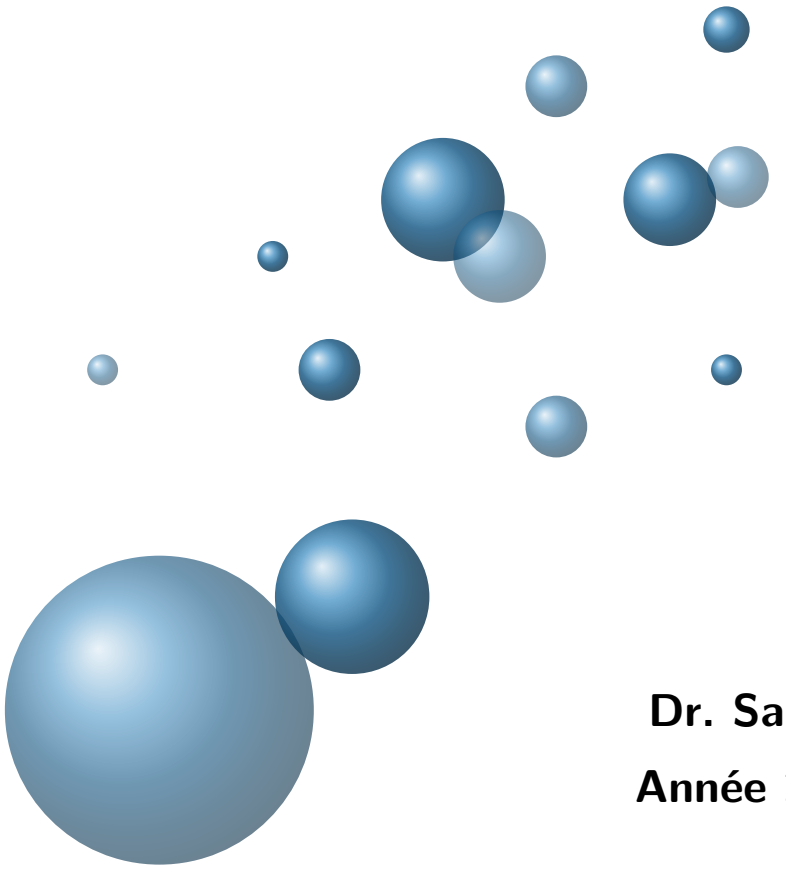
République Algérienne Démocratique et Populaire
Ministère de l'Enseignement Supérieur et de la Recherche Scientifique
Faculté des Sciences Exactes
Département d'Informatique

جامعة بجاية
Tasdawit n Bgayet
Université de Béjaïa

Destiné pour Licence II informatique

Travaux pratiques sur le système d'exploitation Linux



Dr. Salima SABRI

Année 2024 – 2025

AVANT-PROPOS

Ce document a été élaboré dans le cadre des travaux pratiques (TP) de deuxième année de licence en informatique. Il a pour objectif de guider les étudiants dans l'apprentissage et la compréhension du fonctionnement du système Linux ainsi que dans l'initiation à la programmation sur cet environnement.

Les travaux pratiques sont un élément essentiel de la formation en informatique, car ils permettent de mettre en application les théories étudiées et de développer des compétences pratiques indispensables pour tout futur informaticien. Ce rapport a été rédigé dans le but de structurer et de formaliser les résultats obtenus lors des séances de TP, tout en encourageant une démarche méthodique et rigoureuse.

Objectifs du rapport

Ce rapport vise à :

- Naviguer et interagir avec un système Linux en ligne de commande
- Gérer efficacement les fichiers, répertoires, utilisateurs et permissions
- Surveiller et contrôler les processus système
- Automatiser des tâches simples grâce à des scripts Shell

Structure du document

Le rapport est organisé de la manière suivante :

- Prise en main de l'environnement Linux
- Manipulation de dossiers et de fichiers
- Utilisateurs, groupes et permissions
- Manipulation des processus-
- Initiation à la Programmation Shell

Table des matières

Table des figures

1 TP. Prise en main de l'environnement Linux

Sommaire

1.1 Concepts généraux

1.1.1 Qu'est-ce qu'un système d'exploitation ?

Un système d'exploitation se dit Operating System en anglais, que l'on abrège en (OS) est un ensemble de programmes qui gère les ressources matérielles d'un ordinateur et fournit des services essentiels pour exécuter et coordonner les applications. Chaque fois que vous allumez

votre ordinateur, vous voyez un écran où vous pouvez effectuer différentes activités comme écrire, naviguer sur le NET ou regarder une vidéo ou afficher une image. Qu'est-ce qui fait que le matériel informatique fonctionne comme ça ? Comment le processeur de votre ordinateur sait-il que vous lui demandez d'exécuter telle ou telle tâche ? La réponse est évidemment est le système d'exploitation.

Un système d'exploitation agit comme intermédiaire entre l'utilisateur d'un ordinateur et le matériel informatique. Son but est de fournir un environnement dans lequel un utilisateur peut exécuter des programmes de manière pratique et efficace. Il a pour rôle de gérer des ressources matérielles et logicielles d'un ordinateur et permet aux autres programmes de s'exécuter. Il est chargé de s'assurer que le système fonctionne correctement et efficacement d'une manière facile à utiliser. Chaque système d'exploitation a ses propres caractéristiques, interfaces utilisateur et fonctionnalités spécifiques. Les plus courants incluent Windows, macOS, Linux, iOS et Android. Le code source du cœur d'un système d'exploitation comme Linux ou Windows est de l'ordre de cinq millions de lignes de code ou plus [6, 8, 1].

Ils fournissent des services essentiels tels que la gestion des fichiers, la gestion des processus, l'interfacage graphique permettant aux utilisateurs d'interagir avec l'ordinateur et ses ressources, la gestion de la mémoire, la gestion des utilisateurs et de la sécurité, etc. Le système d'exploitation agit comme un intermédiaire entre le matériel et les logiciels, fournissant un environnement dans lequel les programmes peuvent s'exécuter de manière ordonnée et efficace.

En bref :

1. Vous démarrez votre ordinateur.
2. Le Grub (GRand Unified Bootloader) utilisé par le chargeur d'amorçage GRUB s'affiche (1.1), il s'agit d'un programme de boot qui permet de choisir son système d'exploitation au démarrage de l'ordinateur. Il contient des paramètres qui influencent la manière dont GRUB affiche son menu, gère le temps d'attente, sélectionne les options par défaut.

   Avec Grub vous pouvez avoir sur le même ordinateur plusieurs OS, ainsi vous pouvez avoir un linux et un windows qui cohabitent sur la même machine. Voici ci dessus, l'écran de sélection des différents système (voir 1.1).

   Le temps par défaut est de 10 secondes. Pour le modifier, chercher la ligne dans votre

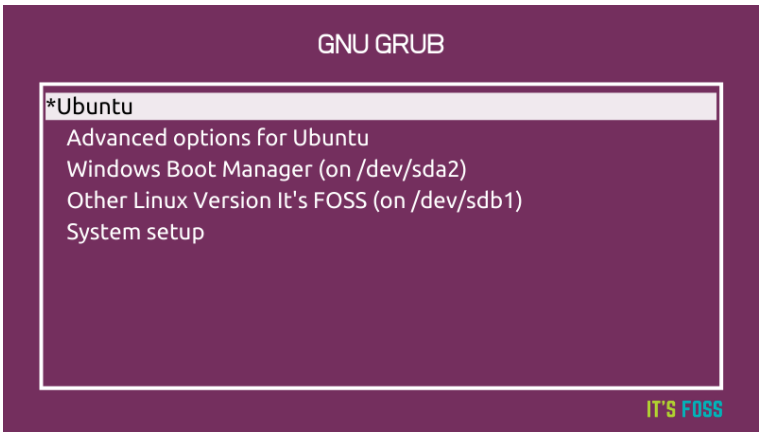


FIGURE 1.1 – GRUB menu

fichier de configuration ”timeout” dans le fichier /etc/default/grub.

Vous devez sélectionner sur la liste le système d'exploitation linux qui va se charger, puis vous arrivez sur votre écran de bureau.

3. Et vous pouvez utiliser vos programmes, jeux et applications en tout genre via votre souris, votre clavier, etc.

Pour mieux comprendre l'environnement dans lequel GRUB évolue, il convient de revenir sur la notion de logiciel libre.

1.1.2 Logiciel libre

Par définition, c'est une suite d'instructions informatiques formant un tout cohérent, mais qui a la particularité d'être libre, c'est-à-dire son utilisation, son étude, sa modification et sa distribution sont permises sans restriction. Ils sont généralement accompagnés de leur code source, ce qui permet aux utilisateurs de le modifier selon leurs besoins. Ils sont souvent développés et distribués dans le cadre d'une communauté collaborative.

La licence d'un logiciel libre prévoit le respect des quatre libertés suivantes :

1. La liberté de l'exécuter pour tous les usages sans aucune restriction, le code source est libre ;
2. La liberté d'en étudier le fonctionnement et de l'adapter aux besoins. Pour que ce principe soit satisfait, l'accès au code source (c'est-à-dire la "recette" du logiciel) est une condition requise ;
3. La liberté de redistribuer des copies et donc d'aider la communauté mondiale sans aucune distinction ;
4. La liberté de l'améliorer et de publier des améliorations sous réserve que ces modifications soient rendues accessibles à toute personne intéressée. Pour que ce principe soit satisfait, l'accès au code source (c'est-à-dire la " recette " du logiciel) est une condition requise.

Logiciel libre = Logiciel Open Source

Exemple : Ubuntu est une distribution et un système d'exploitation GNU/Linux gratuit et open source.

1.1.3 GPL (General Public License) / GNU

GNU et GPL sont deux concepts différents, bien qu'ils soient étroitement liés dans le domaine du logiciel libre. Elles ont été créée par un chercheur américain Richard Stallman, au milieu des années 80 pour formaliser juridiquement cette notion de liberté et de créer un système d'exploitation complet et entièrement libre. L'objectif initial du projet de développer un système compatible avec Unix mais totalement libre et ouvert. Le projet GNU comprend un ensemble d'outils, de bibliothèques et d'applications logicielles [7].

La licence nommée GPL est une licence de logiciel libre créée par la Free Software Foundation (FSF) pour promouvoir la liberté d'utilisation, d'étude, de modification et de distribution des logiciels. La GPL existe en différentes versions, telles que la GPL version 2 (GPLv2) et la GPL version 3 (GPLv3), chacune avec ses propres clauses et conditions spécifiques. Ces

licences ont été élaborées pour promouvoir la collaboration et le partage des connaissances dans le domaine du logiciel libre [5].

GPL n'est pas la seule utilisée pour les logiciels libres : Consultez le site (www.opensource.org) pour de plus amples informations sur les licences libres. Nous nous intéressons au système Linux, en particulier au système Ubuntu, car tout les exercices à réaliser dans les travaux pratiques se font sur un terminal à travers la pratique et le test de commandes Linux. Un bref historique est présente dans ce qui suit.

1.1.4 Historique

- Unix est né aux laboratoires Bell en 1969, Développé par Ken Thompson et Dennis Ritchie (le premier à avoir développé le langage C) ;
- En 1973, Unix a pu être réécrit, presque entièrement, en C (pour faciliter la Portabilité) ;
- Code source vendu à un prix bas aux sociétés ;
- Plusieurs sociétés (IBM , Sun ....) se sont intéressées au système et elles ont repris son développement pour avoir leur propre version (Solaris (Sun Microsystems) , AIX (commercialisé par IBM), HP-UX (développée par Hp), FreeBSD (BSD signifie "Berkeley Software Distribution").....) [4] [8].

1.1.5 Caractéristiques de linux

- Basé sur le principe tout est fichier ;
- Multi-utilisateurs et multi-tâches en temps partagé ;
- Utilisation du Shell comme interpréteur de commandes ;
- La configuration du système est stockée sous forme de texte ;
- Disponibilité sur un large gamme d'architecture matérielle (du PC jusqu'au Super calculateur massivement parallèle) ;
- C'est le système le plus utilisé (Dans les universités, les centres de recherches, les serveurs d'Internet, etc) ;
- Unix produit commercial (système payant).

1.1.6 Distributions de Linux

Une distribution Linux est un système d'exploitation qui est prêt à être installé, il est conçu à partir d'un noyau Linux qui prend en charge des référentiels, des bibliothèques et des programmes utilisateur. Chaque version d'un fournisseur ou d'une communauté est appelée *distribution*.

Étant donné que le système d'exploitation Linux est *Open Source* et *distribué* sous la licence publique générale *GNU*, n'importe qui peut exécuter, étudier, modifier et redistribuer le code source, ou même vendre des copies du code modifié.

C'est là toute la différence avec les systèmes d'exploitation traditionnels, tels que Microsoft Windows, Unix et macOS, qui sont propriétaires et bien plus difficiles à modifier.

Exemples de distributions

Il existe différentes distributions Linux (1.2). Certaines sont développées dans un cadre communautaire et d'autres dans un cadre commercial. Sachez que certaines sociétés développent une distribution communautaire sur laquelle elles fondent leurs distributions commerciales. Le distributions Linux les plus prisées sont les suivantes :

FIGURE 1.2 – Exemples des distributions linux

- Red Hat Linux USA www.redhat.com
- Mandrake Linux France www.linux-mandrake.com
- SuSE Linux (Allemagne www.suse.com)
- Debian (Internet www.debian.org)
- Fedora (www.fedora.org) [version libre de Red Hat]
- Ubuntu (www.ubuntu.com) [version de debian]et toutes ses versions (Gnome, Kubuntu, reposant sur KDE Plasma Desktop, Ubuntu MATE, Xubuntu, Lubuntu, etc.)
- KNOPPIX (https://www.knoppix.org)
- Android (https://developer.android.com)
- Kali Linux (https://www.kali.org/)
- LinuxCentOS (https://www.centos.org/)
- slackware (http://www.slackware.com)
- etc.

1.1.7 Qu'est-ce qu'Ubuntu ? - Présentation -

Le terme Ubuntu est issu de plusieurs langues sud-africaines, il fait référence à une idéologie qui se résume à l'expression : l'humanité envers les autres ; ou 'Je suis ce que je suis à cause de qui nous sommes tous '. Nous trouvons d'autres références qui présente Ubuntu comme 'la croyance en un lien universel de partage qui relie toute l'humanité'. Le célèbre défenseur sud-africain des droits de l'homme, l'archevêque *Desmond Tutu*, a expliqué Ubuntu de cette manière :

Citation de Desmond Tutu

Une personne avec Ubuntu est ouverte et disponible aux autres, affirmant les autres ne se sentent pas menacée par le fait que les autres sont capables et bons, car elle a une véritable assurance en soi qui vient du fait de savoir qu'elle appartient à un tout plus grand et est diminué lorsque les autres sont humiliés ou diminués, lorsque les autres sont torturés ou opprimés.

Ubuntu est une distribution GNU/Linux basée sur Debian et qui se destine à proposer un système convivial, ergonomique, libre et gratuit y compris pour les entreprises, une nouvelle version mise à jour est disponible tous les 6 mois. Conçu pour les ordinateurs de bureau (fixe et portable), aussi à des utilisations plus variées et spécifiques, avec :

- Ubuntu Netbook Edition, une version légère optimisée pour les netbook ;
- Ubuntu Server Edition, une version pour serveur.

Ubuntu se décline en de nombreuses variantes, dont les officielles sont :

- Edubuntu, variante spécialement conçue pour le monde de l'éducation ;
- Kubuntu qui est la variante utilisant le bureau KDE ;
- Xubuntu (prononcer " Zoubountou "), variante destinée à des ordinateurs de configuration modeste, utilisant le bureau Xfce.

La figure 1.3) représente le Logo d'Ubuntu 22.04 LTS, utilisé comme environnement de travail pour les travaux pratiques en systèmes d'exploitation.

FIGURE 1.3 – Logo de Ubuntu 22.04 LTS

Depuis sa création, l'identité visuelle de la distribution s'est basée sur un thème baptisé "Human" (humain), utilisant principalement des teintes brunes et orangées ; ainsi on retrouvait, à chaque sortie de version, un écran de démarrage, un fond d'écran, un écran de connexion, ainsi que le logo de la distribution basée sur ces caractéristiques.

Domaines d'utilisation de Linux ?

- Station de travail : Multimédia et bureautique (openoffice, koffice,...)
- Réseaux et Internet : serveur Web (Apache), messagerie (sendmail), Explorateur (FireFox de Mozila)
- Développement : C/C++, Delphi, Java, PHP,...
- SGBD (Oracle, Informix, MySQL, PostgreSQL....)
- Recherche scientifique

1.1.8 Installation de LINUX

Votre ordinateur doit être équipé d'un système d'exploitation linux, vous êtes libre d'installer chez vous une distribution linux (rpm ou debian ou autres) qui sont gratuites, par contre, au niveau des salles Tps la distribution installée est Ubuntu.

Téléchargeable gratuitement sur :

- http://univ-bejaia.dz/logitheque
- https://ubuntu.com/download/desktop

Il est possible d'exécuter Ubuntu directement à partir d'une clé USB ou d'un DVD, ceci représente un moyen plus rapide et facile de découvrir comment fonctionne le système. Cela ne modifie en rien la configuration de votre ordinateur, il suffit de redémarrer sans clé USB ou DVD restaurer votre machine à son état initial.

Liens intéressants

- https://doc.ubuntu-fr.org/initiation
- https://ubuntu.com/tutorials/command-line-for-beginners
- https://www.linux.org/
- https://www.redhat.com/en/topics/linux/what-is-linux
- https://www.makeuseof.com/tag/5-reasons-linux-can-help-become-better-student/

Après avoir installer Ubuntu sur votre ordinateur, c'est le temps de pratiquer

1.2 Premier pas vers linux

1.2.1 Ouverture de session

Le chargeur est le premier logiciel qui s'exécute quand un ordinateur démarre. C'est lui qui charge et transfère le contrôle au logiciel noyau d'un système d'exploitation. Le système linux étant un système multi-utilisateur, pour y a accéder on doit entrer un login et mot de passe utilisateur.

Se connecter est la première chose à faire. Lorsque vous démarrez l'ordinateur, choisissez Linux comme système d'exploitation. Le système démarre et affiche une fenêtre vous invitant à taper votre identifiant (login) et votre mot de passe (password), ce qui vous permet d'ouvrir une session.

NB : Retenez votre login et votre mot de passe dans un endroit sécurisé. Idéalement, vous devriez les retenir par cœur.

Il est nécessaire de savoir avant d'accéder à la machine que le système est caractérisé par ce qui suit :

1. Système Multi-Utilisateur
   - Utilisateur simple : possède des droits restreints et chaque utilisateur possède un répertoire de travail.
   - Super Utilisateur ou Administrateur (root) : possède tous les droits.
2. Organisation d'utilisateurs en groupes
   - chaque utilisateur doit appartenir à un groupe
3. Connexion au système
   - Mode texte ou console (exécution des commandes Shell)
   - Mode graphique (à l'aide du serveur de graphisme X-Window)

1.2.2 Localiser le noyau système

Le noyau (ou kernel) est le principal composant d'un système d'exploitation Linux et constitue l'interface entre le matériel d'un ordinateur et ses processus (voir 1.4). Il assure la communication entre les deux parties et gère les ressources aussi efficacement que possible.

Le noyau porte ce nom, car à l'instar d'une graine dans sa coque, il existe au sein du système d'exploitation et contrôle toutes les fonctions principales du matériel, qu'il s'agisse d'un téléphone, d'un ordinateur portable, d'un serveur ou de tout autre type d'ordinateur.

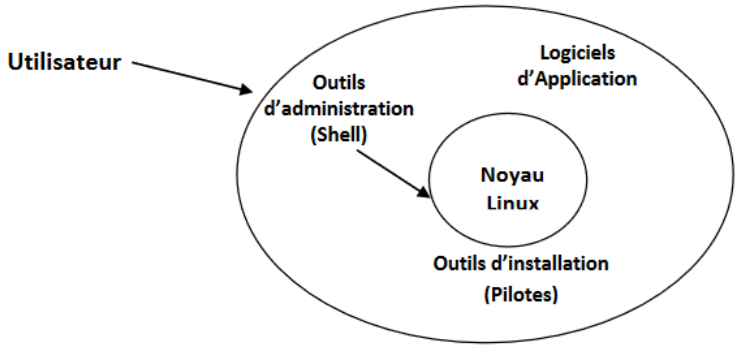


FIGURE 1.4 – Position du noyau

1. **Shell :** Interpréteur de commandes
2. **Pilotes :** Drivers des périphériques (disques durs, carte graphique,...)
3. **Logiciels d'applications :** différents logiciels

Une fois les fonctions de base assurées par le noyau Linux, la gestion de l'affichage graphique est prise en charge par des composants spécifiques tels que X Window, qui permet aux environnements de bureau comme KDE ou GNOME d'offrir une interface graphique complète à l'utilisateur :

- X-Window : gère l'affichage graphique de bas niveau ;
- KDE et Gnome : offrent une interface graphique utilisateur complète (barres de menus, fenêtres, tableaux de bord, applications système ; etc).

1.2.3 Découverte de l'interface graphique

Une fois votre nom d'utilisateur et mot de passe vérifiés, le gestionnaire de bureau, le programme qui gère l'affichage des menus et des fenêtres apparaît.
Environnement : Gnome, KDE, Xfce, Mate, etc Par défaut, l'environnement utilisé par défaut est Gnome.

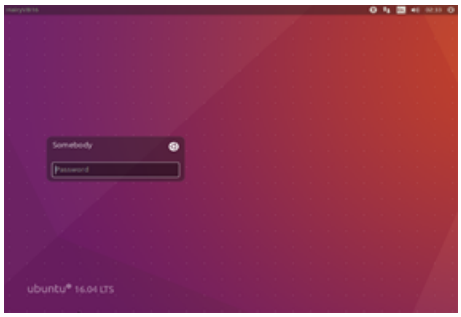


FIGURE 1.5 – Ouverture de session

1.2.4 Shell

Bien que les systèmes Linux disposent d'une interface utilisateur graphique, la plupart des programmeurs et des utilisateurs préfèrent toujours une interface de ligne de commande, appelée shell. C'est un interpréteur qui exécute les commandes une à une après traduction de l'instruction (écrite en langage évolué) en langage interne (écrite en langage machine). Shell veut dire Coquille, qui entoure le noyau [8]. L'utilisateur discute avec le Shell, qui discute avec le noyau, qui à son tour discute avec le matériel (voir la figure ci-dessous pour la position du Shell) :

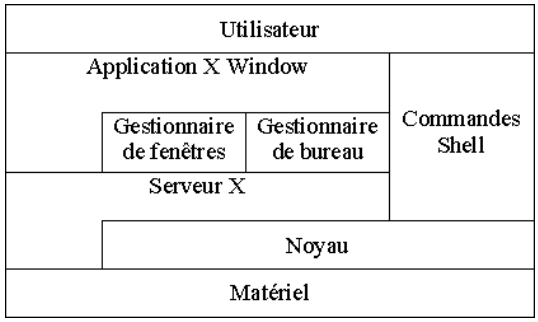


FIGURE 1.6 – Architecture des systèmes GNU/Linux

Le shell est utilisé sous Unix, il s'est répandu depuis avec différentes versions, la forme la plus simple est sh (L'ancêtre de tous les shells).

Voici d'autres formes :

- Bash (Bourne Again Shell) Une amélioration du Bourne Shell, disponible par défaut sous Linux et Mac OS X.
- Csh (C Shell) Un shell utilisant une syntaxe proche du langage C.
- Tcsh (Tenex C Shell) c'est le Csh amélioré
- Zsh : le petit dernier
- ksh : Un Shell puissant assez présent sur les Unix propriétaires, mais aussi disponible en version libre, compatible avec bash.

Il est possible d'avoir plusieurs Shells sur un même système, en se servant de la commande Chsh. *Exemple :* Taper la commande :

- Pour savoir quel shell est actif :

```
#echo $SHELL
```

- Puis ; taper la commande pour basculer vers le shell csh :

```
chsh csh
```

Commandes pour le shell

Lorsque le shell démarre, il s'initialise, un caractère d'invite s'affiche, souvent un pourcentage ou un signe dollar, sur l'écran et attend que l'utilisateur tape une ligne de commande.

— **Connexion en mode texte** : Si les login/password sont valides alors un message de la forme suivante sera affiché sur la ligne de commande : *[user@machine ~]$*

la description détaillée de cette ligne est la suivante :

- **user** le nom de l'utilisateur connecté
- **machine** le nom de la machine
- ~ caractère spécial désigne le répertoire de travail de l'utilisateur connecté
- **Le signe $** indique que vous êtes en un simple utilisateur
- **Le signe #** indique que vous êtes en mode super-user (le mode super utilisateur, appelé aussi root est un mode d'administration qui donne tous les droits sur le système).

NB : Le mode super utilisateur est puissant mais dangereux :

1. Une erreur peut casser le système (par exemple, supprimer un dossier système critique).
2. Il faut l'utiliser avec prudence et seulement quand nécessaire.

— **Les premières commandes**

Format d'une commande shell

Nous apprenons ici à manipuler les commandes en respectant leur format (voir 1.7).

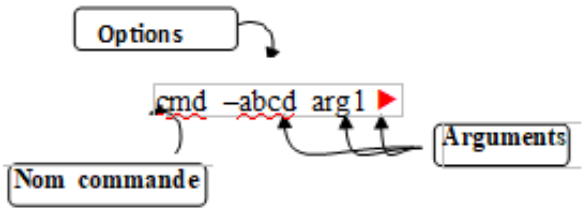


FIGURE 1.7 – Format d'une commande shell

[**Commande**][**option**][**paramètres**]

Chaque élément est décrit comme suit :

- **Option** non nécessaires pour exécuter la commande
- **Paramètres** arguments nécessaires pour exécuter la commande

  - Les trois champs sont séparés par des espaces

  -Plusieurs commandes sur la même ligne : séparer par ';'

Les commandes peuvent prendre des arguments qui sont transmis au programme appelé sous forme de chaînes de caractères.

Par exemple, la ligne de commande ci dessus invoque le programme cp avec deux arguments, src et dest. Cette commande interprète le premier comme étant le nom d'un fichier existant. Il fait une copie de ce fichier et appelle la destination de la copie.

cp src dest

— **Quelques commandes**

Voici un tableau présentant quelques commandes Linux couramment utilisées.

Commande	*Description*
ls	*Affiche l'ensemble des fichiers passés en argument puis la liste des fichiers contenus dans les répertoires passés en argument. Par défaut, elle affiche le répertoire courant '.'*
cd	*Elle permet de se déplacer dans l"arborescence du système de fichier.*
pwd	*Elle permet de connaître le répertoire courant*
date	*Afficher la date et l'heure*
cal	*afficher un calendrier*
cat	*Concaténer plusieurs fichiers sur une sortie standard*
chmod	*Changer le mode de protection des fichiers*
cp	*Copier un ou plusieurs fichiers*
grep	*Rechercher un fichier pour un certain modèle*
sort	*Trier un fichier de lignes par ordre alphabétique*
tail	*Extraire les dernières lignes d'un fichier*
head	*Extraire les premières lignes d'un fichier*
ps	*Liste des processus en cours d'exécution*
make	*Compiler des fichiers pour construire un binaire*
rmdir	*supprimer un répertoire*
uname	*afficher le nom et les caractéristiques du système.*
passwd	*modifier son mot de passe*
man	*manuel pour afficher une page d'aide (forme de commande, option,...)*

1.3 Série d'exercices avec corrigés

1.3.1 Exercice 01

Questions

Q a - Lancez quelques applications, et exercez-vous à agrandir, fermer et déplacer les fenêtres. Parcourez l'ensemble des menus disponibles et essayez de deviner à quoi sert chaque fonction. Avec le bouton droit de la souris, vous pouvez faire apparaître différentes commandes selon l'endroit cliqué (on parle de menus " contextuels ").

Q b - **Découverte du shell :** Comment ouvrir cette fenêtre du shell ou terminal ?

Corrigé- Exercice 01

Q a - L'exercice est à réaliser sur les postes de la salle de TPs :

- Prendre de temps pour se familiariser avec ce nouveau environnement Linux
- Faire toutes les manipulations demandées (pas de compte rendu à compléter)

Q b - Il est facile de trouver l'outil directement soit :

- En tapant "terminal" dans la barre de recherche des applications
- Par un raccourci clavier : appuyer simultanément sur les touches CTRL+ALT+T.
- Par le menu ou sur le tableau de bord :

  ∗**Localiser le terminal** trouver quelque part une icône qui ressemble à la figure 1.8

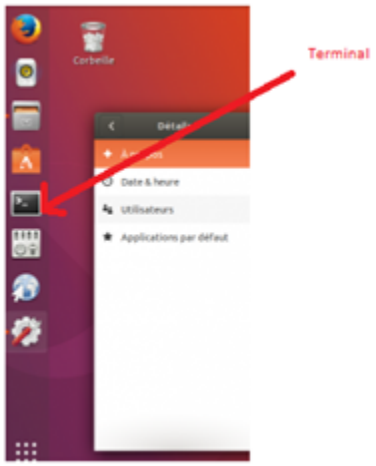


FIGURE 1.8 – Position du terminal

∗**La fenêtre terminal** Le shell (terminal ou console 1.9) vous indique qu'il est prêt

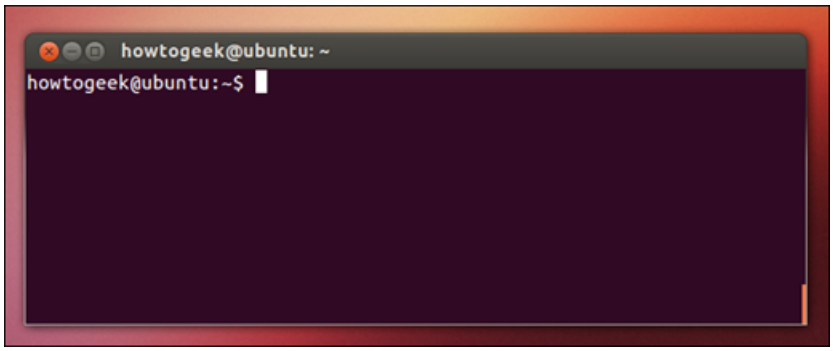


FIGURE 1.9 – Fenêtre shell

en affichant un invite (ou prompt), en général le caractère dollar ($) ou supérieur (>), en début de ligne.

NB : Ubuntu dispose d'une fenêtre *exécuter* pour lancer une commande d'une application installée. On peut alors utiliser celle-ci pour ouvrir le terminal :

- Appuyez sur Alt+F2 pour ouvrir la fenêtre.
- Puis saisissez sur le terminal la commande :
  - $ gnome-terminal pour la variante Ubuntu (Unity) voir d'autres commandes selon les variante de linux. Exemples :
  - $mate-terminal pour Ubuntu Mate ;
  - $xfce4-terminal pour Xubuntu ;
  - $lxterminal Lubuntu.
- Enfin appuyez sur Entrée sur le clavier pour ouvrir le terminal

Il est à noter que dans un terminal le raccourci *Ctrl+C* sert à interrompre l'exécution de la commande en cours. Cependant il est possible de modifier les raccourcis du Terminal afin de le personnaliser.

1.3.2 Exercice 02

Questions

Q a - **Commande ls :** Quelle est la commande qui permet de voir sur le terminal le manuel de la commande ls ? Citer quelques options de cette commande.

Q b - **commande cd :** commande cd sous linux appelée commande de changement (cd pour change directory) de répertoire. Il est utilisé pour changer le répertoire de travail actuel. Prenez des exemples de l'emploi de cette commande.

Q c - **Commande pwd :** Son nom signifie en anglais " print working directory". Tapez la commande pwd. Il vous indique le chemin allant de la racine du système de fichiers à ce répertoire.

- Comment pouvez vous revenir dans ce répertoire à partir de n'importe quel endroit dans le système de fichiers ?

Corrigé- Exercice 02

Q a - Taper sur le terminal pour voir le manuel de la commande *ls* : *man* ls

Quelques options

ls → afficher le contenu d'un repertoire (ou dossier)

ls -l → l'option l affiche la liste en détails

ls -a → l'option a affiche en plus les fichiers cachés (ceux dont le nom commence par un "." (point), comme ".bashrc

ls -S → afficher par taille

Exemple

ls -l / permet d'afficher en détails le contenu du répertoire " / "

$ *ls* -laS cumuler les options et obtenir un listing ordonné (-l) incluant les fichiers cachés (-a) et classé par taille (-S).

Q b - **Commande *cd***

Exemple

~$ *cd* Documents

Documents$

pour sortir de ce répertoire, tester les commandes suivantes :

Documents$ *cd* ..

~$

Autres fonctionnalités

$ *cd* revenir au répertoire ~

$ *cd* aller au repertoire racine "/"

Q c - **Commande pwd**

- Pour revenir dans ce répertoire à partir de n'importe quel endroit dans le système de fichiers il suffit de taper tout simplement : la commande "*cd*".

NB : Le ~ utilisé en premier nom de répertoire remplace le chemin absolu vers son répertoire personnel soit /home/utilisateur, avec utilisateur comme le nom de la session en cours, c'est là que vous trouverez vos fichiers personnels, vos configurations et vos données.

1.3.3 Exercice 03

Questions

Q a - A quoi sert les commandes : *whoami*, *cat*, *less*, *gedit* ?

Indication : vous allez utiliser la commande man pour comprendre le rôle de chaque commande,

Q b - Comment exécuter un programme écrit en langage C

Q c - Écrire un programme C qui affiche un message de " bienvenu "

Q d - comment arrêter le système avec la commande shutdown

Corrigé- Exercice 03

Q a - Utiliser la commande man pour aller sur le manuel des commandes :

whoami Afficher l'identifiant de d'utilisateur associé à l'utilisateur courant.

cat afficher le contenu d'un fichier sur une sortie standard

less permettant d'afficher un fichier texte page par page

gedit éditeur graphique de texte libre (sous licence GPL), il permet l'édition et l'enregistrement de fichiers textes

Q b - **Voici les étapes à suivre dans le terminal (ou shell)**

- Editer le fichier

  gedit exercice2.c

- Compiler le programme : Le *gcc* est un compilateur utilisé principalement pour compiler des programmes en langage C (mais aussi en C++, Objective-C, etc.). Voici le format de la commande utilisée pour compliler le fichier exercice2.c :

  gcc exercice2.c -o exercice2

  Détail :

  (a) exercice2.c : le fichier source contenant le programme en C ;

  (b) -o exercice2 : le nom de l'exécutable généré (sinon il s'appelle par défaut a.out).

- Exécuter le programme : ./ indique qu'on veut exécuter comme programme qui se trouve dans le répertoire courant (le répertoire où on est actuellement dans le terminal).

  ./exercice2

Q c - **Arrêt du système avec la commande shutdown**

- Arrêter immédiat. root : # *shutdown* -h now (h pour halt)
- Redémarrer le système. root : # *shutdown* -r now (r pour reboot)
- Planifier l'arrêt du système $\#Shutdown\ -h + 30$: maintenance du système dans 30 min
- Pour arrêter le système avec la commande *shutdown*, on peut utiliser cette commande dans un terminal :

  sudo shutdown -h now

  Cela initiera un arrêt immédiat du système.

2 TP. Manipulation de dossiers et de fichiers

Sommaire

2.1 Arborescence du système de fichiers

Les fichiers sont regroupés dans des répertoires et sous-répertoires (également appelés dossiers). Cette organisation des fichiers forment une arborescence. Sous linux tout est fichier y compris les périphériques sont représentés par un fichier spécial dans le répertoire /dev, exemples :disques, souris, carte son, ports d'E/S,etc. On distingue plusieurs types de fichiers :

— Les fichiers ordinaires, qui contiennent des données ;

— Les répertoires, qui contiennent une liste de références à d'autres fichiers UNIX ;

— Les fichiers spéciaux, associés par exemple à des pilotes de périphériques ;

— Les liens symboliques (fichiers ”pointant” sur un autre fichier) ;

— Les tubes utilisés pour la communication entre processus ;

— Les sockets.

On notera ici qu'un répertoire n'est donc qu'un type de fichiers particulier. La structure hiérarchique des fichiers et répertoires est la suivante :

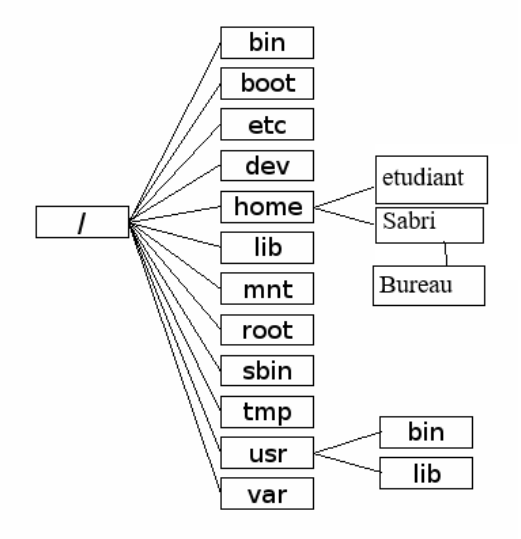


FIGURE 2.1 – Arborescence du système linux

— La racine est dénoté par ‘/‘ ; c'est le ”*gros dossier de base qui contient tous les autres dossiers et fichiers*”

— Les chemins sont séparés par ‘/‘

— Les noms des objets sont des séquences de maximum 255 caractères sensibles à la casse.

— Il est préférable de ne pas utiliser les caractères : ?, *, &. ’, ’,”,<,>

— Il convient de se limiter à : A à Z , a à z, 0 à 9, le caractère de soulignement ’ _’, le tiret ’-’ et le point ’.’

— Le point (.) comme premier caractère d'un nom signifie un fichier caché.

— Éviter des noms contenant des caractères accentués ou des blancs.

— Le point (.) ne sépare pas forcement le nom de son extension (Exemples : rapport.ps.gz, postgresql-2.3.6-src.tar.Z, README)

2.1.1 Fichiers cachés

Sous Linux les fichiers sont cachés si leur nom commence par un point. Dans l'explorateur de fichier nous pouvons les voir en activant l'affichage des fichiers cachés (Ctrl + H)ou bien sur le terminal, en tapant la commande : **ls -a**

2.1.2 Principaux répertoires

Les répertoires principaux dans un système Linux font partie de la hiérarchie du système de fichiers (Filesystem Hierarchy Standard, FHS). Cette hiérarchie organise les fichiers et répertoires de manière standardisée pour faciliter la gestion et la maintenance du système [4].

Voici quelques répertoires :

- **/boot** contient les fichiers nécessaires au démarrage du système.
- **/bin** contient les commandes de base (binaire)
- **/sbin :** contient les commandes du super utilisateur (administrateur)
- **/usr/bin, /usr/sbin,... :** contient des fichiers binaires
- **/etc :** contient les fichiers de configuration du système et des services
- **/home :** contient les répertoires personnels des utilisateurs simples
- **/root :** contient le répertoire personnel de l'administrateur.
- **/usr :** contient le reste des programmes du système et les logiciels.
- **/dev :** contient les noms périphériques.
- **/mnt/ :** souvent vide. Il est prévu pour être utilisé comme un emplacement temporaire pour monter des systèmes de fichiers.
- **/media/ :** monter automatiquement les périphériques amovibles (Clés USB, CD/DVD, etc.).
- **/proc** est un répertoire virtuel, n'a pas d'existence sur disque, contient les informations sur les processus. Taper *cat /proc/meminfo*, pour des informations sur la mémoire.

2.2 Chemin relatif et chemin absolu

Un fichier est localisable sans ambiguité par son chemin. Dans linux, un répertoire de haut niveau est appelé "root ", qui est identifié comme un simple slash ("/"), on peut y accéder par la commande " cd / " . Le répertoire / dispose de plusieurs sous-répertoires comme " /home ", " /bin " et " /usr ".

2.2.1 Chemin absolu

Le chemin absolu est le chemin d'accès complet, depuis le répertoire racine.

Exemple : Accéder à un fichier dans le répertoire personnel d'un utilisateur.
Chemin absolu : /home/utilisateur/Documents/rapport.txt
Explication : Cela signifie que le fichier rapport.txt se trouve dans le répertoire Documents, lui-même situé dans le répertoire personnel de l'utilisateur (/home/utilisateur).

2.2.2 Chemin relatif

Le chemin relatif est une partie du nom de chemin, "relative" au sous - répertoire dans lequel vous travaillez actuellement. Il désigne la succession des répertoires à parcourir depuis le répertoire courant pour accéder au fichier spécifié.

Exemple Vous êtes dans /home/utilisateur
Chemin relatif : Documents/rapport.txt
Explication : Le fichier rapport.txt se trouve dans le sous-répertoire Documents du répertoire courant.

2.3 Manipulation des fichiers et des répertoires

A l'aide de quelques commandes de base, vous serez en mesure d'effectuer toutes les opérations utiles sur le système de fichiers (parcours, copie, déplacement, etc.) par l'intermédiaire du shell. Et vous verrez que cela s'avère souvent bien plus rapide que d'utiliser l'interface

graphique. Nous allons voir les commande suivantes :

pwd indique dans quel répertoire courant

cd se déplacer dans l'arborescence des répertoires

ls Affiche le contenu du répertoire courant

mkdir créer un nouveau répertoire

mv renommer ou déplacer un fichier

rm supprimer un fichier

rmdir supprimer un répertoire non-vide

***rm* -r** supprimer un répertoire non-vide

***rm* -i** supprimer avec confirmation

cp copier un fichier

***cp* -r** copier des répertoires de manière récursive

tar Commande servant à archiver plusieurs fichiers en un seul.

2.4 Les jokers * et ?

Pour faciliter la spécification de plusieurs noms de fichiers à la fois, le shell accepte les caractères magiques, parfois appelés caractères génériques ou jokers. Il en existe plusieurs, mais les principaux sont :

*	Dans un nom de fichier, représente n'importe quelle suite de zéro, un ou plusieurs symboles
?	Il représente exactement un symbole quelconque.

Exemple

— Pour lister tout les fichiers programme en langage C, un astérisque, par exemple, correspond à toutes les chaînes possibles, donc la commande correspondante est :

ls * .c

— Une liste de caractères entre crochets sélectionne n'importe lequel d'entre eux, donc :

ls [abc]*

Elle répertorie tous les fichiers commençant par a , b ou c .

2.5 Historique des commandes

La commande *history* sert à faire une recherche des commandes déjà exécutées. Elle affiche une liste des commandes récentes, chacune précédée d'un numéro. En tapant **!***n* où **n** est un numéro dans l'historique, on rappelle la commande correspondante.

2.6 Commande alias

Certaines commandes sont plutôt longues à taper, notamment lorsqu'il y a des options. Une commande permet de pallier à cet inconvénient en créant ses propres commandes : alias. Le format de cette commande est le suivant :

alias ma_commande = "commande_complète"

Exemple

Pour faire un alias cls de la commande 'clear', comme suit :

$ *alias* cls='clear'

Pour supprimer cet alias nommé cls, tapez la commande :

$ *unalias* cls

2.7 Série d'exercices avec corrigés

2.7.1 Exercice 01

Questions

Q a - Quand devrait-on utiliser le répertoire personnel ?

Q b - Est-il possible de partager des fichiers ou des répertoires personnels avec d'autres utilisateurs ?

Q c - Comment localiser le dossier personnel à partir du terminal ?

Q d - Quel est le contenu du répertoire personnel ?

Corrigé- Exercice 01

Q a - Il est essentiel d'utiliser son répertoire personnel pour sauvegarder tous les fichiers, configurations et données liés à son compte utilisateur. Cet espace, propre à chaque utilisateur dans le système, permet de conserver ses fichiers de manière organisée, contribuant ainsi à maintenir l'ordre et la sécurité.

Q b - Oui, il est possible, mais il est obligatoire de définir les autorisations appropriées pour permettre aux autres d'y accéder. Ces autorisations peuvent être ajustées en fonction du niveau d'accès à accorder, selon les besoins et la politique de sécurité.

Q c - En tapant la commande *echo* $HOME ou en référant par le symbole tilde ~ qui guide directement vers votre espace personnel au sein du système.

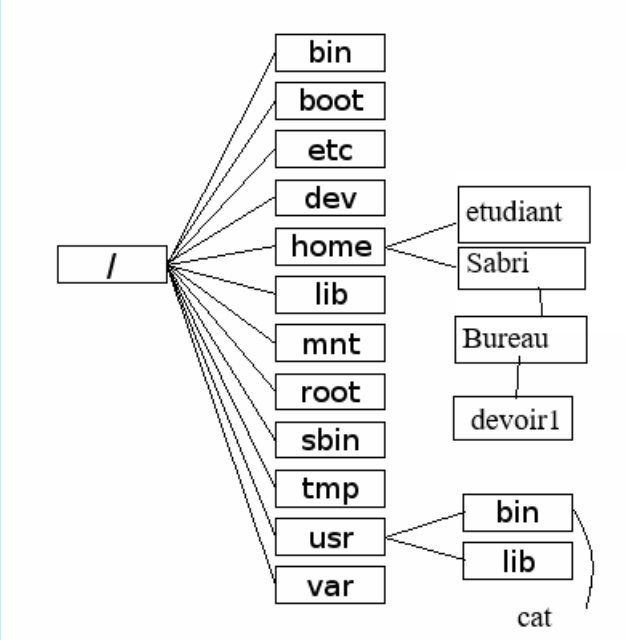
Q d - Via la commande '*ls* -al' un listing détaillé sera renvoyé.

Dans le répertoire personnel, on trouve généralement des documents, des photos, des videos, des configurations d'applications personnelles. En plus, des fichiers et répertoires cachés(commençant par un point).

2.7.2 Exercice 02

Questions

Q a - **Chemin relatif-chemin absolu :** Soit l'arborescence suivante des fichiers :



Déterminez les chemins absolus permettant d'accéder :

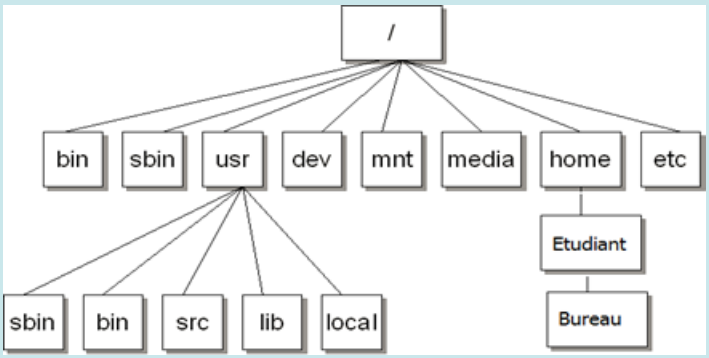
— A la commande *cat*

— Au fichier devoir1

Q b - **Exemple** : *../Bureau/monprog.c* .

Supposant qu'on est positionné au niveau du répertoire personnel.

Parcourir cette arborescence avec la commande **cd** en appliquant les notions des chemins relatifs et absolus.



Corrigé- Exercice 02

Q a - En utilisant les chemins absolus comme suit :

- /usr/bin/cat
- /home/Sabri/Bureau/devoir1

Q b - L'étudiant est appelé à utiliser les différentes manipulation sur des fichiers ou des répertoires en utilisant les chemins absolus et relatifs selon les exemples étudiés.

2.7.3 Exercice 03

Questions

Q a - Le symbole &

- Créer un fichier texte en tapant dans le terminal la ligne de commande : *gedit* fichier.txt
- Une fois l'éditeur ouvert, revenez sur le terminal et tapez une commande (par exemple whoami). Que se passe-t-il ?
- Toujours sur le terminal, gardez la touche Ctrl enfoncée puis pressez et relâchez la touche z. Que se passe-t-il ?
- Répétez les étapes 1 et 2, en ajoutant cette fois le caractère & en fin de commande (soit par exemple gedit fichier.txt &).Voyez-vous une différence ?

Corrigé- Exercice 03

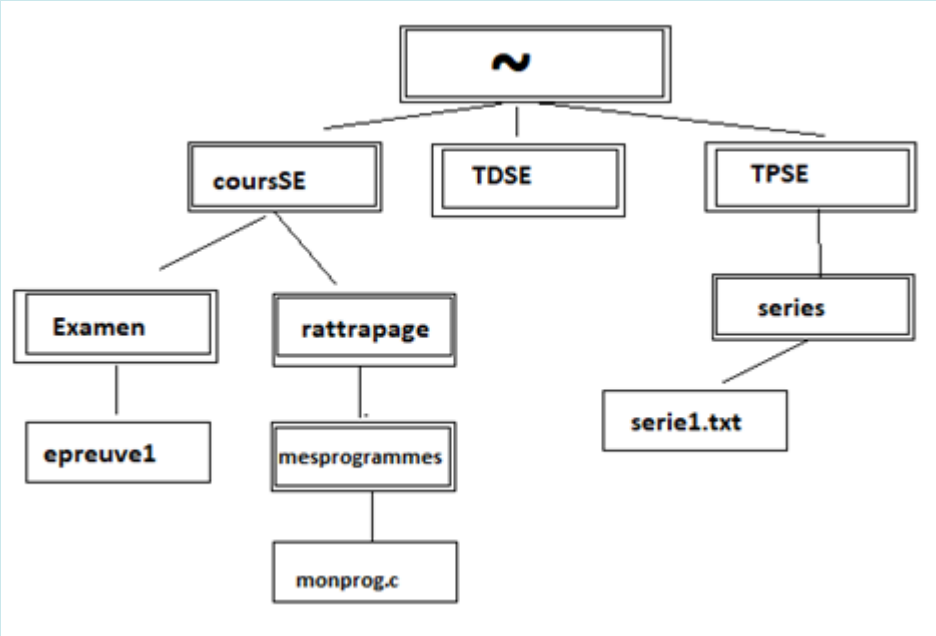
Q a - Le symbole &

- *gedit* fichier.txt (crée dans le repertoire personnel ou dans un sous répertoire)
- En tapant whoami aucun résultat n'est affiché du fait que la commande n'est pas exécutée
- L'application en cours est annulée (gedit)
- En ajoutant cette fois le caractère & en fin de commande (soit par exemple gedit fichier.txt &). L'utilisateur a la main sur le terminal, ainsi d'autres commandes pourront être exécutées.

2.7.4 Exercice 04

Questions

Q a - **Créer l'arborescence suivante :**



- déplacer le dossier " mesprogrammes " vers le dossier " series "
- renommer le dossier " Examen " par " ExamensSE "
- copier le fichiers " epreuve1 " vers le dossier " TDSE "
- archiver les trois dossiers principaux avec la commande tar

Q b - **les jokers**

- Affichez la liste de tous les fichiers dans le répertoire /usr/bin dont le nom commence par k et contient exactement 6 caractères.
- Affichez la liste de tous les fichiers dont l'extension est so dans le répertoire /usr/lib.
- Copier les fichiers C se trouvant dans votre bureau vers votre clé usb

Q c - Créer une nouvelle commande. Testez-la. Puis ouvrez un nouveau terminal et testez-la dans celui-ci. Que constatez-vous ?

Corrigé- Exercice 04

Q a -
- déplacer le dossier " mesprogrammes " vers le dossier " series "

  $ *mv* ./coursSE/rattrapage/mesprogrammes ./TPSE/series
- renommer le dossier " Examen " par " ExamensSE "

  $ *mv* ./coursSE/Examen ./coursSE/ExamenSE
- copier le fichiers " epreuve1 " vers le dossier " TDSE "

  $ *cp* ./coursSE/ExamenSE/epreuve1 ./TDSE
- archiver les trois dossiers principaux avec la commande tar

  tar -cvf coursSE/ TDSE/ TPSE/ archive.tar

Q b -
- *ls* k?????
- /usr/lib$ ls *.so
- *cp* -r ./Bureau/*.c /media/clé

Q c - Dans un nouveau terminal la nouvelle commande n'est pas reconnue.

Le problème avec la commande alias est qu'elle n'agit que dans le terminal dans lequel on est en train d'écrire : dès qu'on se déconnecte, tout est à refaire. Pour stocker ce genre de paramètres une bonne fois pour toutes, on utilise des fichiers de configurations liés au shell. Pour le shell bash, ces fichiers sont .bash_profile et .bashrc. Ce sont des fichiers texte contenant des commandes qui sont lues au moment de l'ouverture d'un terminal. Tout changement dans ces fichiers ne sera donc pris en compte que si on relance le shell.

Soit une nouvelle commande " **list** " qui liste le contenu d'un répertoire en détails définie ainsi :

alias list= 'ls -l '

On va éditer le fichier *.bashrc* contenu dans votre répertoire personnel avec la commande :

$*gedit* .bashrc

On ajoute dans ce fichier la commande suivante : *alias* list= 'ls -l ' , puis, sauvegarder les changements.

3 TP. Utilisateurs, groupes et permissions

Sommaire

3.1 Utilisateurs

Un utilisateur est toute entité (personne physique ou programme particulier) devant interagir avec un système UNIX, il est authentifiée sur cet ordinateur par un utilisateur ou "user". Ceci permet d'identifier un acteur sur un système UNIX. Un utilisateur est reconnu par un nom unique et un numéro unique.

Sur tout système UNIX, il y a un **super-utilisateur**, généralement appelé **root**, qui a tous les pouvoirs sur le système. Il peut accéder librement à toutes les ressources de l'ordinateur, y compris à la place d'un autre utilisateur, c'est-à-dire sous son identité. En général, seul l'administrateur système possède le mot de passe root. L'utilisateur root porte le numéro 0.

3.1.1 Programme sudo

sudo (ou bien substitute user do, en anglais) : "exécuter en se substituant à l'utilisateur" est une commande qui permet à l'administrateur système d'accorder à certains utilisateurs (ou groupes d'utilisateurs) la possibilité de lancer une commande en tant qu'administrateur, ou comme autre utilisateur, tout en conservant une trace des commandes saisies et des arguments.

Voici deux exemples de fichiers accessibles uniquement par l'utilisateur *root*.

3.1.2 Le fichier /etc/passwd

On peut créer un utilisateur de plusieurs manières mais la finalité est toujours la même : pour chaque utilisateur, une entrée doit être créée dans le fichier /etc/passwd sous ce format :

```
account :passwd :UID :GID :GECOS :directory :shell
```

Il est possible d'ajouter un utilisateur directement en insérant la ligne suivante dans le fichier /etc/passwd.

```
echo "user1 :x :2000 :2000 :user1 :/home/user1 :/bin/bash" >> /etc/passwd
```

3.1.3 Le fichier /etc/shadow

Ce fichier a un rôle important dans la configuration des utilisateurs Linux et l'administration système. Il appartient à l'utilisateur *root* et au groupe *shadow* et avec des permissions définies à 640.

Les mots de passes sont enregistrés dans le fichier */etc/shadow* avec les paramètres suivants :

1. Mot de passe chiffré ;
2. Date du dernier changement de mot de passe ; la date à laquelle le mot de passe a été modifié ;
3. Age minimum du mot de passe : Le nombre de jours qui doivent s'écouler avant que le mot de passe de l'utilisateur puisse être modifié ;
4. Age maximum du mot de passe : Le nombre de jours après le mot de passe de l'utilisateur doit être changé ;
5. Période d'avertissement d'expiration du mot de passe : Le nombre de jours avant l'expiration du mot de passe pendant lesquels l'utilisateur est averti que le mot de passe doit être changé ;
6. Période d'inactivité du mot de passe :Le nombre de jours après l'expiration du mot de passe utilisateur avant que le compte utilisateur ne soit désactivé ;
7. Date de fin de validité du compte : la date à laquelle le compte a été désactivé ;
8. Et un champ réservé.

3.1.4 Commandes de manipulation des utilisateurs

Les commandes les plus courantes sont :

— **Supprimer un utilisateur** via la commande userdel. **Exemple :**

```
userdel user1
```

Cette commande va supprimer l'utilisateur user1.

— **Ajoute l'utilisateur** via la commande adduser. ***Exemple :***

```
adduser user1
```

Cette commande va créer un nouveau compte utilisateur nommé *user1*.

— **Attribuer un mot de passe** via la commande passwd. ***Exemple :***

```
passwd user1
```

Cette commande va donner ou changer le mot de passe de *user1*.

— **Changer d'utilisateur** via la commande su . ***Exemple :***

```
su user1
```

Cette commande vous permet de basculer vers l'utilisateur *user1*.

— **Aller au super-utilisateur** via la commande suivante :

```
su
```

On bascule vers le super-utilisateur (administrateur).

3.2 Groupes

Les groupes servent à rassembler des utilisateurs afin de leur attribuer des droits communs. Le groupe principal est le groupe initial de l'utilisateur. Ce dernier peut appartenir à des groupes secondaires.

3.2.1 Fichier /etc/gshadow

Le fichier */etc/gshadow* renferme des données confidentielles concernant les groupes, ce qui justifie sa restriction d'accès pour préserver la sécurité des mots de passe.

Chaque ligne de ce fichier contient les champs : nom du groupe, mot de passe chiffré, administrateurs et membres, séparés par des deux-points.

3.2.2 Le fichier /etc/group

Le fichier */etc/group* contient des nformations sur les groupes, il comporte 4 champs séparés par " :" :

1. Nom du groupe ;
2. Mot de passe du groupe (ou x si le fichier gshadow existe) ;
3. Le GID (Group IDentifier) est un identifiant unique attribué à chaque groupe ;
4. Liste des membres séparés par une virgule.

3.2.3 Commandes de manipulation des groupes

— **Identifiants** pour connaître les noms d'utilisateur et de groupe ainsi que les identifiants numériques (UID ou ID de groupe) de l'utilisateur actuel ou de tout autre utilisateur du serveur.

Exemple : La commande suivante imprime uniquement l'identifiant de groupe effectif :

```
id -g
```

— **Appartenance à un groupe** via la commande : `groups`

— **Ajouter un groupe** via la commande : `addgroupe`

Exemple : ajouter le groupe gpr1 :

```
addgroupe grp1
```

— **Ajouter un utilisateur à un groupe** via la commande : `adduser`

Exemple : ajouter l'utilisateur user1 au groupe grp1 :

```
adduser user1 grp1
```

— **Créer un groupe** via la commande : groupadd

Exemple :crée un groupe grp2 via la commande :

```
groupadd grp2
```

— **Changer le groupe à un utilisateur** via la commande : chgrp

Exemple :le groupe de user est maintenant gpr2 après l'application de cette commande :

```
chgrp grp2 user
```

— **Supprimer un groupe** via la commande : groupdel

Exemple :effacer le groupe gpr2 avec l'application de cette commande :

```
groupdel grp2
```

— **Ajouter un utilisateur à un groupe** via la commande : adduser

Exemple :

```
adduser -ingroup grp1 user1
```

Avec *user1* qui fait dorénavant du groupe *gpr1* (consulter les fichiers : /etc/passwd, etc/group, /etc/shadow).

3.3 Droits d'accès - les permissions-

La grande différence entre Linux et Windows se situe au niveau des droits d'accès à un fichier : Chaque fichier Linux est caractérisé par un droit d'accès en lecture, en modification et en exécution pour le propriétaire du fichier, pour le groupe dont fait partie l'utilisateur et pour toutes les autres personnes.

Ci-dessous, tout ce qu'il faut retenir, avant d'appliquer la commande qui permet d'effectuer les changements sur les droits d'accès.

Les utilisateurs
1. **u** (user, utilisateur) représente la catégorie ”propriétaire”
2. **g** (group, groupe) représente la catégorie ”groupe propriétaire”
3. **o** (others, autres) représente la catégorie ”reste du monde”
La modification que l'on veut faire
1. + : ajouter
2. **-** : supprimer
3. = : affectation
Le droit que l'on veut modifier
1. **r** read lecture (pour un répertoire c'est le droit d'exécuter la commande ls)
2. **w** : write écriture (pour un répertoire c'est le droit d'exécuter des commandes comme : mkdir, rmdir, mv, etc)
3. **x** :execute (pour un répertoire x est le droit d'exécuter la commande cd)

Utilisez ls -l depuis votre répertoire personnel (˜) pour repérer les symboles décrivant les droits d'accès : lecture, écriture ou exécution.

On obtient quelque chose de ce genre :

-rw-r–rw-	4	Sabri	Sabri	4	5	Juil	2023	devoir1.c
drw-r–r–	1	Sabri	Sabri	6	10	Juil	2023	Doss1
-rwxr-xr-x	1	Sabri	Sabri	7	10	Juil	2023	test.txt

Le premier caractère est un **tiret s'il s'agit d'un fichier** ou un **d s'il s'agit d'un répertoire**.

Le champ rw- r– rw- qui suit indique les permissions d'accès, il s'agit de 3 séries d'autorisation en lecture (r), en écriture (w) et en exécution (x).

Chacune de ces séries concernent le propriétaire, le groupe, et tout le monde. Il se décompose en trois parties, prenons, la série du fichier *devoir1.c* par exemple :

1. **rw-** donne une permission en lecture, écriture au propriétaire
2. **r–** donne une permission uniquement en lecture au groupe dont fait partie l'utilisateur
3. **rw-** donne une permission en lecture, écriture à tout le monde

3.3.1 Commande chmod

chmod permet de modifier les droits ou bien le mode (autorisations) et permissions.

NB : N'oubliez pas que seul le propriétaire du fichier ou le super-utilisateur peut modifier le mode d'un fichier ou d'un répertoire [5].

chmod prend en charge deux façons distinctes de spécifier les changements de mode pour le réglage des droits : la notation symbolique et la notation octale ou bien numérique où l'utilisateur est soit spécifié (en cas d'omission, la valeur par défaut est a pour all) [5].

3.3.2 Notation symbolique

En mode symbolique, la commande est de la forme "chmod who opcode permission" , ou who représenté par des lettres qui sont attribuées aux différentes catégories d'utilisateurs et aux différents droits d'accès en combinant ces lettres, il est ainsi facile de préciser quels droits doivent être ajoutés ou supprimés (avec opcode) via la commande *chmod* qui permet d'accorder ou de retirer au(x) fichier(s) (ou répertoire(s))[5].

La figure 3.1 résume les lettres de balisage pour les utilisateurs et les droits dans le cadre de la notation symbolique que nous avons utilisé dans nos séances des travaux pratiques :

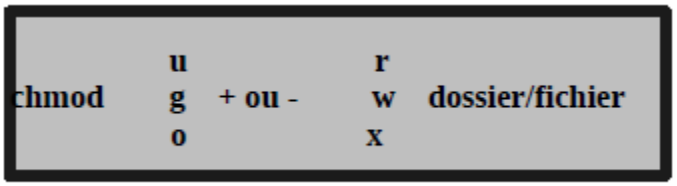


FIGURE 3.1 – Droit d'accès

Exemple :

— Ajouter le droit en exécution pour le propriétaire sur un fichier file, la commande à appliquer est :

```
$chmod u+x file
```

— Soit un fichier " exemple.txt " doit être modifié de manière à ce que non seulement le propriétaire (user) mais aussi tous les autres utilisateurs (group, others) aient la permission de lire et d'écrire, la commande chmod ressemblerait à ceci :

```
$ chmod ugo+rw exemple.txt
```

3.3.3 Notation octale

Il s'agit d'une séquence de trois chiffres, chaque chiffre représentant une catégorie d'utilisateur. La notation octale est basée sur la séquence standard :

Octal	Binaire	mode de fichier
0	000	---
1	001	--x
2	010	-w-
3	011	-wx
4	100	r--
5	101	r-x
6	110	rw-
7	111	rwx

FIGURE 3.2 – Notation octale

Exemple :

— Pour l'exemple précèdent, avant d'appliquer la commande chmod, il est nécessaire de voir les droits actuel sur le fichier, soit par exemple (rwx — —), le fichier " exemple.txt " doit être modifié alors par la commande chmod comme suit :

```
$ chmod 766 exemple.txt
```

— Dans l'exemple suivant, on assigne les droits en lecture/ecriture/execution par le propriétaire, ecriture/execution par groupe et execution uniquement par les autres sur le fichier "file" :

```
$ chmod 751 file
```

En notation symbolique, cela correspond à la commande :

```
$ chmod u=rwx,g=rx,o=x file
```

— Soit pour affecter uniquement le droit en lecture sur le fichier "file" pour tous :

```
$ chmod 444 file
```

En notation symbolique, cela correspond à la commande :

```
$ chmod =r file
```

Exemple :

Consultez le manuel de la commande (man chmod). Voici quelques options de cette commande :

— **L'option -R, –recursive**

Traverse subdirectories recursively, applying changes.

Cela signifie, la modification touchera en plus, tout les sous-répertoires de manière récursive. Soit la commande suivante :

```
chmod -R a+rx mon_dossier
```

Cette commande donne à tous les utilisateurs les droits en lecture et en exécution à tout ce que contient le dossier mon_dossier.

— **-c, –changes**

Print information about files that are changed.

Cela signifie, qu'on peut afficher uniquement les informations sur les fichiers modifiés. Soit la commande :

```
$ chmod -c 644 /home/sabri/test.txt
```

Cette commande modifie les droits en rw-r–r– pour le fichier test.txt et l'afficher.

3.4 Commande umask

Pour définir des droits d'accès par défaut pour l'ensemble des fichiers et des répertoires on utilise la commande *umask*. Il est possible d'accorder certains droits aux membres de groupe ou aux autres par mesure de sécurité, la commande *umask* est utilisée dans les fichiers .cshrc et .profile de façon à protéger chaque nouveau fichier ou répertoire à créer.

La commande suivante permet d'afficher le masque courant [5], en utilisant une notation symbolique :

```
$ umask -S
```

La méthode de calcul des permissions pour la commande *umask* est différente de celle de la commande *chmod*.

Tout d'abord, si la valeur définie pour *umask* est 000, alors, tous les fichiers que vous créez ont les droits d'accès (en lecture, écriture, mais non en exécution) auront les droits suivants :

```
rw-rw-rw- (mode 666)
```

Ainsi, tous les répertoires créés ont les droits d'accès (en lecture, écriture et exécution) suivants :

```
rwxrwxrwx (mode 777)
```

Pour déterminer la valeur à utiliser pour la commande *umask*, vous devez soustraire la valeur des droits d'accès souhaités (au moyen de la valeur que vous définiriez pour la commande *chmod*) des droits d'accès par défaut en cours affectés aux fichiers. Le résultat de l'opération représente la valeur à utiliser pour la commande *umask*.

Exemple :

— Pour définir le droit par défaut des fichiers à 644 (rw- r– r–), on doit soustraire 644 de 666. Le résultat est 022 représente la valeur numérique à utiliser pour la commande umask, comme indiqué ci-dessous :

```
umask 022
```

— La commande suivante :

```
umask 002
```

Elle va produire la configuration du mode suivante : **rw- rw- r–**.

3.5 Série d'exercices avec corrigés

3.5.1 Exercice 01

Questions

Q a - A l'aide de la commande echo "une phrase" > fic, vous pouvez écrire le texte *une phrase* dans le fichier :

(a) Créez un répertoire test

(b) Créer un fichier essai dans ce répertoire, et écrivez-y la phrase de votre choix.

(c) Notez les permissions d'accès du répertoire test et du fichier essai, par la commande " ls -l "

(d) Retirez -vous le droit en lecture et en écriture sur le fichier essai.

(e) Vérifiez l'effet obtenu en essayant d'afficher le contenu du fichier sur la fenêtre du terminal (commande *cat*), puis de remplacer ce contenu par une phrase différente (commande *echo*).

(f) Rétablissez le droit en écriture puis remplacez à l'aide de la commande *echo* le contenu du fichier essai par le texte *echo "bienvenu, ceci est un test"* :

— Ajoutez-vous le droit en exécution, et exécutez le fichier essai en tapant ./essai dans le terminal (depuis le répertoire qui le contient). Quel est le problème ?

— Rétablissez enfin le droit en lecture et tentez à nouveau d'exécuter le fichier. Que se passe-t-il ? Que proposez-vous pour obtenir un résultat plus intéressant ?

Corrigé- Exercice 01

1. Création du répertoire et du fichier

   $ *mkdir* test

   $ *cd* test

2. test $ echo 'ma phrase ' > *essai*
3. Notez les permissions d'accès du répertoire test et du fichier *essai* :

   $ *ls* -l allez à la ligne correspond au dossier test
4. Retirez -vous le droit en lecture et en écriture sur le fichier *essai*

   test $ *chmod* u-rw *essai*
5. Pour vérifier l'effet de cette dernière commande, nous testons le droit en lecture et en écriture respectivement par les deux commandes suivantes :

   tester la lecture : test$ *cat essai*

   tester l'écriture : test$ *echo* 'tenter d'écrire sur le fichier essai' > *essai*

6. Rétablissez le droit en écriture puis remplacez à l'aide de la commande echo le contenu du fichier essai par le texte echo "bienvenu, ceci est un test" :

   test $ *chmod* u+w essai

   test $ *echo* 'bienvenu, ceci est un test' > *essai*

   — Ajoutez-vous le droit en exécution, et exécutez le fichier essai en tapant *./essai* dans le terminal (depuis le répertoire qui le contient). Quel est le problème ?

   test $ *chmod* u+x *essai*

   test $ ./essai impossible d'exécuter ce fichier, pourquoi ?

   On ne peut exécuter un fichier sur lequel le droit en lecture a été enlevé.

   — Rétablissez enfin le droit en lecture et tentez à nouveau d'exécuter le fichier. Que se passe-t-il ? Que proposez-vous pour obtenir un résultat plus intéressant ?

   test $ *chmod* u+r *essai*

   test $ *./essai*

   Le fichier est exécutable ayant un contenu texte qui n'est pas interprétable.

   Refaites à nouveau le test en ecrivant dans le fichier un contenu interprétable.

   Exemple : la commande " ls -la ", comme suit :

   test $ *echo* 'ls -la ' > *essai*

   test $ *./essai*

3.5.2 Exercice 02

Questions

Q a - Refaites l'exercice 01 en utilisant des notations octales.

Corrigé- Exercice 02

Q a - Pour utiliser la notation octale, il est obligatoire de commencer par vérifier les droits actuels par la commande $ ls -l, et ceci avant de répondre à chaque question.

3.5.3 Exercice 03

Questions

Q a - **La commande umask**

Définissez la commande *umask*, de manière à ce que les répertoires lors de leur création aient par défaut les droits 750 (rwxr-x–) et les fichiers 640 (rw-r——).

Corrigé- Exercice 03

Q a - Pour calculer le masque, on distingue un fichier d'un répertoire , comme suit :

Pour un fichier	**Pour un répertoire**
masque maximum pour un fichier 666	masque maximum pour un répertoire
rw-rw-rw-	est de 777 rwxrwxrwx
à retirer 026 —-w-rw-	à retirer 027 —-w-rwx
⇒ droit par défaut 640 rw-r——	⇒ droit par défaut 640 rwxr-x–

Soit la commande : *umask* 027 (Pour un fichier retirer 026 ou 027 sont équivalentes)

4 TP. Manipulation des processus

Sommaire

Les premiers ordinateurs ne pouvaient exécuter qu'un seul programme à la fois, et ce programme avait un contrôle total sur le système et accédait à toutes ses ressources. Cependant, avec l'avancée technologique, les systèmes informatiques contemporains sont capables de charger plusieurs programmes en mémoire et de les exécuter simultanément. Cette transformation a nécessité une gestion plus stricte et une séparation plus claire entre les différents programmes. C'est ainsi qu'est née la notion de processus, qui représente un programme en cours d'exécution dans un environnement informatique moderne[6].

4.1 Qu'est ce qu'un processus ?

Le processus est le concept le plus important dans un système d'exploitation. Tout le logiciel d'un ordinateur est organisé en un certain nombre de processus. On appelle processus un objet dynamique correspondant à l'exécution d'un programme ou d'une commande Linux. Cet objet regroupe plusieurs informations et données qui lui sont propres, ainsi que d'autres informations sur son contexte d'exécution. A distinguer un processus d'un programme : Un programme est une suite d'instructions ; c'est du texte : un code statique. Le processus est un concept dynamique.

De manière informelle, un processus est plus que le code du programme (text). Il inclut également l'activité en cours, représentée par la valeur du compteur du programme et le contenu des registres du processeur. Un processus comprend également la pile de processus (stack), qui contient des données temporaires (telles que des paramètres de fonction, des adresses de retour et des variables locales) et des données (data), qui contient des variables globales. Un processus peut également inclure un tas (heap), qui est une mémoire allouée dynamiquement pendant l'exécution du processus [6].

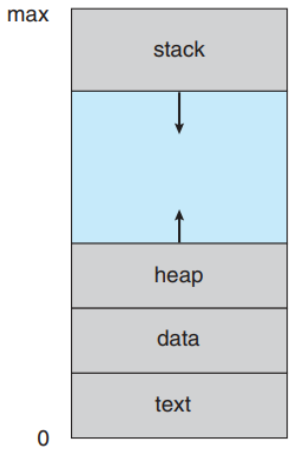


FIGURE 4.1 – Processus en mémoire

Processus ≠ Programme

Chaque processus est représenté dans le SE par un **PCB** (process control block) qui contient ses caractéristiques : on trouve en particulier [1, 8] :

- Son numéro d'identification (PID ou process identifier), qui est un nombre entier positif ;
- L'identifiant du processus qui lui a donné naissance, ou processus parent, appelé PPID (pour parent PID) ;
- Son propriétaire, en général l'utilisateur qui l'a lancé ;
- Eventuellement le terminal dont il dépend (s'il existe) ;
- Son répertoire courant ;
- Sa priorité de travail : elle détermine son ordre d'exécution. Les processus ayant une priorité plus élevée étant exécutés avant ceux à priorité plus faible ;
- Registres ;
- Son temps d'exécution ;
- Liste des fichiers ouverts ;
- Etc.

4.2 Commandes Shell de manipulation de processus

4.2.1 Commande ps

La commande *ps* (process) liste les processus de l'utilisateur.

NB : la commande *top* ou pstree un peu similaire à la commande ps, ces commandes permettent d'afficher en temps réel des informations sur les processus en cours d'exécution. Elle permet de surveiller l'utilisation des ressources système telles que le processeur (CPU), la mémoire, et l'état des processus, et d'afficher une liste dynamique des processus les plus gourmands en ressources.

Vous pouvez consulter la commande suivante pour découvrir toutes les options disponibles pour la commande ps :

```
man ps
```

4.2.2 Etats d'un processus

Un processus possède un cycle de vie allant de sa naissance à sa mort passant par divers phases d'activité et d'attente (voir le cycle de vie d'un processus de la figure 4.2)

L'état d'un processus est défini en partie par l'activité actuelle de ce processus [6].

Il peut se trouver alors dans l'un des trois états principaux :

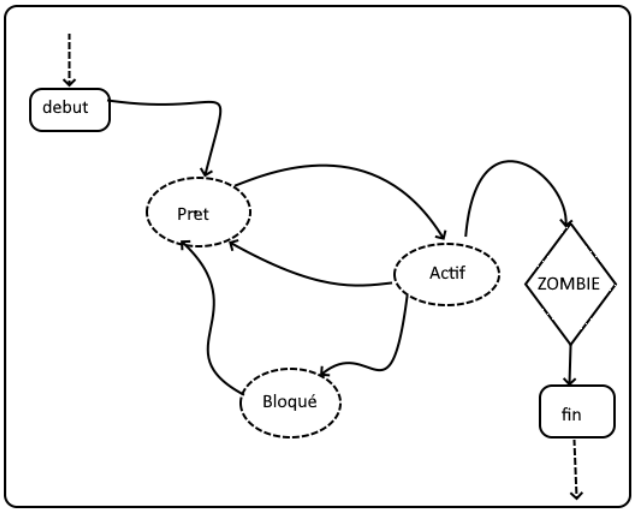


FIGURE 4.2 – Les états probables d'un processus

1. *Actif 'Running'* : il est en cours d'exécution ;
2. *Prêt 'ready'* :il est prêt à être exécuté par le processeur, mais il attend simplement d'être sélectionné par l'ordonnanceur ;
3. *Bloqué 'waiting'* : il attend un événement spécifique avant de pouvoir continuer son exécution, exemple :attente de la fin d'une opération d'entrée/sortie (comme la lecture d'un fichier ou l'attente de données d'un périphérique) ;
4. *Terminé 'terminated'* : il a fini son travail ou a été explicitement arrêté par le système ou l'utilisateur.

NB : Il existe aussi l'état zombie d'un processus. Cet état désigne un processus qui s'est achevé, mais qui dispose toujours d'un identifiant de processus au niveau de la table des processus. On parle aussi de processus défunt (en anglais "defunct" ou identifié par la lettre Z) qui est mentionné à côté du nom de processus avec la commande ps (voir la figure 4.2).

Exemple :

[sabri@home Bureau]$ *ps* -af

UID	PID	PPID	C	STIME	TTY	TIME	CMD
Sabri	1576	3949	0	11 :33	pts/1	00 :00 :00	gedit
Sabri	1705	3949	0	11 :20	pts/1	00 :00 :07	ps -af
Sabri	2000	1	0	11 :00	pts/2	00 :00 :03	wxd

- **L'ID du processus ou PID** : il correspond au numéro d'identification unique utilisé pour faire référence au processus ;
- **L'ID du processus parent ou PPID** : il correspond au numéro du processus (PID) qui a démarré ce processus ;
- **Terminal ou terminal TTY** : il correspond à l'identifiant du terminal associé au processus.

4.2.3 Mode de lancement de processus

Lorsqu'une commande est lancée et que le shell doit attendre la fin de son exécution pour pouvoir lancer une deuxième commande et retrouver le prompt de nouveau, on parle alors d'exécution de **procesus en avant plan** (ou bien **foreground**). Si on ajoute & derrière la commande, ceci permet de lancer un **processus en arrière plan**(ou bien **background**) ; le shell dans ce cas n'attend plus la fin de son exécution. Lorsqu'un utilisateur lance un processus en arrière plan, le Shell affiche entre crochets le numéro de tâche (**job**) et le PID du processus. Le premier processus lancé aura un numéro de tâche = 1.

Exemple :

— Soit la commande *pwd* lancé en arrière plan :

```
pwd &
```

Elle renvoie sur la ligne de commande ceci : [1] 5421

4.2.4 Commande top

La commande ps ne permet pas de suivre en temps réel les processus (affichage figé). Pour avoir un suivi en temps réel, vous pouvez utiliser la commande top.

Exemple :

— Ouvrez 2 terminaux, placez-les l'un à côté de l'autre. Dans l'un des 2 terminaux, exécutez la commande *top*. Observez sur le deuxième terminal ce qui se passe lorsqu'on lance des applications, des processus sur le premier terminal.

— En utilisant la commande *top* dans un terminal, observez ce qui se passe au niveau des processus quand vous ouvrez ou supprimez un onglet dans votre navigateur web.

4.2.5 Contrôler l'exécution d'un processus : La commande kill

Un processus peut recevoir plusieurs signaux engendrés par la commande *kill*, ou bien par l'utilisateur en frappant sur des touches du clavier ou causés par des erreurs matérielles ou logicielles. L'utilisateur peut utiliser des touches pour tuer un processus en avant plan en cliquant sur CTRL^C ou le suspendre en cliquant CTRL^Z.

La commande *kill* permet d'envoyer différents types de signaux à un processus dont on connaît l'identifiant (PID) et ce signal est identifié par un numéro, pour executer cette commande il est obligatoire d'être un utilisateur propriétaire du processus ou bien un utilisateur priviligié [5]. Contrairement à ce que son nom semble indiquer, le rôle de cette commande n'est pas forcément de détruire ou de terminer un processus, mais d'envoyer des signaux aux processus.

La syntaxe d'envoi d'un signal est la suivante :

kill -signal [pid]

Où signal est un numéro ou un nom de signal.

En l'absence de numéro de signal spécifié, le signal envoyé par défaut est TERM [5].

Le signal est l'un des moyens de communication entre les processus. Lorsqu'on envoie un signal à un processus, ce dernier doit l'intercepter et réagir en fonction de celui-ci. Certains signaux peuvent être ignorés. La liste des différents signaux est donné par la commande *kill* (Voir la figure 4.3) ou bien via la commande man (Voir la figure 4.4) :

kill -l

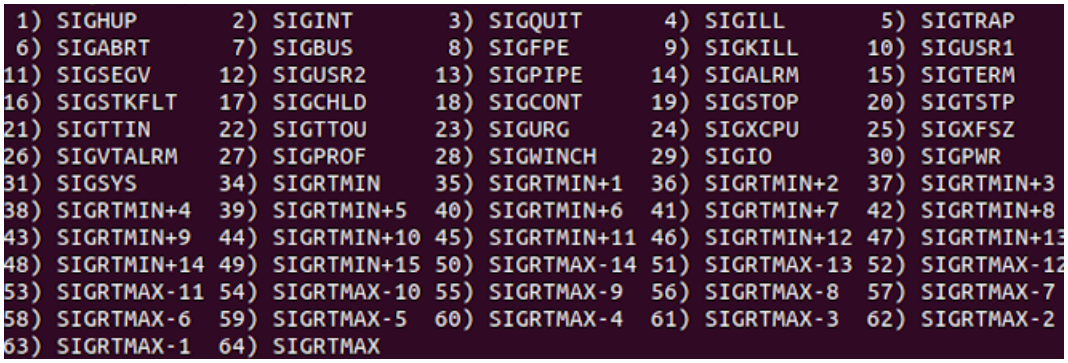
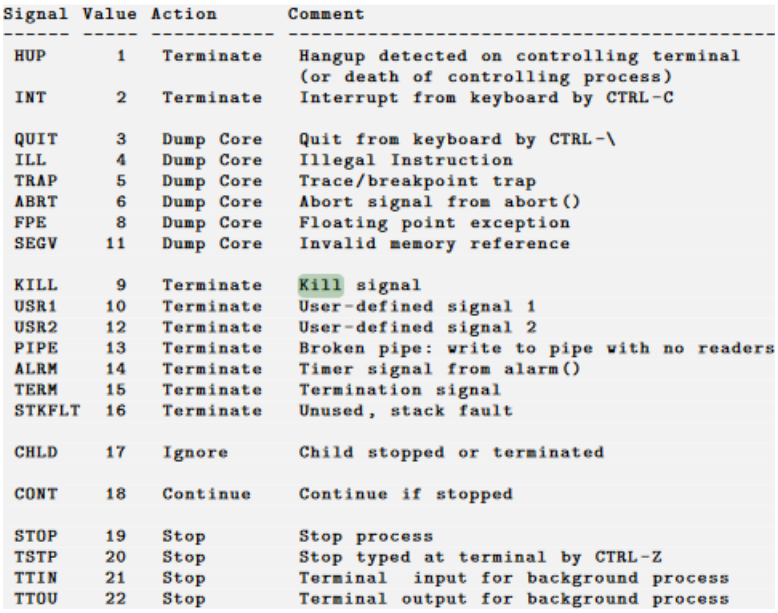


FIGURE 4.3 – La commande kill -l

man 7 signal



```
Signal  Value  Action      Comment
------  -----  ----------  ------------------------------------------
HUP       1    Terminate   Hangup detected on controlling terminal
                           (or death of controlling process)
INT       2    Terminate   Interrupt from keyboard by CTRL-C

QUIT      3    Dump Core   Quit from keyboard by CTRL-\
ILL       4    Dump Core   Illegal Instruction
TRAP      5    Dump Core   Trace/breakpoint trap
ABRT      6    Dump Core   Abort signal from abort()
FPE       8    Dump Core   Floating point exception
SEGV     11    Dump Core   Invalid memory reference

KILL      9    Terminate   Kill signal
USR1     10    Terminate   User-defined signal 1
USR2     12    Terminate   User-defined signal 2
PIPE     13    Terminate   Broken pipe: write to pipe with no readers
ALRM     14    Terminate   Timer signal from alarm()
TERM     15    Terminate   Termination signal
STKFLT   16    Terminate   Unused, stack fault

CHLD     17    Ignore      Child stopped or terminated

CONT     18    Continue    Continue if stopped

STOP     19    Stop        Stop process
TSTP     20    Stop        Stop typed at terminal by CTRL-Z
TTIN     21    Stop        Terminal  input for background process
TTOU     22    Stop        Terminal output for background process
```

FIGURE 4.4 – La commande man 7 signal

Exemples :

— Le signal numéro 9 permet d'arrêter l'exécution du processus dont le Pid= 2000 :

kill -9 2000

— L'option -a de la commande *kill* désigne l'ensemble des processus (en présence de privilège bien évidemment) Pour utiliser cette option, précisez le chemin complet (par exemple : /bin/kill -a gcc).

4.3 Enchaînement séquentiel de processus

Sous Linux, trois caractères permettent d'enchaîner les commandes, et chacun de ces caractères offre un type d'exécution différent lors de l'enchaînement des commandes :

1. Le caractère **;** enchaînera les commandes sans se soucier de la réussite ou de l'échec de chacune. On peut utiliser ces commandes quand nous sommes sûr de leurs réussites ou alors quand leurs bonnes exécutions n'est pas importante.

Exemple :

Si l'on souhaite créer un dossier puis s'y rendre pour y créer un fichier vide, on exécutera le commande suivante :

```
mkdir /home/~/dos1/;cd /home/~/dos1/ ;gedit fich
```

2. Exécution sous condition d'erreur :

```
cmd1 || cmd2 || cmd3 ||... || cmdN
```

Si *cmd1* ne se termine pas correctement, alors *cmd2* est exécuté, et ainsi de suite. Sinon, si *cmd1* s'exécute correctement les autres commandes ne seront pas exécutées.

Le caractère || exécuter une commande uniquement si la commande précédente ne se déroule pas correctement au lieu de mettre fin à toute la ligne de commande.

Exemple :

```
cd /home/~/dos1 || mkdir / /home/~/dos1 && gedit file1
```

3. Exécution sous condition de réussite

```
cmd1 && cmd2 && cmd3 && ... && cmdN
```

Si *cmd1* se termine correctement, alors *cmd2* est exécuté, et ainsi de suite. Sinon, si *cmd1* est exécutée avec erreur les autres commandes ne seront pas exécutées.

Le caractère **&&** Exécuter une commande si et seulement si les commandes précédentes ont réussies.

Exemple :

```
cd /home/~/dos1 && gedit fich
```

Voir si un dossier existe et si c'est le cas y créer un fichier. On peut également vouloir créer le dossier si il n'est pas encore présent.

4.4 Redirections

Un des principaux problèmes pour les débutants sous Unix, c'est d'arriver à gérer correctement les trois symboles :

1. Le symbole > (redirection de sortie) : Le caractère > suivie d'un nom de fichier indique la redirection de la sortie standard vers ce fichier. Ce dernier sera écrasé s'il existe déjà et si la variable booléenne noclobber (noclobber est une variable qui empêche l'écrasement de fichiers existants lors d'une redirection de sortie) n'est pas initialisée ;
2. Le symbole < (redirection d'entrée) : Le caractère < suivie d'un nom de fichier indique la redirection de l'entrée standard à partir de ce fichier ;
3. Le symbole | (pipe) : il sert à rediriger la sortie d'une commande vers l'entrée d'une autre commande, créant ainsi un flux de données entre les commandes.

NB :

— Le caractère 2 > suivie d'un nom de fichier indique la redirection de la sortie d'erreur vers ce fichier.

— Si on double le caractère ≫, l'information ou la redirection sera ajoutée au fichier de sortie.

On peut voir ces symboles comme une écriture sur un fichier pour le supérieur, une lecture à partir d'un fichier pour l'inférieur, un simple tuyau pour le pipe. Ce tuyau servant à relier deux commandes entre elles, c'est-à-dire, la sortie de l'une devient l'entrée de l'autre.

Exemple :

— Les informations données par date vont être placées dans le fichier "file" (si le fichier existe il sera écrasé) via la commande :

$$date > \text{file}$$

— Les informations retournées par who vont être ajoutées au fichier file via la commande :

$$who \gg \text{file}$$

— Déterminer le nombre de fichiers présents dans un répertoire avec la commande :

```
ls -la | wc -l
```

4.5 Communication interprocessus : les tubes

Un tube (ou "pipes") sert à communiquer deux processus. Le principe est que la sortie standard d'un processus est redirigée vers l'entrée d'un tube dont la sortie est dirigée vers l'entrée standard d'un autre processus. Ainsi les deux processus peuvent échanger des données sans passer par un fichier intermédiaire de l'arborescence.

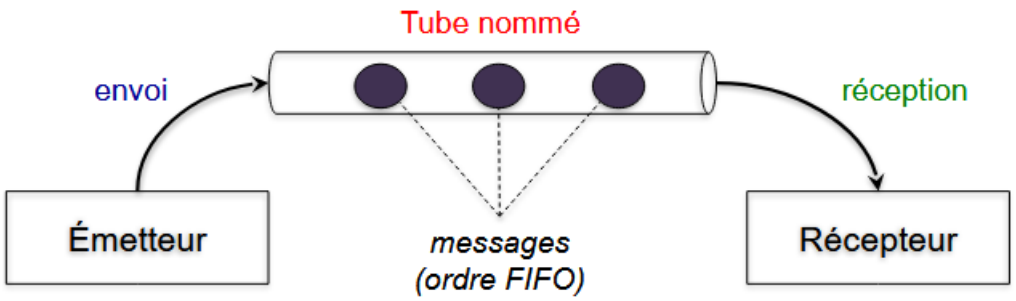


FIGURE 4.5 – Un tube nommé

Créer le pipe

Le lancement concurrent de processus communiquant deux par deux par l'intermédiaire des tubes sera réalisé par une commande de la forme :

```
commande1 | commande2 | ... | commandeN
```

Elle contient le symbole "|" sert à séparer deux commandes.

Exemples :

— Pour vérifier que plusieurs processus existent simultanément, il suffit de lancer la séquence suivante :

```
ps -l | tee /dev/tty | wc -l
```

La commande *tee* permet d'envoyer sur la sortie standard et sur le fichier référencé en paramètre, ce qu'elle lit sur son entrée standard (dans ce cas, le fichier référencé est le fichier écran /dev/tty qui permet de visualiser sur l'écran l'entrée standard de la commande *tee*, qui est en fait, la sortie standard de la commande *ps*).

— Le fichier *test.txt* renferme la liste de tous les fichiers d'un répertoire donné. La sortie de la commande **cat** est utilisée comme entrée pour la commande **grep**. Nous redirigeons ensuite la commande awk, qui affiche la 8^{me} colonne de la sortie filtrée de la commande **grep**.

$ *cat* test.txt |*grep* "file"|*awk* 'print $8'

file1

file2

file3

file4

file5

4.6 Série d'exercices avec corrigés

4.6.1 Exercice 01

Questions

Q a - Lister les processus ;

Q b - Lancez depuis le terminal quelques programme : *xclock*, *xcalc*, *gedit*, un autre terminal (par la commande gnome-terminal) ;

Q c - Sans fermer les programmes que vous venez de lancer, testez la commande ps depuis le terminal ;

Q d - L'option -l de *ps* permet d'afficher plus de détails sur les processus. Testez cette option et repérez les caractéristiques précédemment évoquées. Détaillez, en vous aidant, au besoin du manuel, la signification des colonnes UID, PRI, TT, VSZ, TIME ;

Q e - Comment peut-on afficher la liste de tous les processus du système (notamment ceux dont vous n'êtes pas le propriétaire, et ceux qui ne dépendent pas d'un terminal) ?

Corrigé- Exercice 01

Q a - Utiliser la commande : *ps*

Q b -

```
$ xclock &
$ xcalc &
$ gedit &
$ gnome-terminal
```

Q c - Tester la commande en tapant $ *ps*

Q d - $*ps* -l

Utiliser le manuel de ps pour plus de détails $ *man* ps (vous trouvez l'explication de chaque paramètre).

Q e - $ *ps* -u "nom ou UID du proprietaire" cette commande liste uniquement les processus appartenant au propriétaire dont le nom ou l'UID est donné en paramètre pour afficher la totalité des processus appliquer la commande *ps -ef*.

4.6.2 Exercice 02

Questions

Q a - Créer un nouveau fichier avec la commande *gedit* ;

Q b - Identifier son PID à l'aide de la commande *ps* ;

Q c - Bloquer cette application ;

Q d - Reprendre l'application.

Corrigé- Exercice 02

Q a - Créer un nouveau fichier avec gedit

gedit nouveau.txt &

Q b - Identifier son PID à l'aide de la commande *ps*

ps -el

Q c - Bloquer cette application

Kill -l (chercher le signal adéquat)

Kill -n°signal PID

Q d - Reprendre l'application

Kill -l (chercher le signal adéquat) puis appliquer la commande suivante

Kill -n°signal PID

pour reprendre une application suspendu, le signal a appliquer est SIGCONT dont le numéro est 18.

4.6.3 Exercice 03

Questions

Q a - Ecrire un programme script shell qui affiche la liste des processus triées par ordre alphabétique.

Corrigé- Exercice 03

Q a - Le programme est le suivant :

#!/bin/bash

Cette ligne est utilisée en début de script pour indiquer quel interpréteur doit être utilisé pour exécuter le script. *ps | sort*

Explication : La commande sort permet de trier le résultat affiché par la commande ps selon un ordre spécifique (ici, par ordre alphabétique).

4.6.4 Exercice 04

Questions

Que font les commandes suivantes :

Q a - *ls* | *wc* -l

Q b - *who*|*sort*

Q c - *ls* -l /home | *more*

Q d - *ls* -l /bin | *grep* "d'

Corrigé- Exercice 04

Q a - La commande compte le nombre de lignes données par *ls* (le nombre du fichiers du répertoire courant).

Q b - La commande permet de trier alphabétiquement la liste des utilisateurs connectés.

Q c - Grâce à un pipe, la commande *more* peut recevoir en entrée les données envoyées par une autre commande vers la sortie standard. Par exemple, on peut l'utiliser pour afficher le résultat de la commande *ls* -l /home page par page.

Q d - La commande permet d'envoyer la sortie standard de la commande ls -l /bin (la liste des fichiers et des répertoires du répertoire /bin avec un format détaillé). Ensuite, la commande grep "d' recherche dans le résultat de *ls* -l /bin les lignes qui commencent par d, donc ce sont les lignes qui correspondent à des répertoires.

4.6.5 Exercice 05

Questions

Q a - Affichez une liste rafraîchie en temps réel des processus s'exécutant sur votre machine.

Q b - A quoi sert la commande *top* . Affichez l'aide incluse avec la commande *top*.

Q c - Envoyez le signal 2 (SIGINT) au processus "*top*" sans quitter la commande. Que se passe-t-il ?

Corrigé- Exercice 05

Q a - Utiliser la commande *top*.

Q b - Le rôle de la commande *top* :

La commande *top* affiche en temps réel un tableau des processus en cours, triés par leur utilisation décroissante du CPU.

Le *help* de cette commande est visualisé par la commande man : *man* top

Q c - *kill* -2 PID(top)

5 TP. Initiation à la Programmation Shell

Sommaire

Il est possible de mettre une liste de commandes shell dans un fichier puis de démarrer un shell avec ce fichier comme entrée standard. Le (deuxième) shell les traite simplement dans l'ordre, comme il le ferait avec des commandes tapées sur le clavier. Les fichiers contenant des commandes shell sont appelés **scripts shell**.

Les scripts Shell peuvent attribuer des valeurs aux variables Shell, puis les lire plus tard. Ils peuvent également avoir des paramètres et utiliser des constructions if, for, while et case. Ainsi, un script shell est en réalité un programme écrit en langage shell.

Pourquoi faire des scripts ? Les langages de scripts shell sont beaucoup mieux adaptés aux tâches de gestion d'un système servant à :

— Automatiser des tâches ;

— Débogage des tâches ;

— Personnalisation de l'environnement utilisateur ;

— Développement rapide de prototypes et de scripts utilitaires ;

— Les scripts bien faits sont donc plus clairs et plus faciles à maintenir ;

— Etc.

5.1 Commandes internes et externes

Une commande Unix peut appartenir à l'une des deux catégories : interne et externe.

La commande ***file*** donne une indication sur le type de données contenues dans un fichier (voir des exemples sur la figure 5.1).

L'argument de la commande *file* est un nom de fichier exprimé en relatif ou bien en absolu [3].

```
sabri@sabri-X555LAB:~$ file for.sh
for.sh: POSIX shell script, UTF-8 Unicode text executable
sabri@sabri-X555LAB:~$ file exo5.sh
exo5.sh: POSIX shell script, ASCII text executable
sabri@sabri-X555LAB:~$ file Bureau/
Bureau/: directory
sabri@sabri-X555LAB:~$
```

FIGURE 5.1 – Tester la commande file

5.1.1 Commande type

La commande *type* affichera le type de commande pour un mot donné saisi sur la ligne de commande. Les types de commandes sont répertoriés comme suit [2] :

- Alias
- Function
- Shell built in
- Keyword
- File

La commande *type* nous indique si une commande possède une implémentation interne ou bien externe [2] (voir les exemples sur la figure 5.2).

```
sabri@sabri-X555LAB:~$ type echo
echo est une primitive du shell
sabri@sabri-X555LAB:~$ type cd
cd est une primitive du shell
sabri@sabri-X555LAB:~$ type mkdir
mkdir est /usr/bin/mkdir
sabri@sabri-X555LAB:~$ type rmdir
rmdir est /usr/bin/rmdir
sabri@sabri-X555LAB:~$ type ls
ls est un alias vers « ls --color=auto »
sabri@sabri-X555LAB:~$ type gedit
gedit est /usr/bin/gedit
```

FIGURE 5.2 – Tester la commande type

5.1.2 Commandes internes

Une commande interne est une commande intégré au shell et ce dernier se charge d'exécuter l'action [3].

Exemples de commandes internes : *cd* , *echo* , *for* , *pwd*, etc.

5.1.3 Commandes externes

Une commande externe est un fichier localisé dans l'arborescence.

Exemples de commandes externes : *ls*, *mkdir*, *cat*, *sleep*, les fichiers scripts, etc.

La localisation du code d'une commande externe doit être connue du shell pour qu'il puisse exécuter cette commande. Pour cela, le bash utilise la valeur de sa variable prédéfinie PATH.

Celle-ci est une variable d'environnement qui contient une liste de chemins, séparés par des deux-points (:) et qui indique au système où chercher les exécutables lors de l'exécution de commandes.

Exemples :

— Lorsqu'un utilisateur lance la commande *ls*, le shell demande au noyau Unix de charger en mémoire le fichier **/usr/bin/ls**.

Si nous devons simplement saisir le résultat, nous pouvons utiliser l'option -t. Ceci est utile lorsque nous devons tester le type de commande à partir d'un script et que nous avons uniquement besoin que le type soit renvoyé. Considérez les commandes suivantes :

```
$ type -t ls
```

Cette commande indique que *ls* est un fichier exécutable.

```
type -t alias
```

Cette commande indique que *alias* est une commande intégrée au shell.

— **Délai d'attente**

La commande *sleep* permet d'attendre le nombre de see secondes indiquées. Le script est interrompu durant ce temps. Le nombre de secondes est un entier compris entre 0 et 4 milliards (136 ans). Taper la commande :

```
$ sleep 10
```

5.2 Commandes d'affichage

- La commande *echo* permet de réaliser des affichages à l'écran.
- Le caractère `\n` sert à provoquer un saut de ligne.

Exemples :

Voici un exemple d'application de la commande *echo* :

```
$ echo "Voici un saut de ligne 1 \n Voici un deuxième saut de ligne \n
```

NB : L'utilisation des quotes est obligatoire. Dans l'exemple suivant, le caractère de saut de ligne n'est reconnu que lorsqu'on cadre le text par des quotes :

$ *echo* ligne1 \n ligne2

ligne1 \n ligne2

$ *echo* "ligne1 \n ligne2"

ligne1

ligne2

5.3 Substitutions

Sous Linux, de nombreuses commandes prennent des noms de fichier en argument, elles peuvent être cités littéralement ou être spécifiés de manière plus générique. Le shell propose un certain nombre de caractères spéciaux qui permettent de fabriquer des expressions utilisées comme modèles de noms de fichier [3].

5.3.1 Expressions basiques

1. **Le caractère ***

   Il représente une suite de caractères quelconque.

   Exemples :

   — Afficher tous les noms de fichier de programmes C :

   ```
   ls *.c
   ```

   — Afficher tous les noms de fichier qui commencent par la lettre b :

   ```
   ls b*
   ```

2. **Le caractère ?**

   ll représente un caractère quelconque.

   Exemples :

   — Copier tout les fichiers de taille 6 vers le dossier *Rep* se trouvant sur le Bureau :

   ```
   cp ? ? ? ? ? ? /home/sabri/Bureau/Rep
   ```

   ```
   cp ? ? ? ? ? ? /home/sabri/Bureau/Rep
   ```

   — Afficher tous les noms de fichier qui porte une extension composée d'un seul caractère :

   ```
   *. ?
   ```

3. **Les caractères [ ]**

   Ils permettent de spécifier la liste des caractères que l'on attends à une position bien précise dans le nom du fichier.

   Exemples :

   — Fichiers dont le nom commence soit par a ou h et se termine par le caractère"." suivi d'une minuscule quelconque :

   ```
   ls [ah]*.[a-z]
   ```

   — Fichiers dont le nom ne commençant pas par une minuscule :

   ```
   ls [!a-z]*
   ```

5.3.2 Expressions complexes

1. **?(expression)**

   L'expression sera présente 0 ou 1 fois.

   Exemple :

   — Fichiers dont le nom commence par "file" suivi de 0 ou 1 occurrence de "90", suivi de ".log" :

   ls file ?(90) .log

2. ***(expression)**

   L'expression sera présente entre 0 et n fois.

   Exemple :

   — Fichiers dont le nom commence par "file", suivi de 0 à n occurrences de "90", suivi de ".log" :

   ls file*(90).log

3. **+(expression)**

   L'expression sera présente entre 1 et n fois.

   Exemple :

   — Fichiers dont le nom commence par "file",suivi d'au moins 1 occurrence de "90", suivi de ".log" :

   ls file+(90).log

4. **@(expression)**

   L'expression sera présente exactement 1 fois.

   Exemple :

   — Fichiers dont le nom commence par "file", suivi exactement d'une occurrence de "90", suivi de ".log" :

   ls file@(90).log

5. **!(expression)**

   L'expression ne sera pas présente.

   Exemple :

   — Fichiers dont le nom commence par "file", suivi d'une expression qui n'est pas la chaîne "90", suivi de ".log" :

   ls file!(90).log

6. **Les alternatives**

   Un barre verticale à l'intérieur d'une expression complexe prend le sens de "ou bien".

   ?(expression | expression | ...)

   *(expression | expression | ...)

   +(expression | expression | ...)

   @(expression | expression | ...)

   !(expression | expression | ...)

   Exemples :

   — Fichiers dont le nom commence par "file", suivi de "90" ou "99",suivi de ".log" :

   ls file@(90|99).log

   — Fichiers dont le nom commence par "file", suivi de 1 à n occurrences de "90" ou de "99" ,suivi de ".log" :

   ls file+(90|99).log

5.4 Redirections

Les redirections sont couramment utilisées dans les commandes Unix. Elles permettent de récupérer le résultat d'une ou de plusieurs commandes dans un fichier ou au contraire de faire lire un fichier à une commande. Les redirections sont mises en place par le shell [2].

5.4.1 Entrée et sorties standard des processus

La redirection de l'entrée standard concerne les commandes qui utilisent le descripteur 0 qui est l'entrée standard (souvent le clavier). [2].

Exemples :

— La commande mail lit l'entrée standard jusqu'à réception d'une fin de fichier. Les données saisies seront envoyées dans la boîte aux lettres de l'utilisateur spécifié en paramètre.

```
$ mail sabri
```

```
EXAMEN SE demain à 13 heures Bloc1    (Entrée standard)
SABRI                                 (Entrée standard)
```

Si l'on souhaite faire lire à la commande mail, non plus le clavier, mais le contenu d'un fichier, il suffit de connecter le descripteur 0 sur le fichier désiré comme suit :

```
$ cat message
```

```
EXAMEN SE demain à 13 heures Bloc1
SABRI
```

```
$ mail sabri < message
```

5.4.2 Redirection des sorties en écriture

La redirection en écriture permet d'envoyer les affichages liés à un descripteur particulier, non plus sur le terminal, mais dans un fichier [2].

Exemples :

— **Une simple redirection** : où nom du fichier est exprimé en relatif ou en absolu :

```
$ commande > fichier
```

Soit pour récupérer le résultat de la commande ls dans le fichier TestLs, nous appliquons la commande :

```
$ ls > TestLs
```

NB : Si le fichier TestLs n'existe pas, il est créé. Si le fichier existe déjà, il est écrasé.

— **Une double redirection** : Le résultat d'une commande est concaténé au contenu d'un autre fichier existant.

Soit pour écrire en fin de fichier TestLs, nous appliquons la commande :

```
$ ls >> TestLs
```

NB : Si le fichier n'existe pas, il est créé. Si le fichier existe déjà, il est ouvert en mode ajout

5.5 Variables réservées du shell

Certaines variables sont définies dans un environnement du shell, elles portent des informations nécessaires au fonctionnement de l'interpréteur ou lancées à partir de celui-ci :

∘ **HOME :** contient le chemin absolu du répertoire de connexion de l'utilisateur ;

Exemple :

```
$ echo "le répertoire de connexion est : $HOME"
```

Le répertoire de connexion est : /home/etudiant

- **LOGNAME :** contient le nom de connexion de l'utilisateur ;
- **PATH :** contient la liste des répertoires contenant des exécutables séparés par ' :'. Ces répertoires seront parcourus par ordre à la recherche d'une commande externe ;
- **SHELL :** contient le chemin d'accès absolu des fichiers programmes du shell ;
- **ENV :** permet d'afficher l'ensemble des variables d'environnement pour le shell actif.

Pour l'affichage du contenu d'une variable, le caractère spécial "$" est utilisé.

Exemple :

```
$ echo $HOME
```

5.6 Écriture et lancement d'un script shell

5.6.1 Structure et exécution d'un script

Pour que le shell sache comment l'interpréter :

— Un script shell doit commencer par la ligne : `#!/bin/sh`

Cette ligne indique que ce script doit être exécuté par le shell **sh** dont on indique le chemin d'accès.

— Les instructions et commandes sont regroupées au sein d'un script : Une ligne se compose de commandes internes ou externes, de commentaires ou vide ;

— Chaque ligne sera lue une à une et exécutée ;

— Plusieurs instructions par lignes sont possibles :

  — Séparées par le " ; " ou

  — Liées conditionnellement par " && " ou " || ".

Extension de scripts Shell

Les scripts se terminent par :

— " .sh " (pour le Bourne Shell)

— " .ksh " (pour le Korn Shell)

— " .csh " (pour le C Shell)

Exemple : . Csh

— Syntaxe similaire au langage C

— Le shell de connexion execute des commandes à partir des fichier .cshrc et .login

5.6.2 Exécution d'un script

Utilisez la commande *chmod* pour autoriser uniquement le propriétaire à exécuter le fichier. Pour rendre un fichier exécutable, tapez la commande :

```
$ chmod u+x fichier.sh
```

```
$ ./fichier.sh
```

Commençons par ce qu'il y a de plus simple : afficher un message d'accueil ! Hello World !. Le fichier script qui permet de répondre à notre requête est le suivant :

```
#!/bin/sh
# Fichier "helloworld"
echo "Hello world"
```

5.6.3 Variables spéciales $

Il existe plusieurs variables spéciales, souvent préfixées par $, qui fournissent des informations utiles sur l'environnement d'exécution. Ci-dessous, quelques exemples :

$?	référence la valeur retournée par la dernière commande exécutée
$$	référence le numéro du processus (PID) du shell actif
$!	référence le numéro du processus (PID) du dernier processus lancé en arrière plan
$0	référence le nom de procédure de commande
$1...$9	référence la valeur de n^{ime} paramètre
$#	référence le nombre de paramètres d'un fichier de commandes
$*	référence la liste de tous les paramètres $1...$9, est utilisé si l'on veut substituer tous les arguments sauf le nom de commande ($0)
$@	référence la liste des paramètres sous la forme "$1", "$2",..., "$9"

Les paramètres de position sont des variables spéciales utilisées lors d'un passage de paramètres à un script. Ces variables sont très utiles pour gérer les arguments et les erreurs dans les scripts. Voir un exemple sur la figure 5.3.

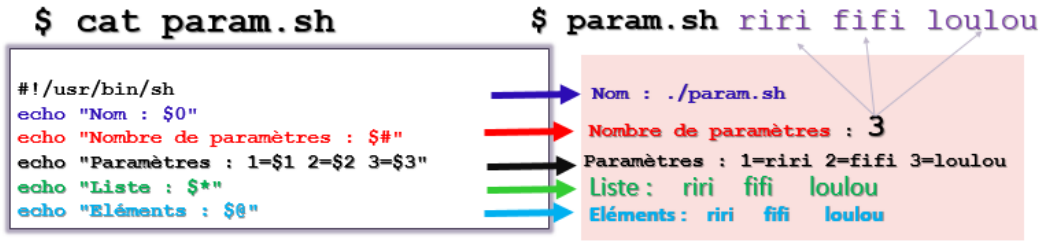


FIGURE 5.3 – Paramètres de position

NB : la différence entre3 $* et $@ en Bash : La différence apparaît avec les guillemets : "$*" regroupe tous les arguments en une seule chaîne, tandis que "$@" conserve chaque argument séparément. Ainsi, "$@" est à privilégier pour traiter les arguments un par un.

Exemple d'un programme Shell :

La commande *test* permet de tester l'existence de fichiers comportant certaines caractéristiques (fichier classique, répertoire, vide, non-vide, etc.). Mais l'option "-e" qui permet de tester la seule existence d'un fichier quel que soit son type n'existe pas en Bourne Shell.

```
# Programme qui affiche si le fichier demandé existe ou n'existe pas
# Usage : prog fichier
#!/bin/ksh
test -e "$1" && echo "Le fichier $1 existe" || echo "Le fichier $1 n'existe pas"
```

5.6.4 Quelques structures de contrôle

if ... then ... else

Syntaxe **if** <commandes_condition> **then**
<commandes exécutées si condition réalisée>
else <commandes exécutées si dernière condition pas réalisée>
fi

Exemple : tester si la variable $1 est un répertoire ou bien un fichier.

```
if test -d $1 then
echo "$1 est un répertoire"
elif test -f $1 then
echo "$1 est un fichier"
```

Choix multiples case

Le *case* gère les choix de l'utilisateur. Chaque option correspond à une action spécifique, voici sa syntaxe :

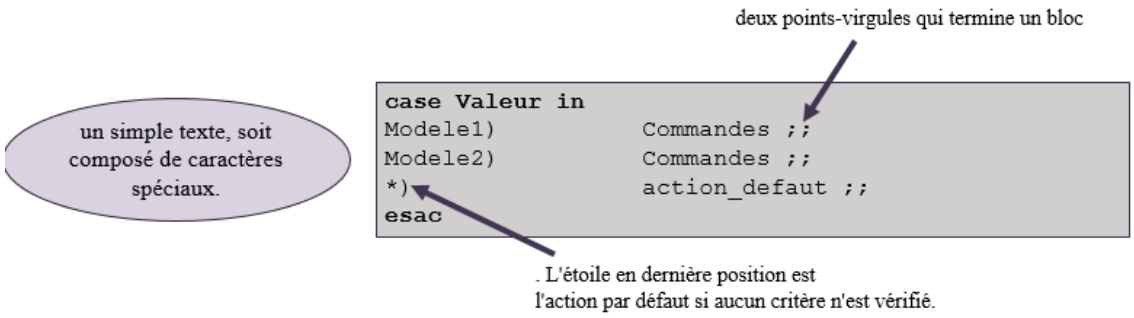


FIGURE 5.4 – Choix multiples case

Exemple :

case $langue **in**

français echo "Bonjour";;

anglais echo "Hello";;

italien echo "Bongiorno";;

esac

5.7 Série d'exercices avec corrigés

5.7.1 Exercice 01

Questions

Q a - Écrire un script qui renvoie le message "répertoire disponible" si l'argument de la commande correspond justement au nom d'un répertoire disponible.

Q b - Écrire un script qui permet d'afficher les nombres de 1 à jusqu'à la valeur passée en paramètre du script et ceci à l'aide d'une boucle while

Corrigé- Exercice 01

Q a - #!/bin/bash

```
# -d est une option de test qui retourne vrai si le chemin correspond à un répertoire
#Affiche le message répertoire disponible si le répertoire donné existe
test -d $1 && echo répertoire disponible
```

Q b - #!/bin/bash

```
i = 1
while test $i -le $1 ; do
echo $i ;
let i=$i+1 ;
done
```

5.7.2 Exercice 02

Questions

Q a - Soit la commande **read** qui lit son entrée standard et affecte les valeurs saisies dans la ou les variables passées en argument. Écrire un programme shell qui affiche ligne par ligne le contenu d'un fichier dont le nom est fourni en paramètre en se servant de la commande **read**.

Q b - Que fait le programme shell suivant :

```
#!/bin/sh
w=‘who | grep $1‘
if [ -z "$w" ] ; then echo "$1 n'est pas ....." ; fi
```

Corrigé- Exercice 02

Q a -
```
# cette ligne indique quel interpréteur utiliser pour exécuter le script.
#!/bin/sh
# Redirection de l'entrée standard : cette ligne redirige l'entrée standard (stdin)
du script vers le fichier dont le nom est passé en premier argument ($1).
exec < $1
#la boucle lit ligne par ligne depuis l'entrée standard (donc depuis le fichier redirigé).
À chaque tour de boucle, le contenu de la ligne lue est stocké dans la variable ligne.
while read ligne ; do
# Affichage de la ligne lue, précédée du caractère >.
echo ">$ligne"
done
```

Q b - Ce programme détermine si l'utilisateur dont le nom est donné comme argument est connecté. Nous remplaçons les points dans le message de echo par ceci : "l'utilisateur $1 n'est pas connecté"

Bibliographie

[1] H. Remzi A. Dusseau and C. Andrea A. Dusseau, *Operating Systems : Three Easy Pieces*. CreateSpace Independent Publishing Platform, North Charleston, SC, USA, 2018.

[2] A. Mallett, *Mastering Linux Shell Scripting*. Community experience distilled. Packt Publishing, 2015.

[3] C.D. Rémy, *Programmation shell sous Unix/Linux : sh (Bourne), Ksh, bash*. Ressources informatiques. ENI, 2008.

[4] W. Shotts, *The Linux Command Line : A Complete Introduction*. No Starch Press, San Francisco, CA, 2nd edition, 2019.

[5] E. Siever, S. Figgins, R. Love, and A. Robbins, *Linux in a Nutshell : A Desktop Quick Reference*. Nutshell. O'Reilly Media, 2009.

[6] A. Silberschatz, *Operating System Concepts*. Wiley (edition 10), 2018.

[7] S. Sylvain and M. Fitzgibbon, *Recueil de fiches explicatives de licences libres*. INRIA, version 2.0 edition, [en ligne]. [Consulté le 15-04-2025].

[8] S. A. Tanenbaum and H. Bos, *Modern Operating Systems*. Pearson, Boston, MA, 4 edition, 2014.