tableau 2.2 présente les commandes à utiliser avec l'éditeur "vi" pour saisir un texte dans un fichier.

### 2.5.2 Quitter et sauver

les ":" signifient que la commande se tape en ligne de commande, par exemple [Echap] :, saisie de la commande, puis [Entrée]. Le Tableau 2.3 présente les commandes à utiliser avec l'éditeur "vi" pour sauvegarder un texte dans un fichier et quitter.

TABLE 2.2 – Les commandes de saisie

Commande	Action
a	Ajout après le caractère actuel
A	Ajout de texte en fin de ligne
i	Insertion devant le caractère actuel, comme dans un traitement de texte
I	Insertion de texte en début de ligne
o	Ajout d'une ligne sous la ligne actuelle
O	Insertion d'une ligne au-dessous de la ligne actuelle

TABLE 2.3 – Les commandes pour sauvegarder et quitter

Commande	Action
ZZ	Sauve le fichier et quitte
:q!	Quitte sans sauvegarde
:q	Quitte si le fichier n'a pas été modifié (apparition d'un message d'erreur sinon)
:w	Sauve le fichier
:wq ou :x	Sauve le fichier et quitte
1,10w fic	Sauve les lignes de 1 à 10 dans fic

### 2.5.3 Correction

Le Tableau 2.4 présente les commandes à utiliser avec l'éditeur "vi" pour corriger un texte dans un fichier.

TABLE 2.4 – Les commandes de correction

Commande	Action
x	Efface le caractère sous le curseur
X	Efface le caractère devant le curseur
r<c>	Remplace le caractère sous le curseur par le caractère <c>
dw	Efface depuis le curseur jusqu'à la fin du mot
d\$ ou D	Efface depuis le curseur jusqu'à la fin de la ligne

#### 2.5.4 Recherche

Contrairement à un éditeur de texte classique, vi peut rechercher autre chose que des mots simples et fonctionne à l'aide de caractères spéciaux et de critères. La commande de recherche est le caractère **/**.

##### Exemples

- **/[FfBb]oule** : Foule, foule, Boule, boule.
- **/[A-Z]e** : tout ce qui commence par une majuscule avec une en deuxième position.
- **/[A-Z a-z 0-9]** : tout ce qui commence par une majuscule, une minuscule ou un chiffre.
- **/[^ a-z]** : plage négative : tout ce qui ne commence pas par une minuscule.

#### 2.5.5 Copier-Coller

La commande **v** permet une sélection visuelle. Le texte est surligné et vous pouvez déplacer le curseur pour sélectionner le texte. Utilisez ensuite l'une des commandes suivantes :

- pour couper (déplacer), c'est la commande « **d** »
- pour coller le texte à l'endroit choisi, c'est la commande **p** (derrière le caractère) ou **P** (devant le caractère).

Si c'est une ligne complète qui a été copiée, elle sera placée en dessous de la ligne active. Les actions suivantes sont possibles en mode commande :

- Pour copier une ligne : **yy**.
- Pour copier cinq lignes : **5yy**.
- Pour placer les lignes copiées à un endroit donné : **p**.

# Chapitre 3

## Gestion des droits d'accès

### Contenu du chapitre 3

- Introduction
- L'organisation des utilisateurs sous Linux
- Utilisateurs et permissions
- Gestion des utilisateurs
- Gestion des groupes
- Les droits spéciaux et utilisation

### 3.1 Introduction

Les droits d'accès définissent la possession d'un fichier ou d'un répertoire à un utilisateur et à un groupe d'utilisateurs. Ils gèrent les actions que les utilisateurs ont droit d'effectuer sur les fichiers. La gestion de ces droits d'accès est cruciale pour la sécurité des systèmes d'exploitation, régisse les actions autorisées telles que la lecture, l'écriture et l'exécution de fichiers. Dans ce chapitre, les mécanismes de contrôle d'accès aux fichiers, répertoires et ressources système sur un système Linux sont explorés. En assimilant comment les droits d'accès sont attribués, gérés et ajustés, les administrateurs système sont en mesure d'assurer la confidentialité, l'intégrité et la disponibilité des données, tout en garantissant un accès adéquat aux utilisateurs et aux processus. Ce chapitre examinera en détail les concepts fondamentaux tels que les permissions de fichier, les propriétaires et les groupes, ainsi que les commandes et les outils utilisés pour configurer et vérifier les droits d'accès sur un système Linux.

### 3.2 Organisation des utilisateurs sous Linux

Dans un système d'exploitation, les utilisateurs sont répartis en plusieurs catégories en discriminant entre les utilisateurs réguliers et le superutilisateur, ainsi que sur l'utilisation de groupes d'utilisateurs. Cette organisation simplifie la gestion des autorisations. La figure 3.1 présente une vue d'ensemble de l'organisation des utilisateurs sous Linux.

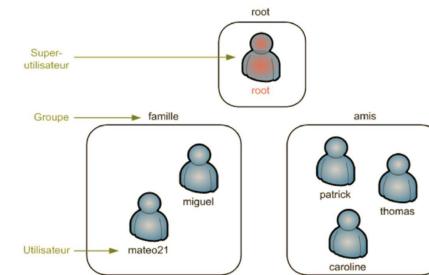


FIGURE 3.1 – Organisation des utilisateurs sous Linux

### 3.3 Utilisateurs et permissions

Sous Linux, on distingue trois catégories d'utilisateurs :

1. **u** : le propriétaire (user), Désigne la personne qui a créé le fichier/répertoire, c'est le propriétaire
2. **g** : le groupe, Désigne les membres du groupe d'utilisateurs
3. **o** : les autres (others), Désigne tous les autres utilisateurs

On distingue également trois types de droits :

1. **r** : lecture (read)
2. **w** : écriture (write)
3. **x** : exécution
4. **-** : aucun droit

La visualisation des permissions sur les fichiers et les répertoires est possible utilisant la commande **ls -l**, tel qu'il est illustré par le tableau 3.1.

Le type de fichier est représenté par "d" pour les répertoires, "-" pour les fichiers et "l" pour les liens symboliques (raccourci).

TABLE 3.1 – Les permissions sur les fichiers et les répertoires

Type	Propriétaire (U)	Groupe (G)	Autres (O)	Nom propriétaire	Groupe	Nom fichier
d	r w x	r - x	r - x	ubuntu	prof	cours
-	r w x	r w x	r w x	ubuntu	prof	img.jpg
-	r w x	- - -	- - -	ubuntu	prof	tp-unix

### 3.3.1 Sémantique des permissions

La sémantique des permissions n'est pas la même pour les fichiers et les répertoires comme il est illustré par le tableau 3.2.

TABLE 3.2 – Les permissions sur les fichiers et les répertoires

Fichier		
r	w	x
Autorise la consultation Affichage copie changement de nom	Autorise la modification modification du contenu suppression du fichier	Autorise l'exécution le fichier doit être un programme
Répertoire		
Autorise la consultation affichage du contenu (ls)	Autorise la modification modification du nom (mv) ajout/suppression du contenu (rm) suppression du répertoire (rm)	Autorise la traversée utilisation dans un chemin d'accès à une entrée (fichier/répertoire) (cd)

### 3.3.2 Chmod : change mode

La modification des permissions accordées sur les fichiers et répertoires est réalisée par la commande chmod. Cette commande est utilisée dans les systèmes Unix et Linux pour contrôler ainsi les droits de lecture, écriture et exécution pour différents utilisateurs et groupes. Il existe deux méthodes principales pour utiliser la commande chmod : l'écriture symbolique et l'écriture en octal.

#### Chmod : Ecriture symbolique

Les modifications peuvent être apportées à un, deux ou les trois catégories d'utilisateurs. On utilise le symbole correspondant à la catégorie user (u), groupe (g) ou autres (o) pour ajouter (+), enlever (-) ou fixer (=) les droits

de lecture (r), d'écriture (w) ou d'exécution (x). Une permission même permission peut être ajoutée à la fois à toutes les catégories en utilisant le symbole (a) qui signifie all (ugo), par exemple, **chmod +r = chmod a+r**  
**= chmod ugo+r.**

Syntaxe : **chmod <qui(u g o) action(+ - =) droit(r w x)> fichier/rep**

#### - Exemple :

**chmod ugo+rwx fichier1 fichier2**

**chmod u+r, g+r, o=rwx fichier3**

**chmod ug=rwx, o=rx fichier4**

#### Chmod : Écriture en Octal

Cette méthode utilise des chiffres pour représenter les différents droits d'accès. Chaque droit (lecture, écriture, exécution) est associé à un chiffre spécifique comme suit :

— r = 4 → (r - - : 100 en binaire)

— w = 2 → (- w - : 010 en binaire)

— x = 1 → (- - x : 001 en binaire)

— - = 0 → (- - - : 000 en binaire)

Les chiffres sont ensuite combinés pour définir les droits d'accès.

#### - Exemple 1 : modification des permissions sur un fichier

— Pour r w x, on aura :  $4+2+1 = 7$

— Pour r w -, on aura :  $4+2+0 = 6$

— Pour r - , on aura :  $4+0+0 = 4$

→ La commande "chmod 764 file" accorde la lecture, l'écriture et l'exécution au propriétaire, la lecture et l'écriture au groupe et la lecture aux autres sur le fichier 'file'.

#### - Exemple 2 : modification des permissions sur un répertoire

Les nouvelles permissions à accorder à un répertoire Rep sont : **rwx r-x**

---  
La valeur correspondante en octal soit :

— Pour r w x, on aura :  $4+2+1 = 7$

— Pour r - x, on aura :  $4+0+1 = 5$

— Pour - - -, on aura :  $0+0+0 = 0$

→ La commande "chmod 750 Rep" accorde tous les droits (lecture, écriture et exécution) au propriétaire du répertoire 'Rep', les

droits de lecture et d'exécution au groupe et aucun droit aux autres utilisateurs sur le répertoire 'Rep'.

Ajoutant l'option **-R** à la commande **chmod** précédente permet de lancer la modification des droits d'accès **récursivement** sur un répertoire. Cela implique que les modifications seront appliquées non seulement au répertoire spécifié, mais également à tous les fichiers et sous-répertoires qu'il contient, assurant ainsi une cohérence dans la gestion des permissions à travers toute la structure de répertoires.

#### - Exemple 3 : modification des permissions sur un répertoire récursivement

```
chmod -R 750 Rep
```

### 3.3.3 Chown : change owner

La commande **chown** (**c**hange **o**wner, changer le propriétaire) permet de changer le propriétaire et/ou le groupe d'un fichier ou d'un répertoire.

Syntaxe : **chown** propriétaire : (groupe) fichier/rep

L'utilisation de la commande chown est restreint à des utilisateurs spécifiques. Seuls le super-utilisateur (root) et le propriétaire du fichier sont autorisés à exécuter cette commande. Cette restriction vise à assurer la sécurité du système.

#### - Exemple 1 : modification du propriétaire du fichier 'file' par le super-utilisateur

```
sudo chown Ali file
```

#### - Exemple 2 : modification du propriétaire et du groupe du fichier 'file' par le super-utilisateur

```
sudo chown Ali :Info file
```

### 3.3.4 Chgrp : change group

La commande **Chgrp** (**c**hange **g**roup, changer le groupe) permet de changer le groupe d'un fichier ou d'un répertoire.

Syntaxe : **chgrp** groupe fichier/rep

L'utilisation de la commande chgrp est restreint à des utilisateurs spécifiques. Seuls le super-utilisateur (root) et le propriétaire actuel du fichier sont auto-

risés à exécuter cette commande. Cette restriction vise à assurer la sécurité du système.

#### - Exemple 1 : modification du groupe du fichier 'file' par le propriétaire du fichier

**chgrp Elect file** : le fichier 'file' appartient maintenant au groupe 'Elect'. Tous les membres du groupe 'Elect' auront accès à ce fichier selon les permissions du groupe.

### 3.3.5 umask : user mask

Le **umask** (masque utilisateur) est utilisé pour définir les droits d'accès par **défaut** attribués à un fichier ou un répertoire lors de sa création. Ainsi, lorsqu'un fichier est créé, les droits d'accès par défaut sont de 666 (valeur maximale des droits d'accès : rw- rw- rw-) auxquels la valeur de umask est soustraite. Pour les répertoires, les droits d'accès par défaut sont de 777 (valeur maximale des droits d'accès : rwx rwx rwx) auxquels la valeur de umask est soustraite également.

Ainsi, le umask se présente comme un filtre pour les droits d'accès par défaut, en spécifiant quelles permissions ne seront pas accordées lors de la création de fichiers ou de répertoires.

#### - Exemple 1 : Pour un fichier

**umask 022** : À partir des droits maximum 666, on retranche 022, on obtient donc 644

→ Par **défaut**, les fichiers créés auront comme droit **644** (rw- r- r-)

#### - Exemple 2 : Pour un répertoire

**umask 022** : À partir des droits maximum 777, on retranche 022, on obtient donc 755

→ Par **défaut**, les répertoires créés auront comme droit **755** (rwx r-x r-x)

**umask** peut être utilisé comme commande Linux pour afficher sa valeur actuel considérée par le système, ou pour modifier sa valeur (**\$umask 422**).

### Exercice : les droits d'accès

1. Créez un répertoire **Ex\_droit**. Les droits d'accès par défaut sur ce répertoire sont à 755.

Quelles sont les commandes (**en notation symbolique et en octal**) pour lui donner les droits mentionnés dans le tableau 3.3. Il est à noter que les commandes sont indépendantes.

2. Créez un fichier **file\_droit** dans le répertoire **Ex\_droit**. Les droits d'accès par défaut sur ce fichier sont à 644.

Supposons que vous êtes positionnés dans le répertoire parent du répertoire **Ex\_droit**, pour chaque commande de la question précédente, essayez :

- D'accéder au répertoire **Ex\_droit**,
- De lister le contenu de **Ex\_droit**,
- De modifier le fichier **file\_droit**.

TABLE 3.3 – Liste des permissions pour le répertoire Ex\_droit

	user			group			others		
	r	w	x	r	w	x	r	w	x
cmd1	oui	oui	oui	oui	non	oui	non	non	oui
cmd2	oui	non	oui	non	oui	non	non	non	oui
cmd3	non	oui	non	non	non	oui	oui	non	non
cmd4	non	non	oui	oui	non	oui	non	non	non

## 3.4 Gestion des utilisateurs

Dans un système d'exploitation Linux, la gestion des utilisateurs est une tâche cruciale qui est réservée au super-utilisateur. Seul le super-utilisateur est autorisé à effectuer des opérations telles que l'ajout, la suppression ou la modification d'un utilisateur.

La commande principale utilisée pour ajouter un nouvel utilisateur est **useradd**. Cette commande permet de créer un nouvel utilisateur sans créer automatiquement son répertoire personnel. L'option **-m** peut être ainsi utilisée avec la commande **useradd** pour spécifier la création du répertoire personnel de l'utilisateur lors de la création du compte utilisateur.

Syntaxe : **useradd [options] nomUser**

Une liste d'options est réservée à la commande useradd :

- **-m** Crée un répertoire personnel
- **-s** Définit le shell pour l'utilisateur
- **-e** Date à laquelle le compte de l'utilisateur sera désactivé
- **-b** Répertoire de base du répertoire de base de l'utilisateur

- **-u** UID
- **-g** Numéro de groupe initial
- **-G** Groupes supplémentaires par nom
- **-c** Commentaire

Une autre commande **adduser** peut être aussi utilisée pour ajouter un utilisateur. Un répertoire personnel du nouvel utilisateur (nomUser) est automatiquement créé sous le répertoire **HOME** (/home/NomUser).

Syntaxe : **adduser nomUser**

Les commandes **adduser** et **useradd** sont toutes deux des commandes utilisées pour la gestion des utilisateurs sous le système d'exploitation Linux. La principale différence entre les deux réside dans leur niveau d'abstraction et leur fonctionnement. **adduser** est une commande de plus haut niveau conçue pour simplifier le processus d'ajout d'utilisateurs. Elle prend en charge la configuration automatique du dossier personnel de l'utilisateur et d'autres paramètres à partir du fichier de configuration /etc/adduser.conf. D'un autre côté, **useradd** est une commande de niveau inférieur qui offre plus de contrôle et de flexibilité. Elle est utilisée pour ajouter des utilisateurs au système sans aucune configuration automatique. Les administrateurs doivent spécifier manuellement tous les paramètres nécessaires, tels que le répertoire personnel de l'utilisateur, les groupes supplémentaires, etc. Bien que moins conviviale que adduser, useradd offre une plus grande personnalisation pour les cas où un contrôle précis est nécessaire.

Pour supprimer un utilisateur, la commande **userdel** est utilisée. Cette commande supprime le compte d'un utilisateur et les fichiers associés. L'option **-r** peut être utilisée pour supprimer le répertoire personnel de l'utilisateur lors de sa suppression.

Syntaxe : **userdel [-r] nomUser**

Les informations utilisateur sont stockées dans les fichiers :

- /etc/passwd : Informations utilisateur de base
- /etc/shadow : Informations sur le mot de passe utilisateur
- /etc/group : Informations de base sur le groupe d'utilisateurs
- /etc/gshadow : Informations sur le groupe d'utilisateurs
- /home/nom d'utilisateur : Fichiers personnels de l'utilisateur par défaut et
- /var/spool/mail/nom d'utilisateur : la boîte aux lettres

## 3.5 Gestion des groupes

Seule le super-utilisateur peut ajouter/supprimer/modifier un groupe. La commande **addgroup** permet d'ajouter un groupe.

Syntaxe : **addgroup nomGroupe**

La commande **usermod** permet de modifier un utilisateur en l'ajoutant à un groupe.

Syntaxe : **usermod -g nomGroupe nomUser**

Un utilisateur peut appartenir à plusieurs groupes. Pour ce faire, utilisez le paramètre **-G** (majuscule).

Syntaxe : **usermod -G grp1.grp2.grpN nomUser**

Pour ajouter des groupes à un utilisateur sans perdre les groupes auxquels il appartenait avant cela, utilisez l'option **-a**.

- Exemple

**usermod -aG amis,paris,collegues patrick**

La commande **delgroup** permet de supprimer un groupe.

## 3.6 Les droits spéciaux et utilisation

Parallèlement aux droits standards, des droits étendus, SUID, SGID, Sticky bit, existent et répondent à des besoins spécifiques :

- **SUID : (Set User ID)** : Lorsque ce bit est positionné, une commande peut être lancée par un autre utilisateur avec l'UID de son propriétaire (exemple : **passwd**)
- **SGID : (Set Group ID)** : idem que SUID, une commande peut être lancée par un autre utilisateur avec le GID de son propriétaire. Quand le SGID est assigné à un répertoire, tout fichier créé dans ce répertoire appartiendra au même groupe propriétaire de ce répertoire
- **Sticky bit (bit collant)** : Si ce bit est placé sur un dossier, seul le propriétaire d'un fichier pourra le renommer ou le supprimer. (exemple : **/tmp**)

La commande **chmod** permet de placer le sticky bit, SUID-Bit, GUID-Bit.

Syntaxe : **delgroup nomGroupe**

TABLE 3.4 – Les droits spéciaux

Droit	Octale	Symbolique
Stiky bit	1000	o+t
SUID	4000	u+s
GUID	2000	g+s

### 3.6.1 Le stiky bit

Le droit **Sticky Bit** (appelé également bit collant) permet d'interdire à tout utilisateur (sauf le super-utilisateur) de renommer/supprimer un fichier dont il n'est pas le propriétaire, quels que soient ses droits.

**Exemple :** Dans le répertoire **/tmp** tous les utilisateurs ont le droit de lire et d'écrire (supprimer) des fichiers.

Appliquant le sticky bit : **chmod 1755 tmp/** ou **chmod o+t tmp/**. Ce droit s'affiche en lieu et place du droit en exécution de la catégorie autres

### 3.6.2 SUID

En exécutant un programme possédant un droit SUID, un utilisateur s'approprie les droits du propriétaire du fichier exécutable durant le temps d'exécution du programme.

L'exemple le plus flagrant est celui du programme **/usr/bin/passwd** qui appartient à l'utilisateur root. Comment donc modifier votre mot de passe si vous n'êtes pas autorisé à écrire sur le fichier stockant les mots passés ? !!

**/usr/bin/passwd** vous y autorise grâce au droit SUID : vous utilisez le droit en écriture du root durant l'exécution du programme qui change votre mot de passe.

**Remarque :** la présence du droit SUID suppose la présence du droit en exécution qui permet de lancer le fichier exécutable.

#### Notation symbolique

Syntaxe : **chmod u+s fichier-exécutable**

- Exemple : **chmod u+s monprog**

## Notation numérique

Syntaxe : **chmod 4+droits fichier-exécutable**

- Exemple : **chmod 4755 monprog**

Dans cet exemple, on donne les droits **rws r-x r-x** au fichier **monprog**.

Ne pas oublier que le droit SUID s'affiche à la place du droit d'exécution du propriétaire sans que ce dernier soit supprimé ! Si la SUID bit est défini sur un fichier qui n'a pas de capacités exécutables, un "S" sera positionné.

## 3.6.3 GUID

### Notation symbolique

Syntaxe : **chmod g+s fichier-exécutable**

- Exemple : **chmod g+s monprog**

## Notation numérique

Syntaxe : **chmod 2+droits fichier-exécutable**

- Exemple : **chmod 2755 monprog**

Dans cet exemple, on donne les droits **rwx r-s r-x** au fichier **monprog**.

Ne pas oublier que le droit GUID s'affiche à la place du droit en exécution du groupe sans que ce dernier soit supprimé !

### Exercice : Gestion des utilisateurs

- Création de 4 utilisateurs, u1, u2, u3 et u4, et de 2 groupes : projet1, projet2.
- Les deux premiers utilisateurs, u1 et u2, sont membres du groupe projet1.
- Les deux derniers utilisateurs, u3 et u4, sont membres du groupe projet2.
- L'utilisateur, u2, est aussi membre du groupe projet2.
- L'utilisateur, u4, est aussi membre du groupe projet1.

- Outre leur répertoire de travail /home/projet1 et /home/projet2, les utilisateurs ont accès à un répertoire commun /home/Rep. Dans ce répertoire, ils peuvent écrire, créer des fichiers, mais ne peuvent pas effacer les fichiers.
- Question : Détaillez les étapes de création des groupes, des utilisateurs et des répertoires en indiquant quels fichiers vous utilisez et quelles commandes vous utilisez sur ces fichiers.

## Webography

- [1] *Historique et liste des versions de Windows.* URL : <https://www.malekal.com/historique-versions-windows/#Historique%20et%20Liste%20Des%20Versions%20de%20Windows%20Depuis%20Les%20Ann%C3%A9es%202080> (visité le 0004-2023).
- [2] *MAC OS X ET MACOS : TOUTES LES VERSIONS PUBLIÉES JUSQU'À PRÉSENT.* URL : <https://frenchmac.com/actualite/mac-os-x-macos-versions-publiees-present/> (visité le 0011-2022).
- [3] *Évolution de Linux.* URL : <https://linux.goffinet.org/administration/introduction-a-linux/evolution-de-linux/> (visité le 0009-2020).
- [4] *Les systèmes d'exploitations les plus populaires 2004-2019.* URL : [https://www.youtube.com/watch?v=Km9zf5MwQgQ&ab\\_channel=J%27aimeleclassement](https://www.youtube.com/watch?v=Km9zf5MwQgQ&ab_channel=J%27aimeleclassement) (visité le 0009-2020).
- [5] *Richard Stallman's Personal Site.* URL : <https://stallman.org/> (visité le 0009-2023).
- [6] *Le projet GNU.* URL : <https://www.gnu.org/gnu/thegnuproject.fr.html> (visité le 0009-2023).
- [7] *25 ième anniversaire de KDE.* URL : <https://25years.kde.org/fr/> (visité le 0009-2023).
- [8] *KDE.* URL : <https://kde.org/fr/announcements/> (visité le 0009-2023).
- [9] *gnome.* URL : <https://www.gnome.org/> (visité le 0009-2023).
- [10] *2023's Ultimate Linux Distro Guide - The Top 7 You Can't Miss! (NEW).* URL : <https://www.youtube.com/watch?v=qd6FibkktRM> (visité le 0009-2023).
- [11] *Les 10 distributions Linux incontournables de 2024.* URL : <https://www.clubic.com/telecharger/actus-logiciels/article-842846-1-10-distributions-gnu-linux-preferees-dire-adieu-windows-10.html> (visité le 0009-2023).
- [12] *Les 10 meilleures distributions Linux pour 2024.* URL : <https://www.justgeek.fr/les-meilleures-distributions-linux-107294/> (visité le 0009-2023).
- [13] *ubuntu.* URL : <https://www.ubuntu-fr.org/> (visité le 0009-2021).
- [14] *Create a bootable USB stick with Rufus on Windows.* URL : <https://tutorials.ubuntu.com/tutorial/tutorial-create-a-usb-stick-on-windows#0> (visité le 0009-2023).
- [15] *How to install Linux on Windows with WSL.* URL : <https://docs.microsoft.com/en-us/windows/wsl/install-win10> (visité le 0009-2023).
- [16] *VirtualBox.* URL : <https://www.virtualbox.org/wiki/Downloads> (visité le 0009-2023).
- [17] *Comment installer Ubuntu 22.10 LTS sur Virtualbox en 2023.* URL : [https://www.youtube.com/watch?v=P1Jx0XpyILU&ab\\_channel=UNEHEURELINUX](https://www.youtube.com/watch?v=P1Jx0XpyILU&ab_channel=UNEHEURELINUX) (visité le 0009-2023).

## Bibliography

- [18] Sébastien ROHAUT. "Linux, préparation à la certification LPIC-1". In : *Certifications, ENI Edition.* 2014.
- [19] A. Boyanov N. Larrousse Z. BOUZIRI N. H. Andriambelo. "Préparation à l'examen 101 pour la certification de l'Institut professionnel de Linux, niveau junior (LPIC-1)". In : *Agence universitaire de la Francophonie, Paris.* 2010.