Linux
https://www.linuxfoundation.org/

Une introduction à Linux et à la ligne de commande

Sébastien LE ROUX sebastien.leroux@ipcms.unistra.fr

INSTITUT DE PHYSIQUE ET DE CHIMIE DES MATÉRIAUX DE STRASBOURG,
DÉPARTEMENT DES MATÉRIAUX ORGANIQUES,
23 RUE DU LOESS, BP43,
F-67034 STRASBOURG CEDEX 2, FRANCE

NOVEMBER 28, 2025

Contents

Licence

Système d'exploitation

Un système d'exploitation (SE), est un logiciel central qui gère le matériel informatique, les ressources logicielles et fournit des services communs pour les programmes informatiques.

En gros, le SE est le premier programme qui démarre lorsque vous allumez votre ordinateur, sans le SE, il est impossible d'utiliser d'autres programmes.
Il existe deux grandes familles de SE, la famille MS-DOS et la famille UNIX. MS-DOS est l'ancêtre de Windows et UNIX est l'ancêtre de MacOSX, BSD et plus important encore Linux:

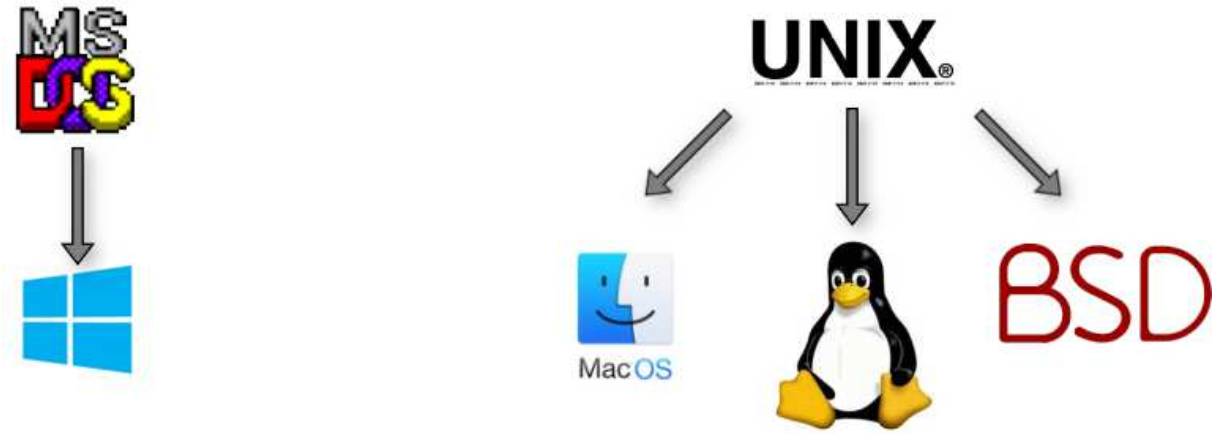


Figure 2.1: L'arbre des principales familles de systèmes d'exploitation.

Comme illustré sur la figure [Fig. 2.1], MacOS (OSX, iOS), Linux et BSD (Berkley Software Distribution, open source UNIX) sont cousins.

2.1 Quelques chiffres sur les SE

2.1.1 Ordinateurs personnels

Table 2.1: Répartition globale sur les ordinateurs personnels.

2.1.2 Téléphones et appareils mobiles

Table 2.2: Répartition globale sur les téléphones et appareils mobiles.

2.1.3 Linux et Internet

Au moment où j'écris ce document, le web mondial est constitué d'un ensemble de plus de **270 000 000** serveurs web répartis sur environ 18 000 000 de serveurs physiques situés partout dans le monde. Ces 18 000 000 d'ordinateurs fonctionnent 24h sur 24, 365 jours par an pour garantir qu'Internet soit l'Internet que nous connaissons aujourd'hui.
Voici quelques chiffres pour illustrer l'importance de Linux dans cet Internet mondial:

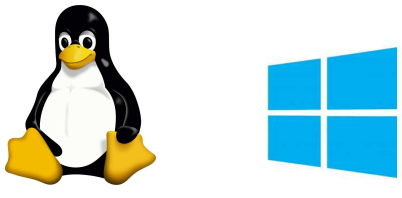


Table 2.3: Répartition globale sur les serveurs web d'Internet.

Quelques chiffres supplémentaires pour souligner encore d'avantage l'importance majeure de Linux dans ce domaine:

- Pour le top `1 000 000`, c'est-à-dire les `1 000 000` de serveurs web les plus utilisés et les plus importants sur Internet, Linux représente `96.5 %` des SE.
- Pour l'ensemble des systèmes de cloud computing, Linux représente `90 %` des SE.
- Les centres de calcul à haute performance (HPC, supers calculateurs) utilisent exclusivement Linux.

2.1.4 Tout type d'appareils

`40.5 %` `34.2 %` `22.3 %` `1 %`

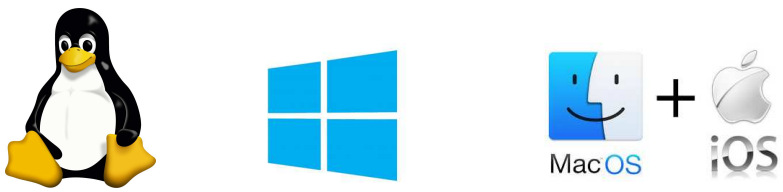
Table 2.4: Répartition globale sur tous types d'appareil.

Comme vous pouvez le voir dans le tableau [Tab. 2.4], et cela ne devrait pas vous surprendre, Android est le système d'exploitation le plus populaire. Cela semble évident compte tenu du nombre de smartphones et d'appareils mobiles.

Ce qui pourrait vous surprendre en revanche c'est qu'Android est en fait une distribution Linux (voir la section [Sec. 4.1] pour plus de détails) !

Si vous utilisez un téléphone mobile sous Android, sachez que vous utilisez donc déjà Linux !!!

Par conséquent la répartition des SE, tout type d'appareil confondus est:



`41.5 %` `34.2 %` `22.3 %`

Table 2.5: La véritable répartition des SE, en considérant tous les appareils.

Oui, la vérité est que Linux est déjà le système d'exploitation le plus populaire, bienvenue dans le monde des logiciels libres !

Logiciels libres ou open source

Adapté de la page web Wikipedia

Un logiciel libre, ou open source, est un logiciel informatique qui est publié sous une licence qui permet aux utilisateurs de l'utiliser, de l'étudier, de le modifier et de le redistribuer à quiconque et pour n'importe quel but. Un logiciel libre peut être développé de manière collaborative et publique, ce qui en fait un exemple notable de collaboration ouverte, ce qui signifie que tout utilisateur capable peut participer en ligne au développement, ce qui rend le nombre de contributeurs potentiellement infini. De plus la capacité à examiner le code facilite la confiance publique dans le logiciel en question, et dans les logiciels libres en général.

3.1 Un peu d'histoire

- 1983: Création de GNU et de la licence GPL par Richard M. Stallman (**rms**)

- 1985: Création de la Free Software Foundation par RMS

- 1991: Linux est développé par Linus Torvalds

- 2004: Les logiciels Libres entrent officiellement au patrimoine mondial de l'UNESCO

3.2 Les quatre libertés essentielles du logiciel libre

Par la suite, le mot "libre" ne fait pas référence au prix, il fait référence à la liberté.
La première définition publiée par la FSF en février 1986 comportait deux points:

- La liberté de copier un programme et de le redistribuer à vos voisins, afin qu'ils puissent l'utiliser comme vous.
- La liberté de modifier un programme, afin que vous puissiez le contrôler au lieu qu'il vous contrôle ; pour cela, le code source doit être mis à votre disposition.

En 1996, lorsque le site web https://www.gnu.org a été lancé, le "logiciel libre" a été défini en faisant référence à "trois niveaux de liberté" en ajoutant une mention explicite de la liberté d'étudier le logiciel (qui pouvait être lu dans la définition à deux points comme faisant partie de la liberté de modifier le programme).

Enfin, une autre liberté a été ajoutée, pour dire explicitement que les utilisateurs devraient être en mesure de lancer le programme. Les libertés existantes étaient déjà numérotées de un à trois, mais cette liberté devrait venir avant les autres, elle a donc été ajoutée comme "liberté zéro".
La définition moderne définit le logiciel libre par le fait que le destinataire a les 4 libertés suivantes:

0. La liberté d'exécuter le programme comme vous le souhaitez, pour n'importe quel but (liberté 0).

1. La liberté d'étudier comment fonctionne le programme, et de le modifier pour qu'il fasse votre travail informatique comme vous le souhaitez (liberté 1). L'accès au code source est une condition préalable.

2. La liberté de redistribuer des copies afin de pouvoir aider votre voisin (liberté 2).

3. La liberté de distribuer des copies de vos versions modifiées à d'autres (liberté 3). En faisant cela, vous pouvez donner à l'ensemble de la communauté une chance de bénéficier de vos modifications. L'accès au code source est une condition préalable.

Les libertés **1** et **2** nécessitent que le code source soit disponible car étudier et modifier un logiciel sans son code source est très, voir trop, complexe.
Un logiciel propriétaire peu probablement vous donner accès à l'une de ces libertés, en particulier la liberté 0, si vous payez et sous certaines conditions.
Un logiciel open source / libre vous donnera accès à la liberté 0 et/ou 1, et/ou 2, et/ou 3, selon la licence du logiciel.
En effet, comme tout autre produit, un logiciel libre doit être protégé par une licence, et il existe de nombreuses licences de logiciels libres:

- les licences GPL
- les licences BSD
- la licence Apache
- la licence Creative Commons
- etc...

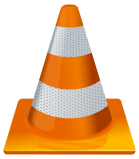
3.3 Logiciels libres célèbres

Voici quelques logiciels libres célèbres:

- Bureautique:



- Multimédia:


- Systèmes d'exploitation:
  - Distributions Linux:


  - BSD (UNIX open source):

- Gestionnaires de bureau:


- Internet:


Linux

Figure 4.1: Tux, la mascotte officielle du noyau Linux.

Adapté de la page web Wikipedia

Linux est une famille de systèmes d'exploitation libres, open source, Unix-like basés sur le noyau Linux, un noyau publié pour la première fois le 17 septembre 1991 par Linus Torvalds.

Linux a été développé à l'origine pour les ordinateurs personnels basés sur l'architecture Intel x86, mais a depuis été porté sur plus de plates-formes que tout autre système d'exploitation.
En raison de sa domination sur les smartphones, Linux a également la plus grande base d'installation de tous les systèmes d'exploitation généraux.
Linux s'exécute également sur les systèmes embarqués, c'est-à-dire les appareils dont le système d'exploitation est généralement intégré au micrologiciel et est fortement adapté au système. Cela inclut les routeurs, les contrôles d'automatisation, la technologie domotique, les téléviseurs, les automobiles (par exemple, Tesla, Audi, Mercedes-Benz, Hyundai et Toyota utilisent tous Linux), les enregistreurs de vidéo numérique, les consoles de jeux vidéo et les montres intelligentes.

Linux est l'un des exemples les plus éminents de collaboration dans les logiciels libres. Le code source peut être utilisé, modifié et distribué commercialement ou non par quiconque sous les termes de ses licences respectives, telles que la licence GNU General Public License.

Greg Kroah-Hartman est le responsable principal du noyau Linux et guide son développement. William John Sullivan est le directeur exécutif de la Free Software Foundation, qui à son tour soutient les composants GNU. Enfin, plusieurs individus et entreprises développent des composants non-GNU tiers. Ces composants tiers constituent un vaste corps de travail et peuvent inclure à la fois des modules de noyau et des applications et des bibliothèques utilisateur.
Les fournisseurs de Linux et les communautés combinent et distribuent le noyau, les composants GNU et les composants non-GNU, avec des logiciels de gestion de packages supplémentaires sous la forme de ce qui est communément appelé distributions Linux.

4.1 Distributions Linux

Adapté de la page web Wikipedia

Une distribution Linux, ou GNU/Linux, (souvent abrégée en distro) est un système d'exploitation constitué d'une collection de logiciels basée sur le noyau Linux et, souvent, d'un système de gestion de packages. Les utilisateurs de Linux obtiennent généralement leur système d'exploitation en téléchargeant l'une des distributions Linux, qui sont disponibles pour une large gamme de systèmes allant des appareils embarqués (par exemple, OpenWrt) aux ordinateurs personnels (par exemple, Linux Mint) jusqu'aux supercalculateurs les plus puissants (par exemple, Rocks Cluster Distribution).
Une distribution Linux typique comprend un noyau Linux, des outils et des bibliothèques GNU, des logiciels supplémentaires, des documents, un système de fenêtrage (le plus courant étant le système de fenêtrage X, ou, plus récemment, Wayland), un gestionnaire de fenêtres et un environnement de bureau.
La plupart des logiciels inclus sont libres et open source, disponibles à la fois sous forme de binaires compilés et sous forme de code source, permettant des modifications du logiciel d'origine. Généralement, les distributions Linux incluent également des logiciels propriétaires qui ne sont pas disponibles sous forme de code source, tels que des blobs binaires nécessaires pour certains pilotes de périphériques.

Une distribution Linux peut également être décrite comme un ensemble particulier de logiciels d'application et d'utilitaires (des utilitaires GNU et des bibliothèques, par exemple), emballés avec le noyau Linux de telle sorte que ses capacités répondent aux besoins de nombreux utilisateurs. Le logiciel est généralement adapté à la distribution et emballé en paquets logiciels par les maintainers de la distribution. Les paquets logiciels sont disponibles en ligne dans des dépôts, ou magasin d'application pour reprendre le terme popularisé par les GAFAM, qui sont des emplacements de stockage généralement répartis dans le monde. À côté des composants de base, tels que les installateurs de distribution (par exemple, Debian-Installer et Anaconda) ou les systèmes de gestion de packages, il n'y a que très peu de paquets qui sont écrits à partir de zéro par les maintaineurs d'une distribution Linux.

Comme illustré dans les figures [Fig. 4.2 et Fig. 4.3], l'écosystème Linux est extrêmement large, presque mille distributions Linux existent. En raison de la grande disponibilité de logiciels, les

distributions ont pris une grande variété de formes, notamment celles adaptées à l'utilisation sur les bureaux, les serveurs, les ordinateurs portables, les netbooks, les téléphones mobiles et les tablettes, ainsi que des environnements minimalistes généralement utilisés dans les systèmes embarqués. Il existe des distributions soutenues commercialement, telles que Red Hat Enterprise Linux, Fedora, CentOS (soutenu par Red Hat), openSUSE (soutenu par SUSE) et Ubuntu (soutenu par Canonical Ltd.), et des distributions entièrement impulsées par la communauté, telles que Debian, Slackware, Gentoo ou Arch Linux.
La plupart des distributions sont prêtes à l'emploi et précompilées pour un ensemble d'instructions spécifique, tandis que certaines distributions (comme Gentoo) sont distribuées principalement sous forme de code source et compilées localement pendant l'installation.

Dans les exemples suivants, des captures d'écran seront prises à partir de:

Ubuntu 22.04.2 LTS et 24.04.3 LTS, exécutant le bureau GNOME.

Pour plus de détails, voir le chapitre [Chap. 5].

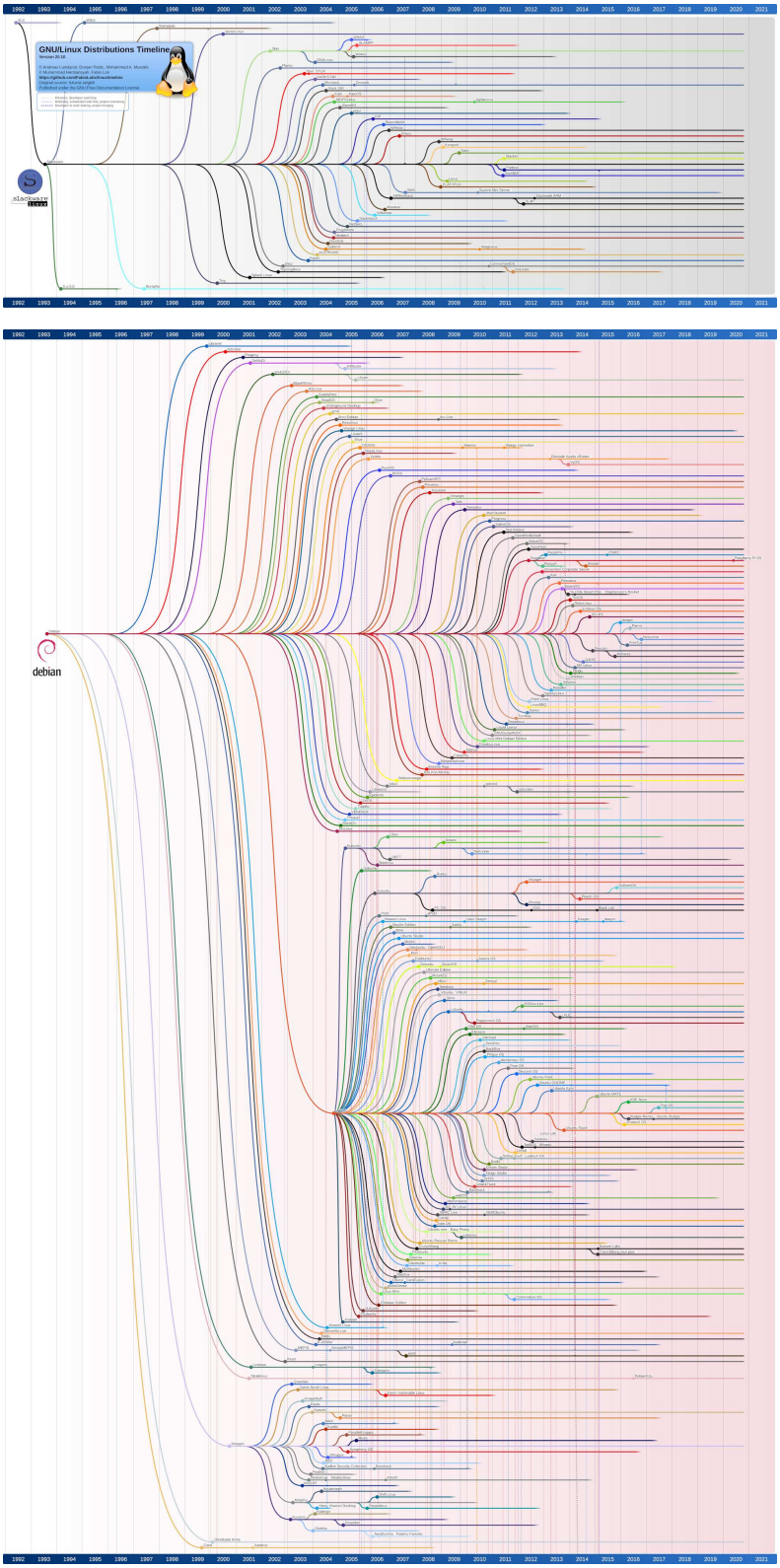


Figure 4.2: L'arbre des distributions Linux, en haut: l'arbre Slackware, en bas: l'arbre Debian.

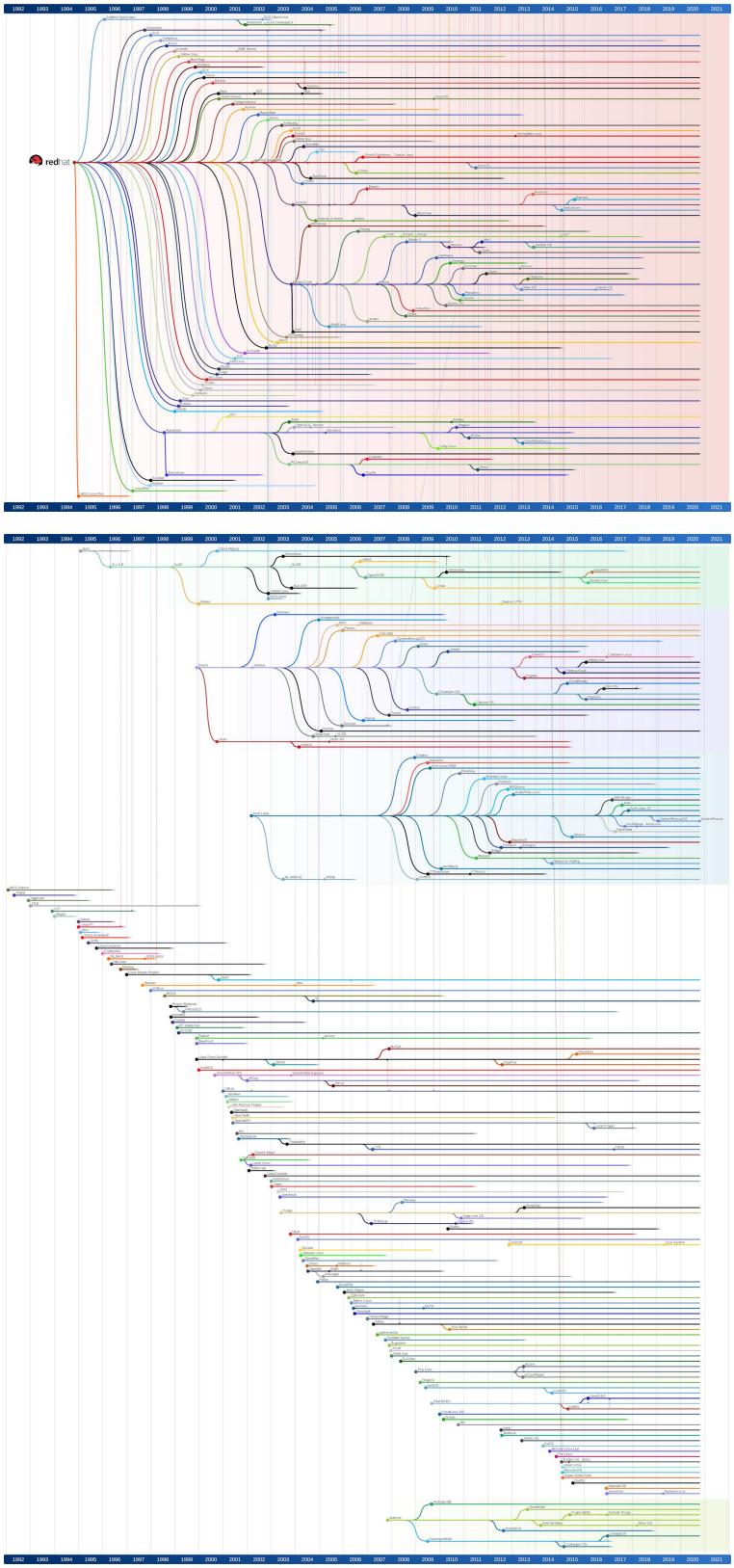


Figure 4.3: L'arbre des distributions Linux, en haut: l'arbre Red Hat, en bas: l'arbre des autres distributions.

4.2 Fondamentaux de Linux

4.2.1 Le terminal

Cela pourrait sembler surprenant de le mettre en premier, mais en considérant que la plupart des exemples et des commandes qui suivent proviennent de l'utilisation du terminal, il semble approprié de l'introduire au début. De plus, l'objectif de ce tutoriel est de présenter les bases nécessaires pour comprendre comment utiliser cet outil puissant.

La ligne de commande Linux est une interface texte à votre ordinateur. Souvent appelée shell, terminal, console, invite ou divers autres noms, elle peut donner l'impression d'être complexe et confuse à utiliser.

Pourtant un jour, peut-être plus tôt que prévu, vous rechercherez sur Internet comment obtenir de l'aide pour résoudre vos problèmes Linux, alors vous serez probablement confronté à cette vérité toute simple: les réponses sont, et seront toujours, fournies sous forme de commandes que vous devez copier et coller du site Web au terminal. Bien sûr vous trouverez peut-être une aide plus traditionnelle, en termes non Linux, en utilisant la souris et/ou les menus, mais cela reste moins probable. Associé au pouvoir et à la flexibilité que la ligne de commande offre, savoir l'utiliser peut être essentiel lorsqu'il s'agit de suivre des instructions en ligne, et encore plus si vous voulez obtenir le meilleur de votre système Linux.

Après avoir ouvert le terminal (voir section [Sec. 5.2.2.3]), vous devriez aboutir à une fenêtre assez terne avec un peu de texte étrange en haut, comme la figure [Fig. 4.4].

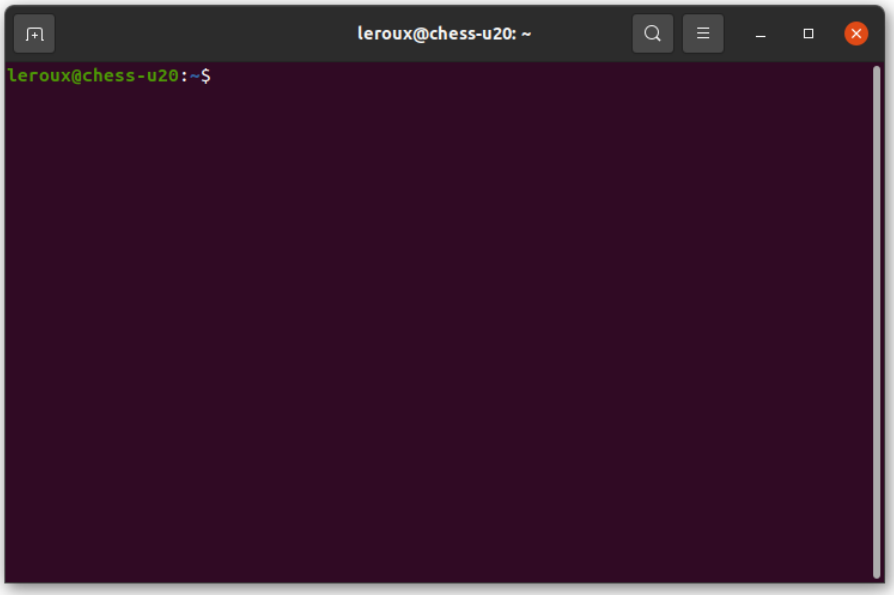


Figure 4.4: Le terminal GNOME dans Ubuntu 20.04.3 LTS

Selon votre système Linux, les couleurs peuvent ne pas être les mêmes, et le texte dira probablement quelque chose de différent, mais la disposition générale d'une fenêtre avec une grande zone de texte (la plupart du temps vide) devrait être similaire.
Lorsque vous tapez une commande, elle apparaît sur la même ligne que le texte étrange:

L'invite, peut ressembler à (Ubuntu Linux):

```
user@localhost:~$
```

ou (Fedora Linux):

```
user@localhost ~]$
```

user	=	le nom de l'utilisateur qui a ouvert le terminal
localhost	=	le nom de l'ordinateur
~	=	l'emplacement actif dans l'arborescence du système de fichiers (voir [Sec. 4.2.2.1]).

Figure 4.5: L'invite, ou, interpréteur de commande.

Ce texte [Fig. 4.5] signifie que l'ordinateur est prêt à accepter une commande, c'est la façon dont l'ordinateur vous invite à l'utiliser. En fait, il est souvent appelé invite, et vous pouvez voir des instructions qui disent "afficher une invite", "ouvrir une invite de commande", "à l'invite bash", ou des noms similaires. Ce sont juste des façons différentes de vous demander d'ouvrir un terminal.
Lorsque vous exécutez une commande, toute sortie qu'elle produit sera généralement affichée directement dans le terminal, puis vous verrez une autre invite une fois que l'action réalisée par la commande sera terminée. Certaines commandes peuvent produire beaucoup de texte, d'autres peuvent fonctionner en silence et ne produire aucune sortie du tout. Ne vous alarmez pas si vous exécutez une commande et qu'une autre invite apparaît immédiatement, car cela signifie généralement que la commande a réussi.

Sensibilité à la casse: soyez très prudent lorsque vous utilisez la ligne de commande.
Taper **`LS`** au lieu de **`ls`** produira une erreur, mais parfois la mauvaise casse peut entraîner une commande qui semble fonctionner, mais ne fait pas ce que vous attendiez.

Le chapitre [Chap. 6] présente les bases de l'utilisation de la ligne de commande, et vous pouvez en savoir plus dans The Linux Command Line de William Shotts.

vous trouverez un glossaire des commandes de base dans le chapitre [Chap. 8].
Et mon "Tutoriel de base pour la programmation BASH" illustre de façon beaucoup approfondie plus ce qui peut être fait avec le terminal en utilisant la programmation.

4.2.2 Norme de hiérarchie des fichiers

Adapté de la page web Wikipedia

La norme de hiérarchie des fichiers (Filesystem Hierarchy Standards FHS en Anglais) définit la structure de répertoire et le contenu des répertoires dans les distributions Linux. Elle est maintenue par la Linux Foundation. La dernière version est la 3.0, publiée le 3 juin 2015.

4.2.2.1 Structure de répertoire

Dans la FHS, tous les fichiers et répertoires apparaissent sous le répertoire racine `/`, même s'ils sont stockés sur des périphériques physiques ou virtuels différents. Le tableau [Tab. 4.1] présente une brève description de la structure de répertoire Linux, la plupart de ces répertoires existent dans tous les systèmes d'exploitation Unix-like et sont généralement utilisés de la même manière ; cependant, les descriptions ici sont celles utilisées spécifiquement pour la FHS et ne sont pas considérées comme obligatoires pour les plates-formes autres que Linux.

Répertoire	Description
/	Racine principale de la hiérarchie et répertoire racine de l'ensemble de la hiérarchie des fichiers.
/bin	Binaires de commande essentiels qui doivent être disponibles pour chaque utilisateur, notamment pour démarrer le système ou le réparer, ex: cat, ls, cp.
/boot	Fichiers de démarrage (par exemple, noyaux, initrd).
/dev	Fichiers de périphériques (par exemple, /dev/null, /dev/disk0, /dev/sda1, /dev/tty, /dev/random).
/etc	Fichiers de configuration système spécifiques à l'hôte "Editable Text Configuration" ou "Extended Tool Chest".
/home	Répertoires personnels des utilisateurs, contenant des fichiers sauvegardés, des paramètres personnels, etc.
/lib	Bibliothèques essentielles pour les binaires dans /bin et /sbin.
/lib32	Bibliothèques essentielles pour les binaires dans /bin et /sbin.
/lib64	Bibliothèques essentielles pour les binaires dans /bin et /sbin.
/media	Points de montage pour les médias amovibles tels que les clés USB ou les CD-ROM.
/mnt	Systèmes de fichiers montés temporairement.
/opt	Paquets de logiciels d'application ajoutés.
/proc	Système de fichiers virtuel d'information sur les processus et le noyau sous forme de fichiers. Généralement, généré automatiquement par le système.
/root	Répertoire personnel de l'utilisateur root.
/run	Données de variable de fonctionnement: informations sur le système depuis le dernier démarrage, ex: utilisateur(s) connectés, démons en cours d'exécution.
/sbin	Binaires de commande système essentiels (par exemple, fsck, init, route).
/srv	Données spécifiques générées par le système, ex: données et scripts pour les serveurs Web, données des serveurs FTP et référentiels de gestion de versions.
/sys	Contient des informations sur les périphériques, les pilotes et certaines fonctionnalités du noyau.
/tmp	Fichiers temporaires à conserver entre les redémarrages du système.
/usr	Hiérarchie secondaire pour les données utilisateur en lecture seule ; contient la majorité des outils et logiciels. Devrait être partageable et en lecture seule.
/usr/bin	Binaires de commande non essentiels (vos applications quotidiennes) ; pour tous les utilisateurs.
/usr/include	Fichiers d'en-tête standard (à des fins de programmation).
/usr/lib	Bibliothèques pour les binaires dans /usr/bin et /usr/sbin.
/usr/lib32	Bibliothèques au format alternatif
/usr/lib64	Bibliothèques au format alternatif
/usr/local	Hiérarchie tertiaire pour les données locales, spécifiques à cet hôte. Généralement comporte des sous-répertoires supplémentaires (par exemple, bin, lib, share).
/usr/sbin	Binaires de commande système non essentiels (par exemple, démons pour divers services réseau).
/usr/share	Données partagées indépendantes de l'architecture.
/usr/src	Code source (par exemple, le code source du noyau avec ses fichiers d'en-tête).
/usr/X11R6	X Window System, Version 11, Release 6 (jusqu'à FHS-2.3, facultatif).
/var	Fichiers variables: dont le contenu est censé être actualisé durant le fonctionnement normal du système, ex: journaux, courriers électroniques temporaires.
/var/cache	Données de cache d'application.
/var/lib	Informations d'état. Données persistantes modifiées par les programmes lorsqu'ils s'exécutent (par exemple, métadonnées de gestion de packages, etc.).
/var/lock	Fichiers de verrouillage. Fichiers qui suivent les ressources actuellement en cours d'utilisation.
/var/log	Fichiers de journal. Divers journaux.
/var/mail	Fichiers de boîte aux lettres. Dans certaines distributions, ces fichiers peuvent se trouver dans le répertoire déprécié /var/spool/mail.
/var/opt	Données variables provenant de paquets d'applications ajoutés qui sont stockés dans /opt.
/var/run	Données de variable de fonctionnement. Ce répertoire contient des informations sur le système depuis le dernier démarrage.
/var/spool	Spool pour les tâches en attente de traitement (par exemple, les files d'impression et les files de courrier sortant).
/var/spool/mail	Emplacement déprécié pour les boîtes aux lettres des utilisateurs.
/var/tmp	Fichiers temporaires à conserver entre les redémarrages du système.

/obligatoire, /facultatif

Table 4.1: La structure de répertoire Linux

4.2.2.2 Écriture de chemins de fichiers

Sur un système Linux, le chemin d'un fichier ou d'un répertoire est écrit à partir du répertoire racine **/**, les répertoires sont séparés en utilisant le symbole **/**.
En utilisant cette notation, le chemin pour atteindre le répertoire personnel d'un utilisateur peut être écrit:

- En partant du répertoire racine **/**:

```
/home/user
```

- Le chemin particulier pour atteindre le répertoire nommé `Bureau` et situé dans ce répertoire personnel, peut être écrit en utilisant les notations suivantes:
  - En partant du répertoire racine **/**:

```
/home/user/Bureau
```

  - En partant du répertoire personnel **~**:

```
~/Bureau
```

De la même façon pour écrire le chemin vers le fichier `MonFichier` situé sur le `Bureau` de l'utilisateur `user`:

- En partant du répertoire racine **/**:

```
/home/user/Bureau/MonFichier
```

- En partant du répertoire personnel **~**:

```
~/Bureau/MonFichier
```

4.2.3 Gestion des utilisateurs

Le système Linux est conçu comme un système multi-utilisateur, il est possible de distinguer trois types d'utilisateurs sur un système Linux:

- Le "super"-utilisateur ou administrateur (souvent appelé **root**) qui peut tout faire, notamment administrer et ainsi modifier la configuration du système.
- Les "**sudo**ers" pour "**super user do**"-utilisateurs qui peuvent utiliser le commande "**sudo**" pour demander des privilèges d'administrateur. Lorsqu'ils utilisent la commande "**sudo**", les **suoders** doivent confirmer leur identité en saisissant leur mot de passe utilisateur.
- L'utilisateur "standard" qui utilise l'ordinateur mais avec un accès restreint.

Selon la distribution Linux, le super utilisateur peut:

- Avoir un compte sur l'ordinateur et donc accéder à son répertoire personnel (`/root`). Dans ce cas, pour administrer l'ordinateur, le super utilisateur doit se connecter en utilisant la commande "**su -**".
- Ne pas avoir de compte, alors seuls les utilisateurs qui ont reçu l'autorisation peuvent administrer l'ordinateur en utilisant la commande "**sudo**".

Notez qu'il est courant de trouver des groupes d'utilisateurs sur un système Linux, c'est-à-dire des utilisateurs qui partagent les mêmes privilèges et qui ont été regroupés dans un groupe.

4.2.4 Permissions de fichiers

L'objectif de cette section est d'introduire les idées de base sur le système de permissions de fichiers sur un système Linux/Unix. Sur un système Linux, gérer les permissions de fichiers n'est pas nécessairement synonyme d'administrer le système en installant le pilote pour le nouveau matériel ou en mettant à jour la bibliothèque de programmes de l'ordinateur. Comprendre le système de permissions de fichiers du système Linux est la condition préalable la plus basique pour devenir un utilisateur "Avancé", c'est-à-dire un utilisateur ayant le pouvoir de comprendre ce qu'il fait avec l'ordinateur.

La règle d'or d'un système Linux/Unix est que tout est un fichier. L'ordinateur est vu par le système d'exploitation comme un arbre de fichiers et donc chaque composant (écran, clavier, souris, carte graphique ...) est vu comme un fichier. Lorsqu'un composant matériel est ajouté à l'ordinateur, un fichier est ajouté dans l'arboresecnce du système d'exploitation **/**. Par conséquent, les permissions d'accès, d'utilisation ou de modification d'un fichier sont gérées comme celles d'un fichier sur le disque dur.
Dans un arbre de fichiers, on peut distinguer deux objets: les "fichiers simples" et les "répertoires". Pour les deux, les permissions sont gérées de la même manière:

- Les différentes permissions qui peuvent être accordées à un fichier sont:
  - lire: pour visualiser son contenu
  - écrire: pour modifier son contenu (par exemple, édition)
  - exécuter: pour exécuter son contenu (par exemple, programme)
- Les différentes permissions qui peuvent être accordées à un répertoire sont:
  - lire: pour visualiser son contenu
  - écrire: pour modifier son contenu (par exemple, ajouter de nouveaux fichiers)
  - exécuter: pour entrer dans ce répertoire (par exemple, changer de répertoire)

Pour obtenir les informations concernant les permissions d'un fichier dans l'arborescence Linux, on peut utiliser la commande **ls** (voir section [Sec. 8.1]):

```
user@localhost:~/Bureau$ ls -lh MonFichier
-rwxrw-r--. 1 user ipcms 1.0K  15 févr.  2011 MonFichier
```

La syntaxe de cette ligne est la suivante:

- "-" signifie "fichier", tandis que "d" signifie "répertoire" et "l" signifie "lien symbolique".
- "rwxrw-r-" sont les permissions sur le fichier.
- "1" est le nombre de liens physiques du fichier avec le disque dur.
- "user" est le nom de l'utilisateur propriétaire du fichier.
- "ipcms" est le nom du groupe auquel appartient l'utilisateur propriétaire du fichier.
- "1.0K" est la taille du fichier sur le disque dur.
- "15 févr. 2011" est la date de dernière modification du fichier.
- "MonFichier" est le nom du fichier

Les permissions sur le fichier peuvent être décomposées en 3 séries de 3 lettres: r (pour lire), w (pour écrire) et x (pour exécuter), et le symbole - à la place d'une lettre signifie que la permission est refusée. La première série de 3 lettres fait référence au propriétaire du fichier, la deuxième au groupe auquel appartient le propriétaire du fichier, et la troisième à tous les autres utilisateurs de l'ordinateur. Dans le cas du fichier `MonFichier`, le propriétaire du fichier, `user`, a toutes les permissions (lire, écrire, exécuter). Les membres du groupe auquel appartient le propriétaire du fichier (c'est-à-dire dans cet exemple les membres du groupe `ipcms`) peuvent lire et modifier le fichier. Enfin, les autres utilisateurs ne peuvent que lire le fichier `MonFichier`.

4.2.4.1 Ajustement des permissions de fichiers: en utilisant le terminal

Il est possible de définir les permissions d'accès, d'utilisation ou de modification d'un fichier en utilisant la commande **chmod** (voir section [Sec. 8.1]) et en utilisant 3 chiffres:

(**0**	=	aucune permission du tout	=	`---`)
1	=	exécuter	=	`--x`
2	=	écrire	=	`-w-`
4	=	lire	=	`r-`

Ainsi que leurs combinaisons:

3	=	**1 + 2**	=	exécuter + lire	=	`r-x`
5	=	**1 + 4**	=	exécuter + écrire	=	`rw-`
6	=	**2 + 4**	=	écrire + lire	=	`rw-`
7	=	**1 + 2 + 4**	=	exécuter + écrire + lire	=	`rwx`

Linux distingue 3 classes d'utilisateurs pour lesquels il est possible d'accorder des permissions sur un fichier:

- Le propriétaire du fichier.
- Le groupe auquel appartient le propriétaire du fichier.
- Tous les autres utilisateurs reconnus par le système.

Les permissions sur le fichier étant accordées pour chaque catégorie d'utilisateur par une seule combinaison des nombres 1, 2 et 4.

Par exemple, il est possible d'accorder les permissions 644 au fichier `MonFichier`:

```
user@localhost:~/Bureau$ chmod 644 MonFichier
```

La commande accordera la permission de lire et de modifier (écrire) le fichier au propriétaire, et supprimera aussi la permission d'exécuter. Elle accordera la permission de lire le fichier aux membres du groupe auquel appartient le propriétaire, supprimera les permission de modifier et d'exécuter le fichier. Et elle fera de même pour les permissions accordées à tous les autres utilisateurs de l'ordinateur.

4.2.4.2 Ajustement des permissions de fichiers: en mode graphique

La procédure est illustrée dans la figure [Fig. 4.6]:

- En utilisant le clavier: appuyez sur Ctrl + i
- En utilisant la souris:
  1. Utilisez le menu contextuel de la souris sur le fichier ou le répertoire à vérifier.
  2. Cliquez sur "Propriétés".
  3. Sélectionnez l'onglet "Permissions" pour vérifier et ajuster les permissions de fichiers ou de répertoires.

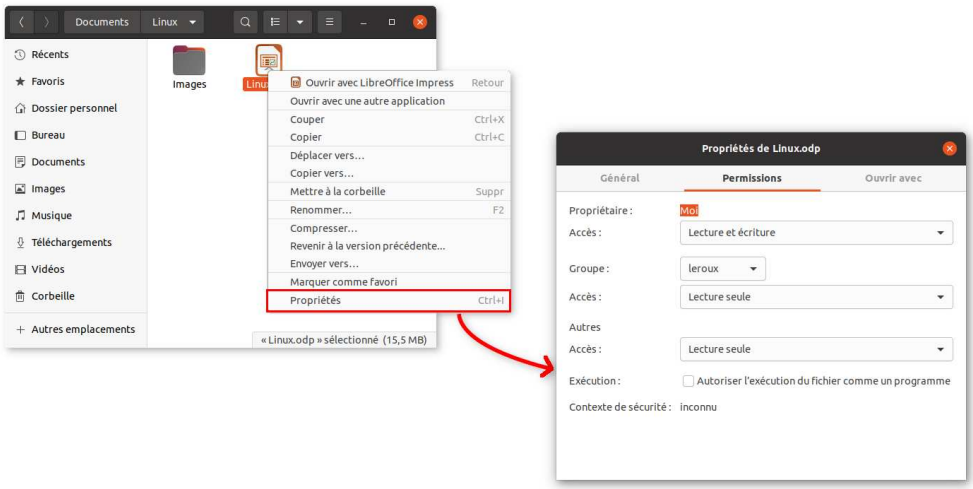


Figure 4.6: Ajustement des permissions de fichiers en mode graphique.

4.2.5 Variables d'environnement

Adapté de la page web Wikipedia

4.2.5.1 Introduction

Les variables d'environnement sont un ensemble de valeurs dynamiques nommées qui peuvent affecter la façon dont les processus s'exécutent sur un ordinateur. Elles peuvent être considérées comme créant l'environnement de travail dans lequel un processus s'exécute.
Dans un système Linux, chaque processus a son propre ensemble privé de variables d'environnement. Par défaut, lorsqu'un processus est créé, il hérite d'un environnement dupliqué de son processus parent, sauf pour les modifications explicites apportées par le parent lorsqu'il crée l'enfant. À partir de shells tels que bash, vous pouvez modifier les variables d'environnement pour une invocation de commande particulière en invoquant indirectement `env` ou en utilisant la notation de commande `VARIABLE_ENV=VALUE` (voir section [Sec. 8.3]).
Des exemples de variables d'environnement incluent:

- **`PATH`**: liste des répertoires dans lesquels le shell recherche les commandes que l'utilisateur peut taper sans avoir à fournir le chemin complet.
- **`HOME`**: emplacement du répertoire personnel d'un utilisateur dans le système de fichiers.
- **`PWD`**: indique le répertoire de travail actif.
- **`TERM`**: spécifie le type de terminal informatique ou d'émulateur de terminal utilisé.
- **`SHELL`**: spécifie le type d'interpréteur de commande utilisé (BASH, TCSH ...).
- **`PS1`**: spécifie comment l'invite est affichée dans le shell Bourne et ses variantes.

Les scripts de shell et les fichiers batch utilisent des variables d'environnement pour communiquer des données et des préférences aux processus enfants. Ils peuvent également être utilisés pour stocker des valeurs temporaires pour référence ultérieure dans le script.
Dans Unix/Linux, une variable d'environnement modifiée dans un script ou un programme compilé n'affectera que ce processus et éventuellement les processus enfants. Le processus parent et tout processus non lié ne seront pas affectés.
Les variables d'environnement sont généralement initialisées pendant le démarrage du système par les scripts d'initialisation du système, et héritées par tous les autres processus du système. Les utilisateurs peuvent modifier celles-ci dans le script de profil du shell qu'ils utilisent.
Les variables peuvent être utilisées à la fois dans les scripts et dans la ligne de commande. Elles sont référencées en plaçant des symboles spéciaux devant ou autour du nom de la variable. Par exemple, pour afficher le chemin de recherche de programme, l'utilisateur doit entrer:

```
user@localhost:~$ echo $PATH
```

La commande **`env`** (voir [Sec. 8.3]), liste toutes les variables d'environnement et leurs valeurs.

4.2.5.2 Principes de fonctionnement des variables d'environnement

Quelques principes simples régissent la façon dont les variables d'environnement fonctionnent.

- **Local au processus**
  Les variables d'environnement sont locales au processus dans lequel elles ont été définies. Cela signifie que si nous ouvrons deux fenêtres de terminal (deux processus différents exécutant un shell) et que nous modifions la valeur d'une variable d'environnement dans une fenêtre, cette modification ne sera pas visible par l'autre fenêtre.

- **Héritage**
  Lorsqu'un processus parent crée un processus enfant, le processus enfant hérite de toutes les variables d'environnement et de leurs valeurs que le processus parent possédait.

- **Sensibilité à la casse**
  Les noms des variables d'environnement sont sensibles à la casse.

- **Persistance**
  La persistance des variables d'environnement peut être de session ou de système.

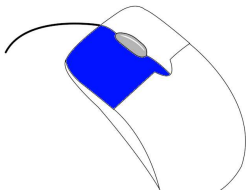
4.3 Trucs et astuces pour Linux

4.3.1 La copie/collage de la souris

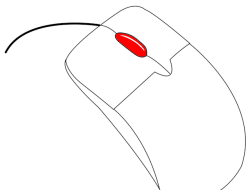
Il existe un outil de copie/collage très pratique disponible lors de l'utilisation de la souris sous Linux, et c'est une fonctionnalité standard pour tous les Linux:

1. Si vous lisez/éditez du texte, n'importe où, et dans n'importe quel logiciel:

2. Si vous sélectionnez ce texte en utilisant le bouton gauche de la souris:



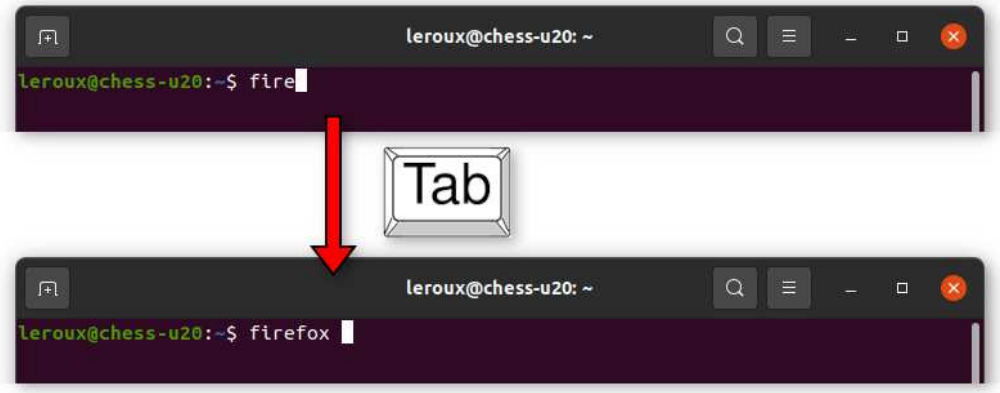
3. Ce texte est immédiatement copié dans un tampon de mémoire (dans la mémoire de votre ordinateur) et vous pouvez facilement le coller en appuyant sur le bouton du milieu de la souris (molette ou autre):



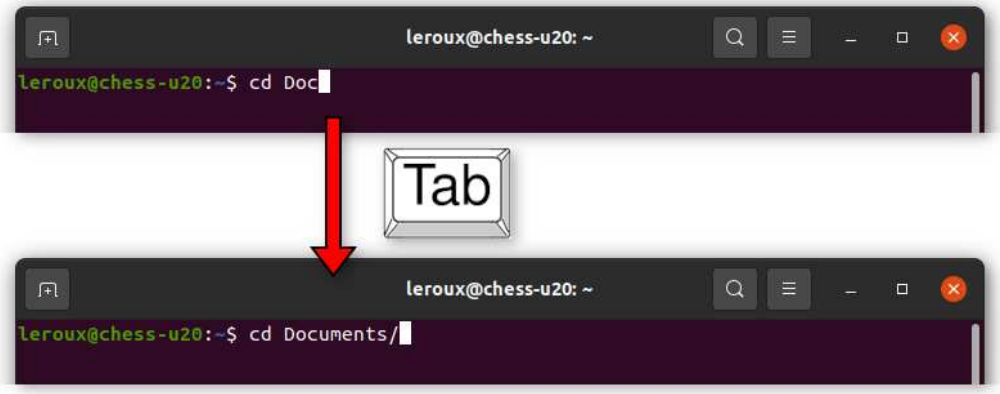
4.3.2 Auto-complétion

L'auto-complétion, ou complétion, est une astuce de terminal. Lorsque vous saisissez des commandes, si le nom de la commande est partiellement saisi (écrit dans le terminal), alors si vous appuyez sur la touche ⟷ / tab du clavier, Linux essayera de compléter la commande/ligne pour vous:

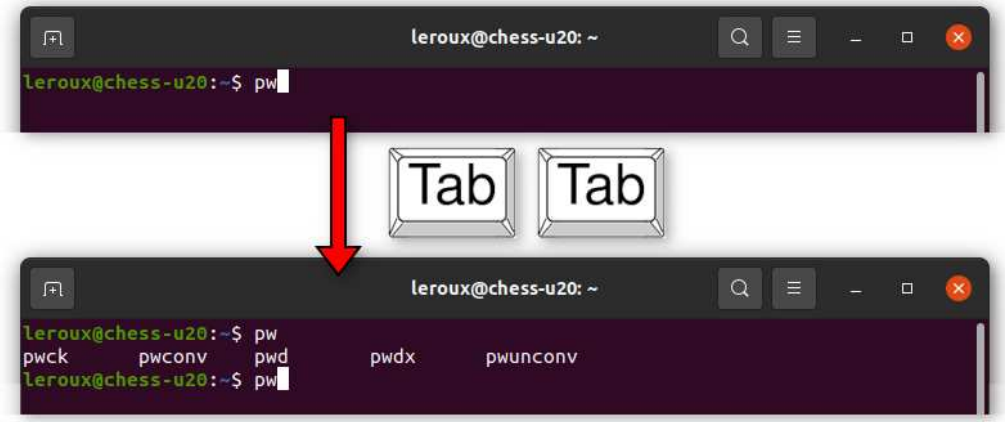
- Si il n'y a qu'une seule possibilité, le système complétera immédiatement la commande/ligne:
  - Pour une commande:



  - Pour un fichier ou un chemin:



- Si il y a plusieurs options, appuyez une deuxième fois sur la touche ⟷ / tab et le système affichera les options disponibles pour compléter la commande/ligne:
  - Pour une commande:



  - Pour un fichier ou un chemin:

4.3.3 Trucs et astuces pour la ligne de commande

Vous trouverez ci-dessous quelques astuces bien utiles pour utiliser la ligne de commande:

Se déplacer dans le répertoire personnel de l'utilisateur:	`cd`
Se déplacer dans le répertoire précédent dont vous venez:	`cd -`
Se déplacer dans le répertoire parent dans l'arborescence:	`cd ..`
Se déplacer au début de la ligne:	Ctrl + a
Se déplacer à la fin de la ligne:	Ctrl + e

Table 4.2: Trucs et astuces pour la ligne de commande.

Ubuntu

Figure 5.1: Les logos officiels de la distribution GNU/Linux Ubuntu.

Ubuntu est une distribution Linux basée sur Debian, composée principalement de logiciels libres et open source, et développée par Canonical.
Toutes les 6 mois, 4 éditions de Ubuntu sont publiées:

- Bureau
- Serveur
- Internet des Objets (IOT) et robots
- Cloud

Avec des versions de support à long terme (LTS, support pendant 5 ans) publiées tous les 2 ans. Chaque version de Ubuntu a un numéro de version avec l'année et le numéro du mois de la publication. Ainsi la première version, Ubuntu 4.10, a été publiée le 20 octobre 2004.
La version de support à long terme la plus récente est Ubuntu "Noble Numbat" 24.04 LTS, qui est prise en charge jusqu'en 2029 sous support public et jusqu'en 2034 en option gratuite pour les particuliers (payante pour les entreprises) qui s'inscrivent en ligne. La dernière version standard est Ubuntu 25.10 "Questing Quokka", qui est prise en charge pendant neuf mois.

Ubuntu est nommé d'après la langue Nguni Bantu, la philosophie de ubuntu, Canonical indique que cela signifie "humanité envers les autres" avec une connotation de "Je suis ce que je suis parce que nous sommes tous ce que nous sommes".

5.1 Téléchargement et installation de Ubuntu

Toutes les versions actives de Ubuntu peuvent être trouvées sur le site Web https://releases.ubuntu.com/:

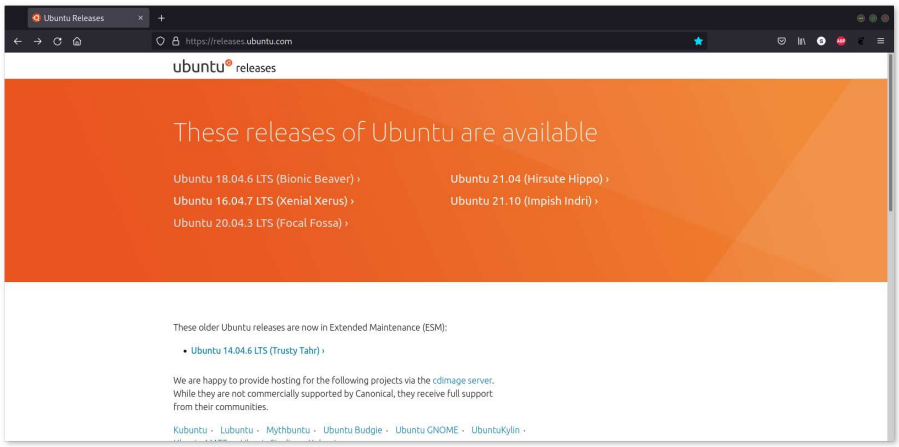


Figure 5.2: Site Web des versions de Ubuntu.

Pour commencer avec Linux, je vous recommande vivement d'essayer, et/ou d'installer, la version LTS la plus récente d'Ubuntu, au moment où je rédige ce manuel: Ubuntu "Noble Numbat" 24.04 LTS.
Télécharger l'image ISO, soit depuis le lien à partir du site Web de la version (voir figure [Fig. 5.2]), soit directement depuis le site Web dédié à la version LTS la plus récente:

https://ubuntu.com/download/desktop

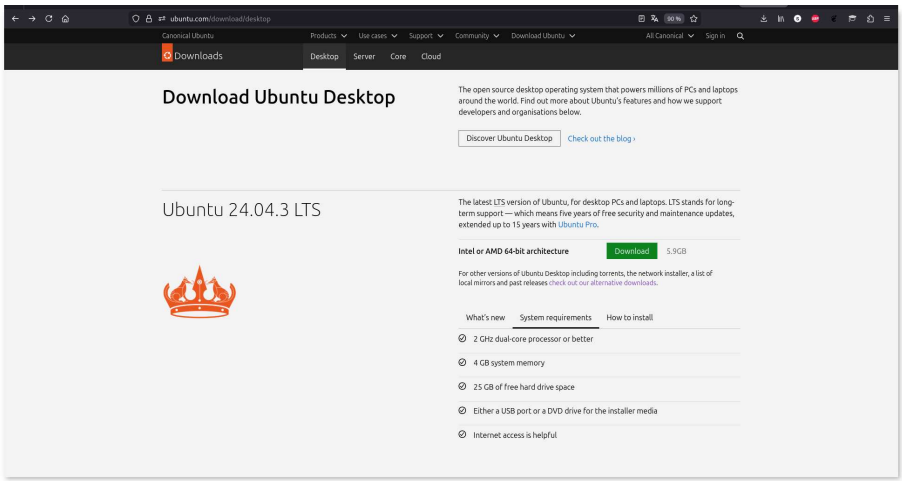


Figure 5.3: Site Web de la version LTS de bureau de Ubuntu la plus récente.

Le lien direct pour télécharger l'image ISO Ubuntu "Noble Numbat" 24.04 LTS

https://releases.ubuntu.com/noble/ubuntu-24.04.3-desktop-amd64.iso

5.1.1 Après le téléchargement

Après avoir téléchargé Ubuntu "Noble Numbat" 24.04 LTS, 2 choix s'offrent à vous:

1. Essayer simplement sans l'installer: la façon la plus simple de découvrir Ubuntu et Linux.
2. L'installer sur votre disque dur.

Dans les deux cas, vous devrez préparer une clé USB et vous devrez démarrer à partir de cette clé USB au démarrage, c'est-à-dire lorsque vous démarrez votre ordinateur, demandez-lui de démarrer à partir de la clé.

Pour préparer la clé USB, suivez simplement l'un des tutoriels suivant:

- Pour MS Windows: https://ubuntu.com/tutorials/create-a-usb-stick-on-windows
- Pour MacOSX: https://ubuntu.com/tutorials/create-a-usb-stick-on-macos

Il y a un assistant complet qui vous guidera pas à pas à travers les quelques étapes nécessaires:

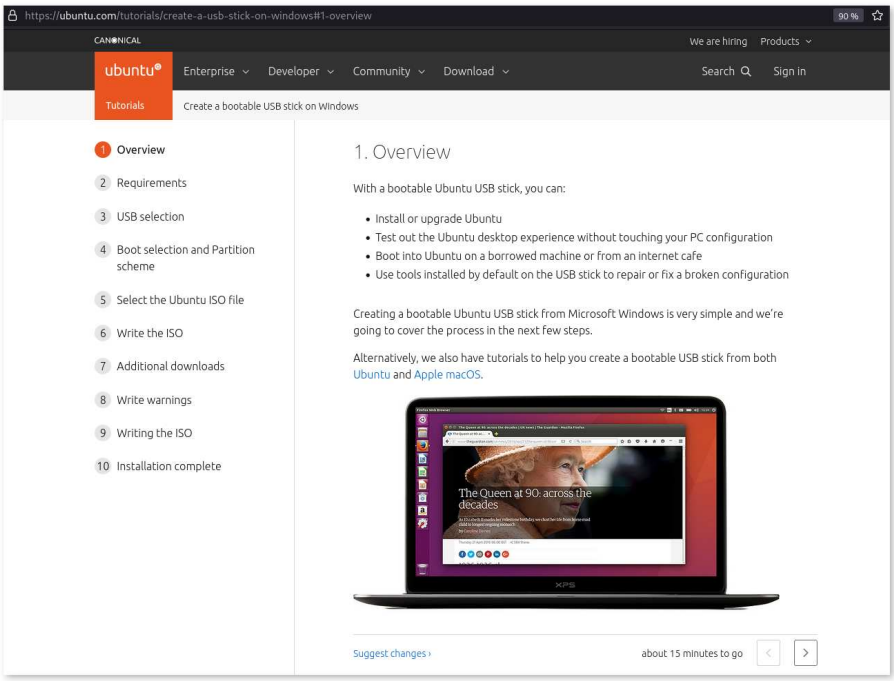


Figure 5.4: Préparation d'une clé USB pour tester ou installer Ubuntu.

Dans la suite, je considérerai que vous avez préparé la clé USB et que vous êtes prêt à l'utiliser.

5.1.1.1 Essayer Ubuntu

Pour cela, démarrez simplement votre ordinateur avec la clé USB branchée, et démarrez depuis la clé USB. Si votre ordinateur ne démarre pas automatiquement depuis la clé USB, essayez de maintenir F12 lorsque votre ordinateur démarre. Avec la plupart des machines, cela vous permettra de sélectionner le périphérique USB à partir d'un menu de démarrage système spécifique.
Cependant, vous devrez peut-être entrer dans le menu BIOS pour modifier la séquence de démarrage, puis redémarrez en demandant à votre ordinateur de démarrer en démarrant sur la clé USB.
Lorsque cela est fait, sautez simplement à la section [Sec. 5.2] de ce manuel.

5.1.1.2 Installer Ubuntu

Dans ce manuel, je ne couvrirai pas le processus d'installation, simplement parce qu'il est maintenant vraiment simple d'installer Linux, et, parce que de nombreux tutoriels étape par étape existent déjà et sont certainement beaucoup mieux écris et pensés que ce que je pourrais proposer.
Pour installer Ubuntu sur votre disque dur, vous pouvez suivre le tutoriel à:

https://ubuntu.com/tutorials/install-ubuntu-desktop

Quelques recommandations personnelles:

1. Faites une copie de toutes vos données avant d'installer Linux
2. Le démarrage multiple, c'est-à-dire avoir plusieurs systèmes d'exploitation sur le même ordinateur, est naturellement possible:

   https://www.tecmint.com/install-ubuntu-alongside-with-windows-dual-boot/

   N'oubliez pas ceci après l'installation:

   - Linux pourra voir, monter, utiliser la partition MS Windows.
   - MS Windows ne pourra pas voir, monter, ou utiliser la partition Linux.

Dans l'ensemble, si vous avez besoin d'aide, demandez-la !

5.2 Utiliser Ubuntu

Dans la section suivante, je vais utiliser des exemples de Ubuntu "Noble Numbat" 24.04 LTS et de Ubuntu "Jammy Jellyfish" 22.04 LTS. La première version de ce manuel était basée sur Ubuntu "Focal Fossa" 20.04 LTS et seuls quelques détails visuels ont changé par rapport à Ubuntu "Noble Numbat" 24.04 LTS, les exemples sont parfaitement valables. J'indiquerais les différences si nécessaire.

5.2.1 Les premiers pas

Immédiatement après l'installation, ou après avoir démarré l'ordinateur sur la clé USB pour essayer Ubuntu, vous serez confronté à l'écran suivant:

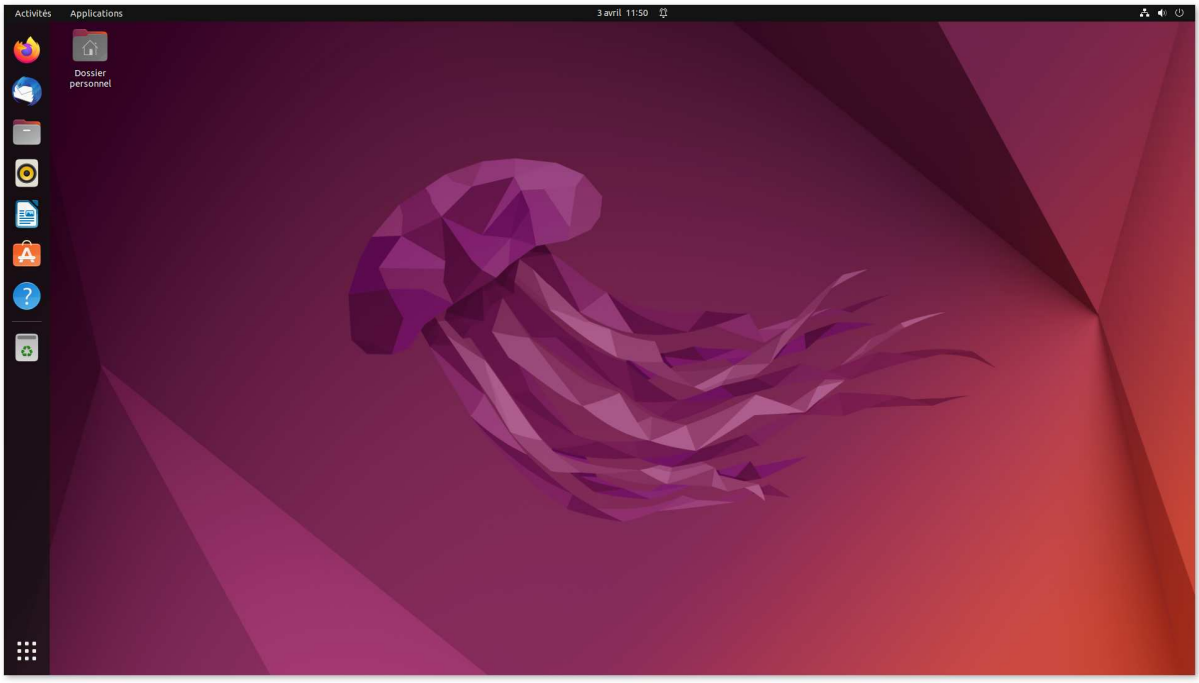


Figure 5.5: Ubuntu "Jammy Jellyfish" 22.04 LTS, avec le bureau GNOME.

Ceci est le bureau GNOME par défaut configuré pour Ubuntu.
Si c'est votre premier pas avec Linux, cela vous permet de découvrir à quoi ressemble Ubuntu, mais attention il existe beaucoup d'aspects visuels différents et auxquels vous êtes succeptible d'être confronté. Pas seulement parce que GNOME peut être personnalisé, mais aussi parce qu'il existe de nombreux gestionnaires de bureau différents dans le monde Linux / des logiciels libres.
Par exemple, Linux peut également ressembler aux illustrations de la figure [Fig. 5.6].

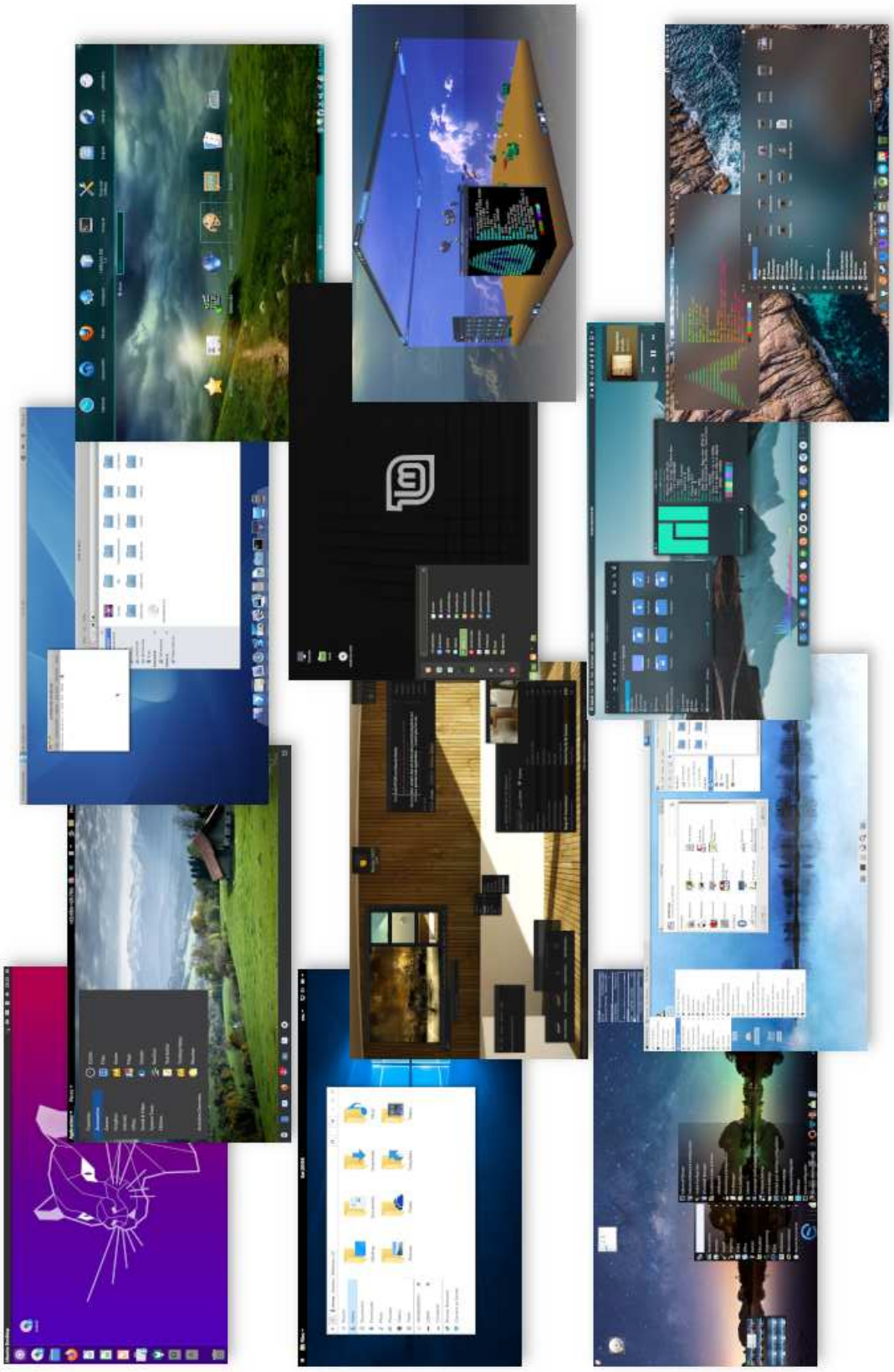


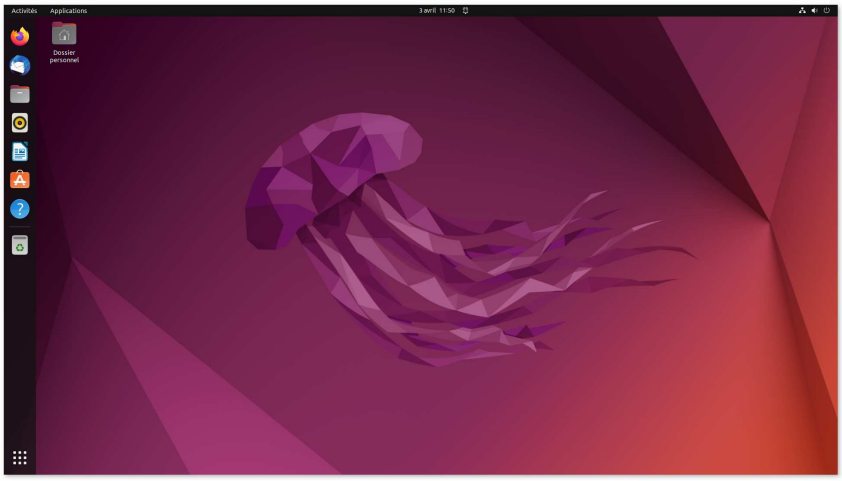
Figure 5.6: Quelques exemples de gestionnaires de bureau Linux.

Le tableau [Tab. 5.1] présente une liste de quelques-uns des gestionnaires de bureau parmi les plus populaires dans le monde Linux, mais cette liste n'est pas exhaustive, rappelez-vous que vous êtes maintenant dans le monde des logiciels libres et que vous pourriez tomber sur quelque chose de différent.

GNOME	https://www.gnome.org/	
KDE	https://kde.org/	
Cinnamon	https://projects.linuxmint.com/cinnamon/	
Xfce	https://www.xfce.org/	
MATE	https://mate-desktop.org/	
LXQt	ttps://lxqt-project.org/	
Enlightenment	https://www.enlightenment.org/	
Deepin	https://www.deeping.org/	
Pantheon	https://elementary.io/	

Table 5.1: Quelques-uns des gestionnaires de bureau Linux les plus populaires.

Dans Ubuntu, le gestionnaire de bureau par défaut s'appelle GNOME, il est vraiment intuitif et facile à utiliser, et il a également été personnalisé par l'équipe Ubuntu afin que tout soit plus facile pour les débutants. Gardez à l'esprit que la plupart des exemples qui suivent sont adaptés à la combinaison Ubuntu "Noble Numbat" 24.04 LTS + bureau GNOME" et que quelques aspects peuvent changer pour une autre distribution Linux, et encore plus pour un autre gestionnaire de bureau.


Activités
Applications
3 avril 11:50
Dossier personnel


GNOME™

5.2.2 Le bureau GNOME dans Ubuntu

Immédiatement après l'installation, ou après avoir démarré l'ordinateur sur la clé USB pour essayer Ubuntu, vous serez confronté à l'écran de la figure [Fig. 5.5].
Sur cet écran, vous remarquerez un raccourci vers votre répertoire personnel (`/home/user`) sur le bureau, ainsi qu'un raccourci vers la corbeille. Vous remarquerez également une barre, ou dock, sur le côté gauche de l'écran qui présente des raccourcis vers vos programmes préférés.
Sur le bureau Ubuntu/GNOME, vous trouverez 2 zones d'intérêt principales:

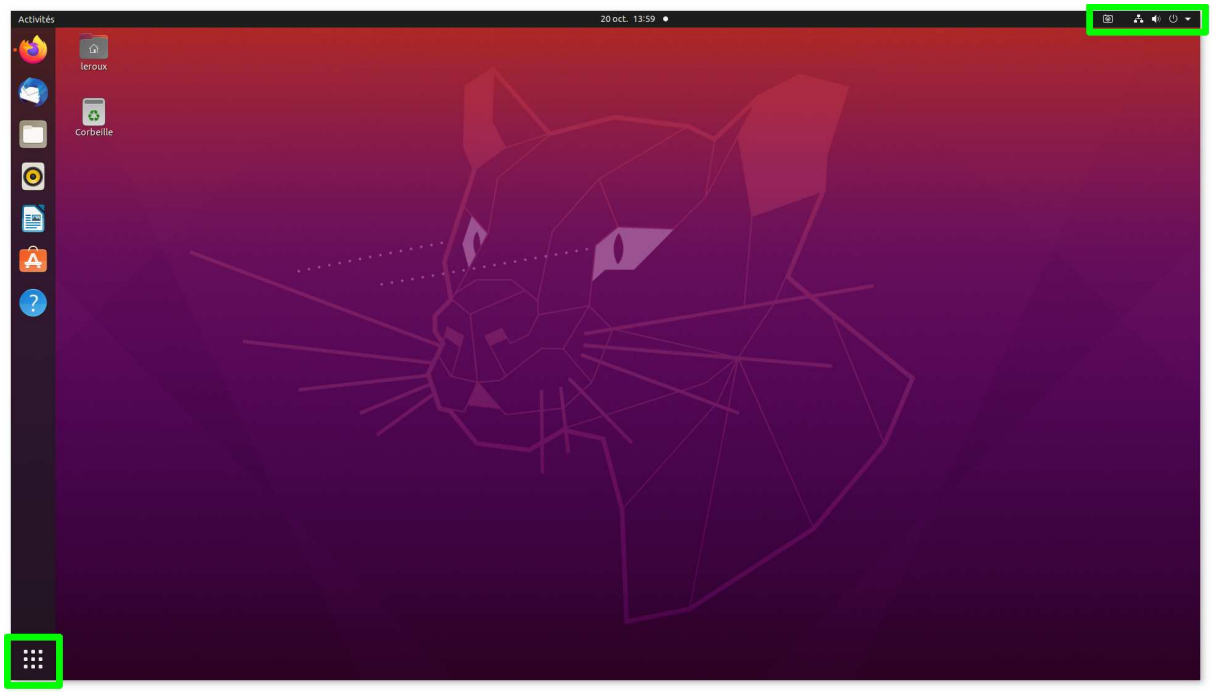


Figure 5.7: Premiers pas avec le bureau GNOME dans Ubuntu "Focal Fossa" 20.04 LTS, seule l'image de fond est différente dans Ubuntu "Noble Numbat" 24.04 LTS.

- Le premier dans le coin supérieur droit de l'écran, permet d'ouvrir le menu de paramètres/arrêt. À partir de là, vous pouvez accéder au panneau de contrôle d'Ubuntu et éteindre votre ordinateur.
- Le second dans le coin inférieur gauche de l'écran, permet d'ouvrir le menu d'applications. À partir de là, vous pouvez parcourir les programmes disponibles sur votre ordinateur.

5.2.2.1 Le panneau de contrôle

Comme illustré sur la figure [Fig. 5.8], si vous sélectionnez "Paramètres" sur le menu disponible dans le coin supérieur droit de l'écran, ou alternativement dans le menu disponible après un clic droit sur le bureau, vous pouvez accéder au panneau de contrôle d'Ubuntu

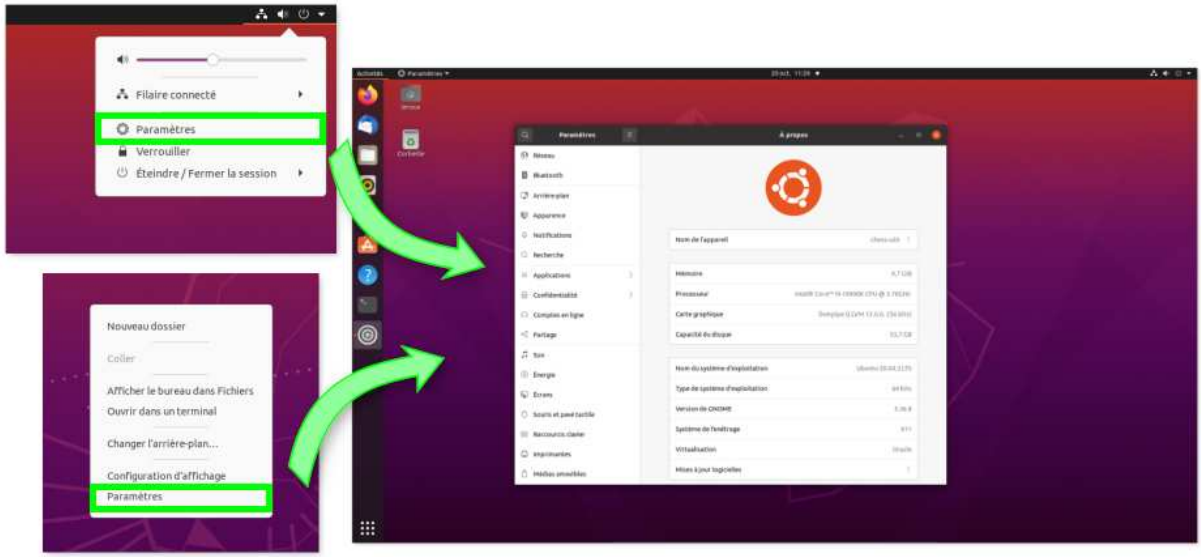


Figure 5.8: Ouvrir le panneau de contrôle dans Ubuntu.

5.2.2.2 Le menu d'applications

Comme illustré dans la figure [Fig. 5.9], si vous cliquez dans le coin inférieur gauche du bureau GNOME, alors vous ouvrirez le menu d'applications:

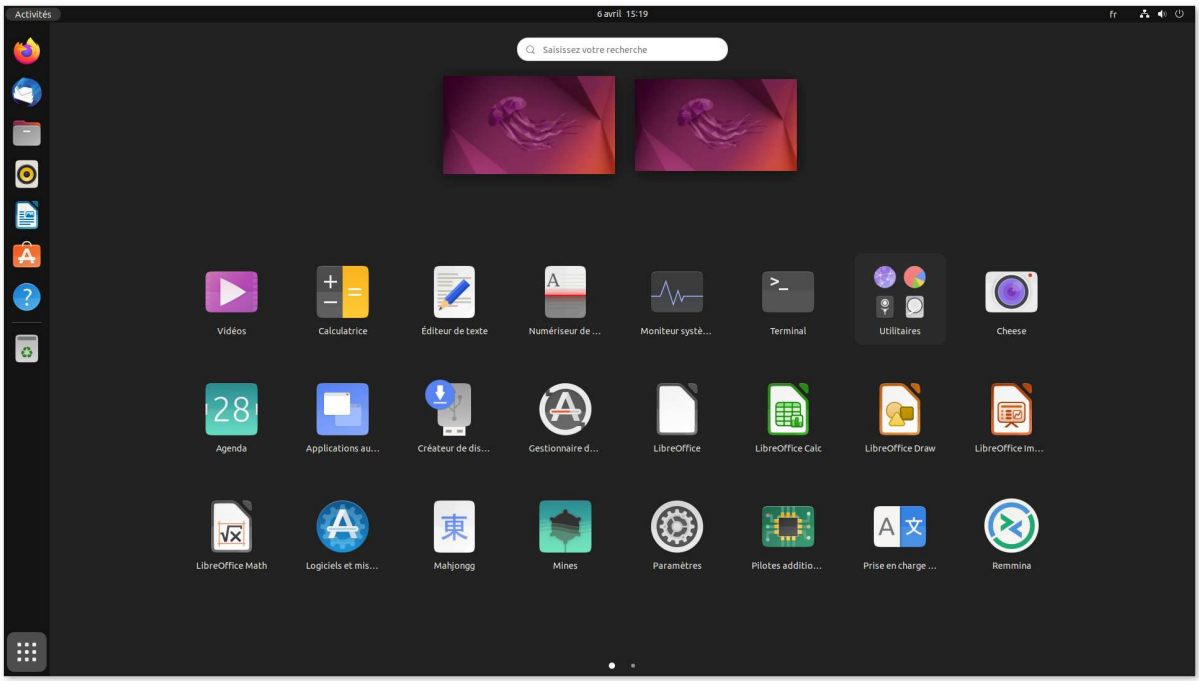


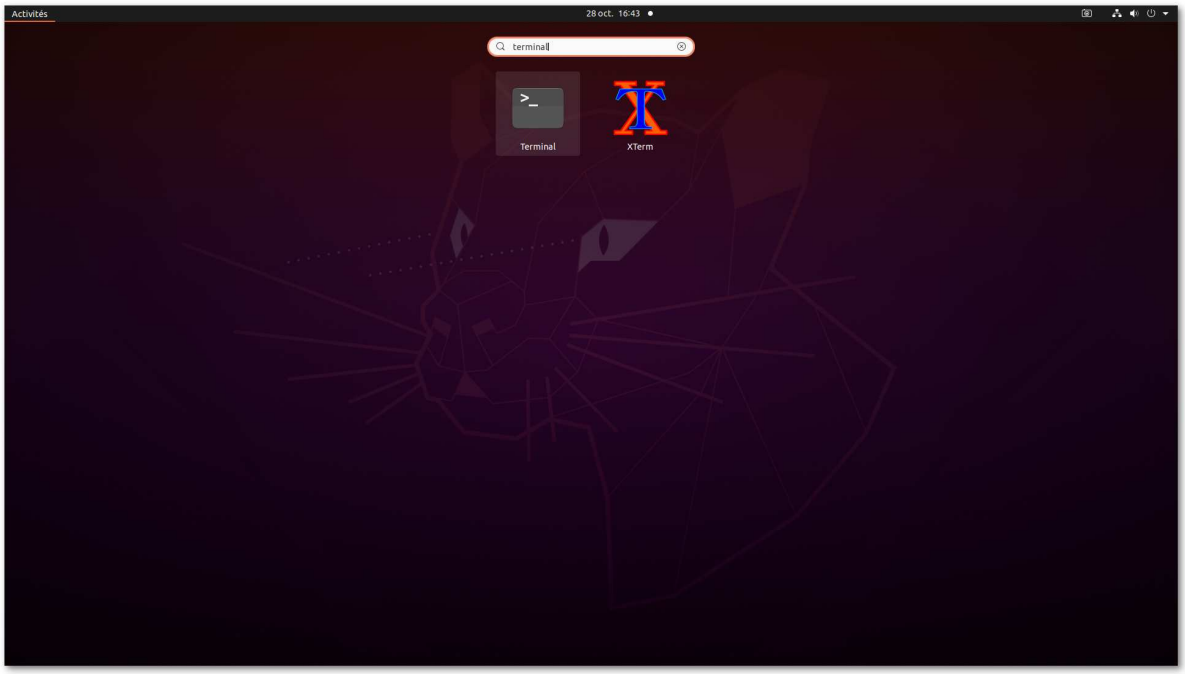
Figure 5.9: Ouvrir le menu d'applications dans Ubuntu "Jammy Jellyfish" 22.04 LTS.

À partir de là, vous pouvez facilement parcourir les programmes immédiatement disponibles, si ce que vous cherchez n'apparaît pas immédiatement, vous pouvez le rechercher en utilisant la zone de recherche.

5.2.2.3 Terminal

Pour ouvrir un terminal, vous pouvez utiliser:

- La souris: ouvrir le panneau d'applications GNOME et rechercher "Terminal":



- Le raccourci clavier (fonctionne pour la plupart des Linux), appuyez: Ctrl + Alt + t

Jusqu'à présent, j'ai couvert la plupart des points de base pour vous préparer à essayer et à tester Linux et Ubuntu, donc si vous n'êtes que simplement en train d'essayer Ubuntu, alors vous êtes débuter, en bref découvrir le monde Linux.

Sinon, si vous êtes dans une configuration post-installation de votre système Ubuntu "Noble Numbat" 24.04 LTS tout neuf, alors lisez les prochaines pages. Je vous donnerai quelques conseils et astuces pour vous aider à démarrer de la manière la plus simple et la plus sûre avec Ubuntu.

5.2.3 Après l'installation

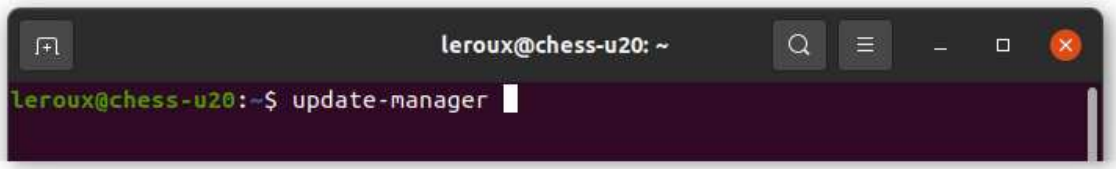
Voici plusieurs choses que je vous recommande de faire immédiatement après avoir installé Ubuntu:

- Vérifiez les mises à jour: voir section [Sec. 5.2.3.1]
- Activez les dépôts des partenaires: voir section [Sec. 5.2.3.2]
- Installez les pilotes de matériel éventuellement manquants
- Installez des codecs multimédia supplémentaires

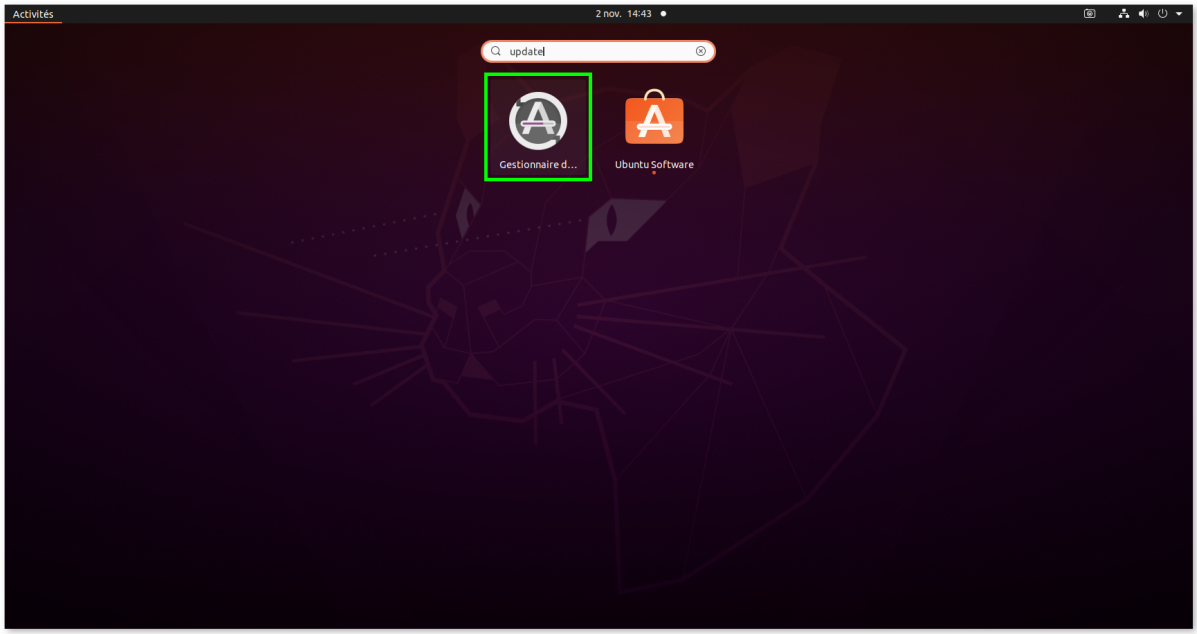
5.2.3.1 Vérifiez les mises à jour

Pour vérifier les mises à jour, en supposant que vous ne l'ayez pas fait pendant l'installation:

1. Ouvrez l'utilitaire de gestion des mises à jour:
    - En utilisant le terminal, entrez simplement la commande: `update-manager`

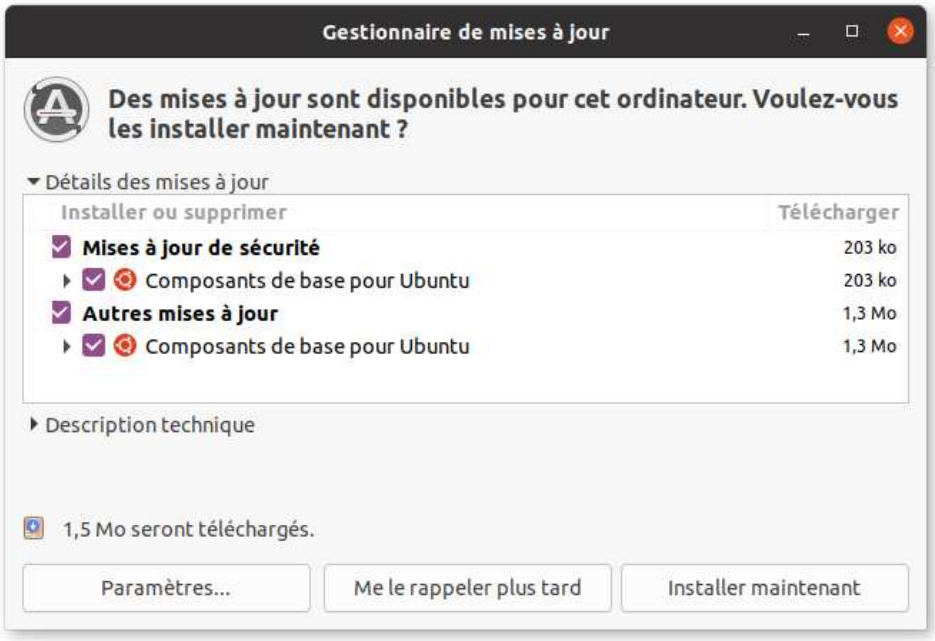


    - En utilisant le panneau d'applications, recherchez "mises à jour":



2. Après avoir appuyé sur Enter , ou cliqué sur l'icône appropriée, le système commencera à rechercher les mises à jour:

3. Si des mises à jour sont disponibles, alors le dialogue "**`Gestionnaire de mises à jour`**" apparaîtra:



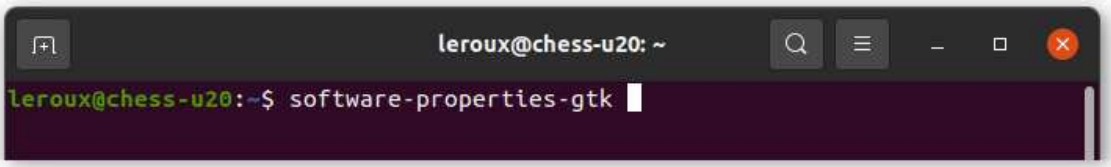
Pour mettre à jour l'ordinateur, cliquez simplement sur "**`Installer maintenant`**" et suivez la procédure.

5.2.3.2 Activer les dépôts des partenaires

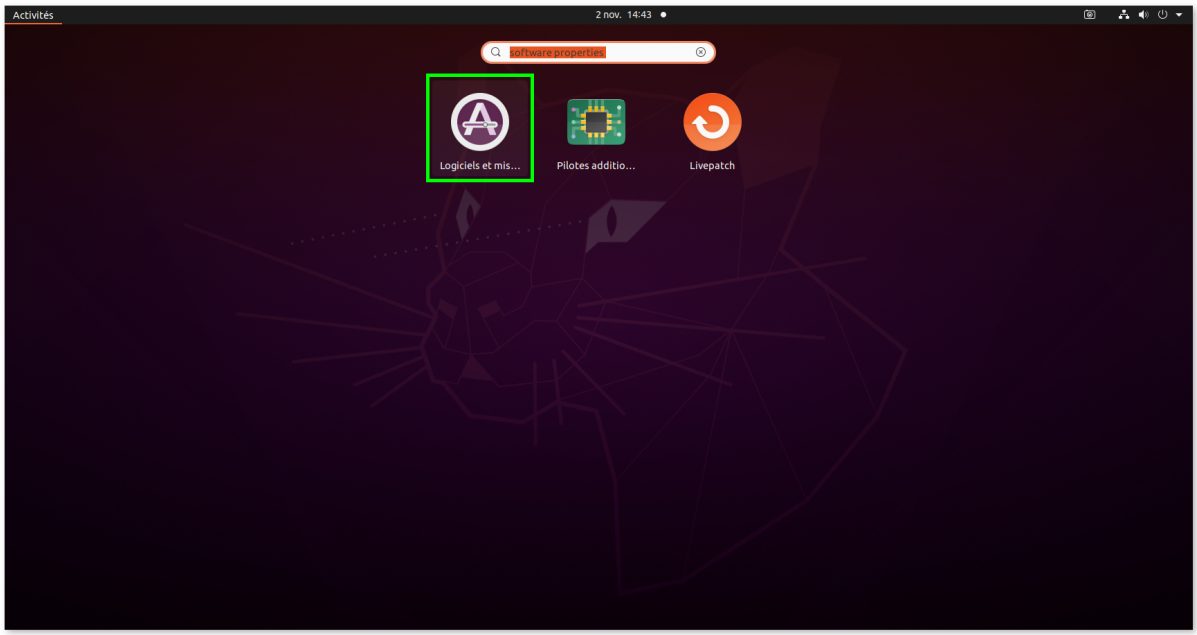
Certains logiciels tiers sont inclus dans le composant multiverse d'Ubuntu que vous devrez peut être activer. Le paquet "ubuntu-restricted-extras" contient également des logiciels qui peuvent être légalement restreints, notamment la prise en charge de la lecture de fichiers MP3 et DVD, les polices TrueType de Microsoft, l'environnement d'exécution Java de Sun, le plugin Adobe Flash Player, de nombreux codecs audio/vidéo courants, et unrar, un désarchiveur pour les fichiers compressés dans le format de fichier RAR.

1. Ouvrez l'utilitaire de configuration des mises à jour:

   - En utilisant le terminal, entrez simplement la commande: `software-properties-gtk`



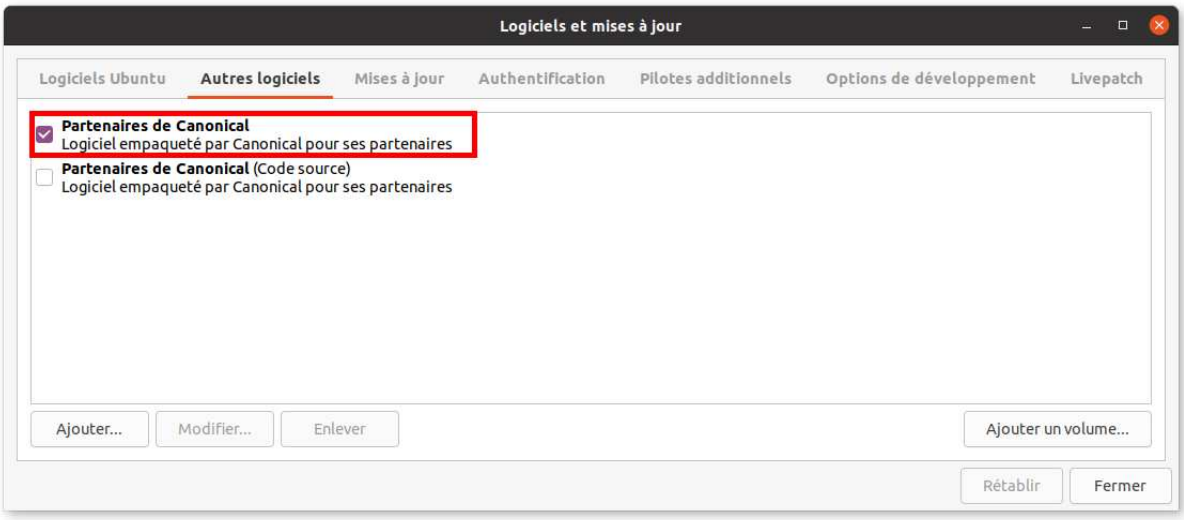
   - En utilisant le menu d'applications, recherchez "propriétés du logiciel":



2. Ensuite, le dialogue "**`Logiciels et mises à jour`**" apparaîtra:

   Sur l'onglet "**`Ubuntu software`**", vous devez vous assurer que toutes les options sont cochées comme illustré.

3. Après cela, basculez vers l'onglet "**`Autres logiciels`**":



   Et assurez-vous que l'option "**`Canonical partners`**" est cochée.

Après cela, votre Ubuntu aura beaucoup plus d'endroits, dépôts de logiciels ou magasins d'application, pour rechercher des paquets en ligne lorsque vous cherchez à installer des programmes.

5.2.3.3 Installez les pilotes de matériel éventuellement manquants

Sur Linux, les pilotes libres, open source, sont installés pendant le processus d'installation, et non les pilotes propriétaires créés par le fabricant de votre carte vidéo (ou d'autres périphériques). Cependant, les pilotes des fabricants sont généralement plus rapides pour rendre le contenu 3D que les pilotes open source, pour vérifier si des pilotes plus adaptés sont disponibles pour votre matériel, utilisez le dialogue "**`Propriétés du logiciel`**" et ouvrez l'onglet "**`Pilotes supplémentaires`**":

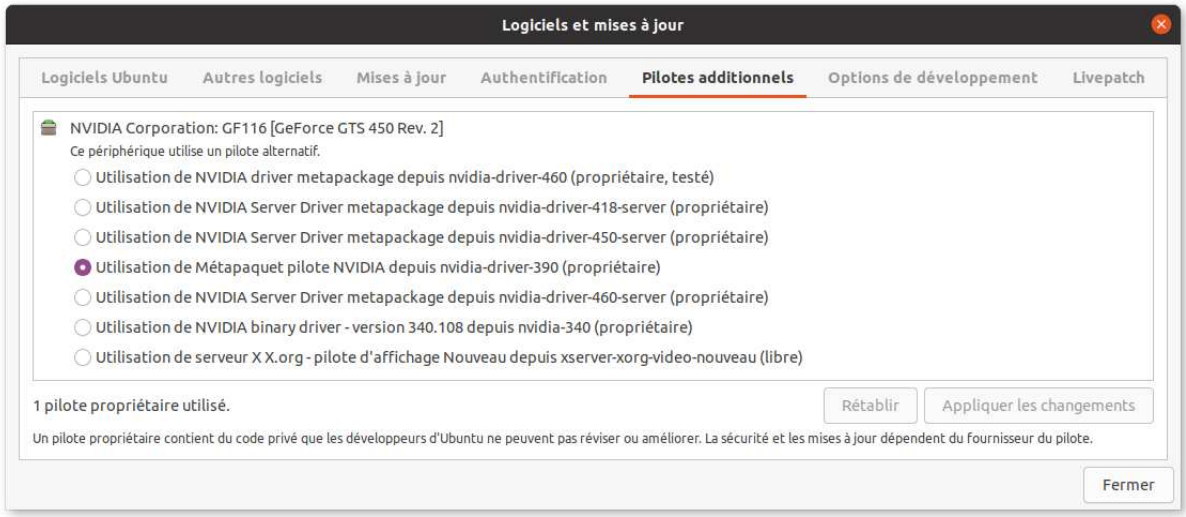


Figure 5.10: L'onglet "Pilotes supplémentaires" de "Propriétés du logiciel".

Si des pilotes sont disponibles pour votre carte vidéo, ou d'autres périphériques, ils seront probablement tous répertoriés ici, et vous devrez choisir celui (ceux) que vous pensez approprié(s). Pour les cartes graphiques, les pilotes propriétaires sont conditionnés par génération, grossièrement en fonction de l'ancienneté de votre carte. Comme illustré dans la figure [Fig. 5.10], la carte graphique de mon ordinateur est un GPU nvidia, et Ubuntu propose plusieurs pilotes différents. Si vous souhaitez installer le pilote prorpriétaire, et pour savoir lequel installer, vérifiez le site Web du fabricant de votre GPU:

- NVIDIA: https://www.nvidia.com/en-us/drivers/unix/
- AMD/(anciennement ATI): la société fournit des pilotes spécifiquement conçus pour Ubuntu, ainsi que pour d'autres Linux, vous n'aurez donc peut-être qu'une seule option à choisir, sinon vous pouvez simplement rechercher le paquet approprié sur le site Web suivant:

  https://www.amd.com/en/support

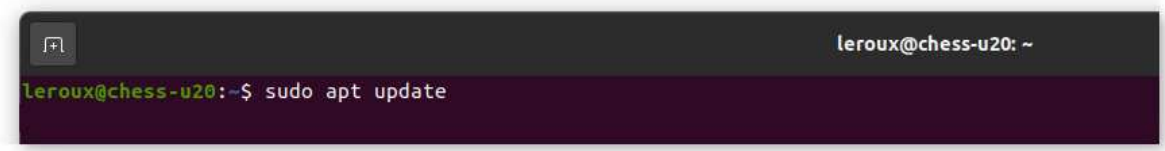
  Le site Web fournit également de l'aide pour le processus d'installation ici:

  https://amdgpu-install.readthedocs.io/en/latest/

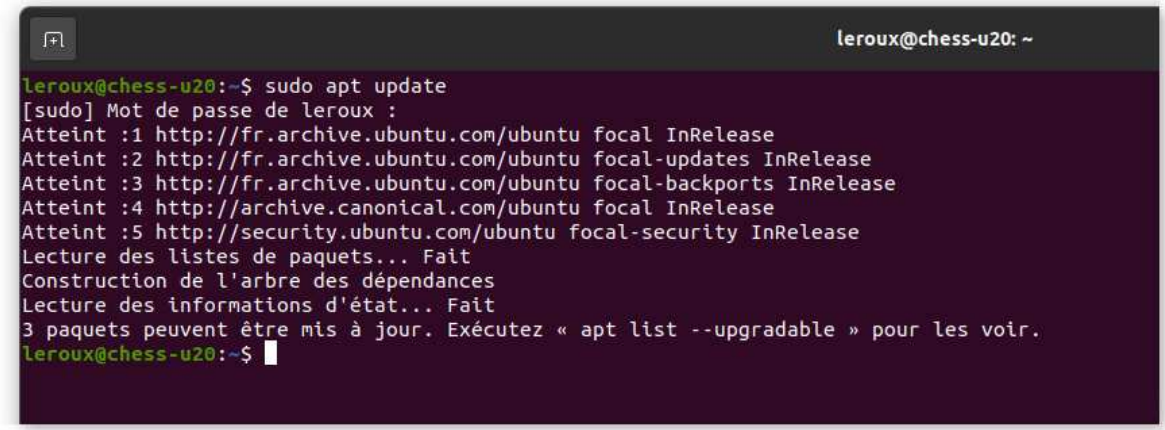
5.2.3.4 Installez des codecs multimédia supplémentaires

Les codecs multimédia sont essentiels pour lire des fichiers audio et vidéo. Par défaut, seuls les codecs multimédia open source sont installés sur Ubuntu 24.03 LTS. Pour installer tous les autres codecs et outils, vous devrez faire ce qui suit (je vais utiliser ici quelques commandes qui seront discutées dans la section [Sec. 5.2.4]:

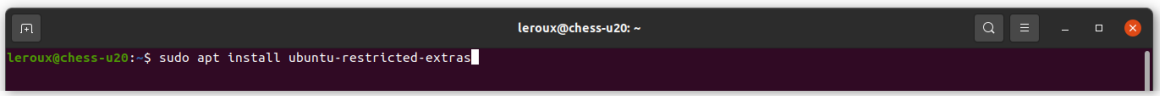
1. Ouvrez le terminal (voir section [Sec. 5.2.2.3]
2. Mettez à jour la base de donnée des logiciels disponibles (juste pour s'assurer que ce qui a été fait dans la section [Sec. [5.2.3.2] est actif):
   - Entrez la commande: **sudo apt update**



   - Appuyez sur Enter, entrez votre mot de passe et suivez la procédure, vous devriez finir avec quelque chose comme:

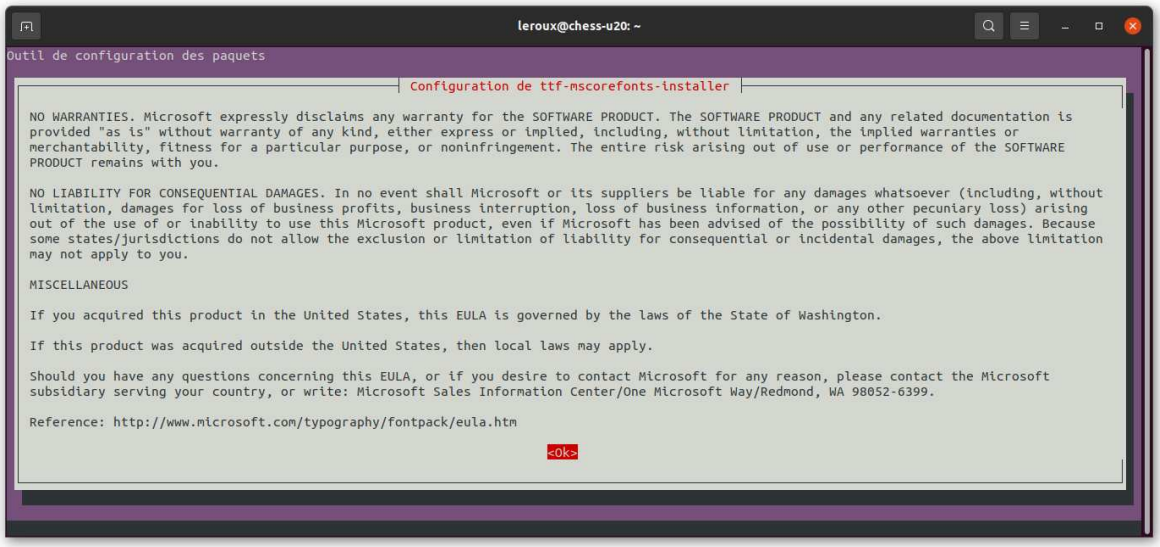


3. Maintenant, pour installer les codecs:
   - Entrez la commande: **sudo apt install ubuntu-restricted-extras**

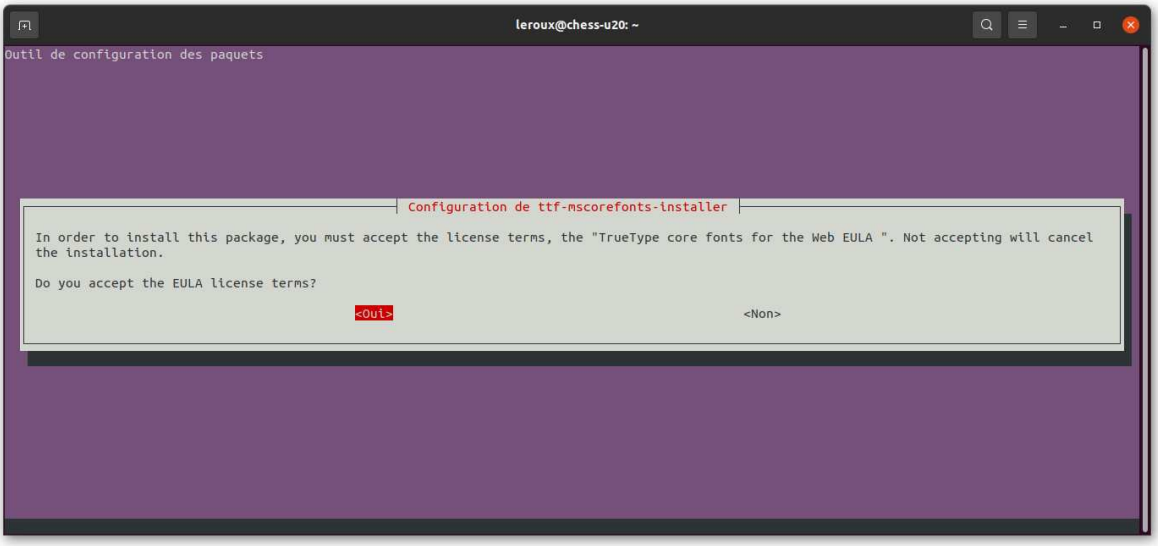


- Après avoir appuyé sur Enter , vous devriez finir avec quelque chose comme:

- Suivez la procédure pour installer les codecs et les paquets supplémentaires, à un moment donné, l'installateur propose d'installer les polices MS pour Linux (ce qui peut être utile pour des raisons de compatibilité avec vos anciens documents pendant votre transition vers le monde ouvert et Linux) et vous finirez avec l'écran suivant:



- Appuyez sur Tab/ tab pour activer le bouton `<Ok>`, puis appuyez sur Entrée et validez un choix pour installer ou non les polices:



4. Après cela, beaucoup de texte est susceptible d'apparaître dans le terminal, ne vous inquiétez pas, lorsque c'est terminé, les codecs et les utilitaires supplémentaires seront installés.

5.2.4 Installation de nouveaux logiciels

5.2.4.1 Utilisation du terminal et de `apt`

Pour installer des logiciels, ouvrez le terminal et utilisez **`apt`**:

```
user@localhost:~$ sudo apt install nom-de-paquet
```

Exemple:

```
user@localhost:~$ sudo apt install synaptic
```

Vous pouvez voir que l'installation de logiciels nécessite d'emprunter les privilèges d'administrateur en utilisant commande "**`sudo`**" vous devrez donc être dans le groupe des "**`sudo`**`ers`" pour effectuer cette action.
Tous les autres exemples fournis par la suite pour installer des logiciels sont des outils qui combinent un moteur de recherche, pour rechercher des programmes, et une interface graphique aux commandes **`sudo`**+**`apt`**.

5.2.4.2 Utilisation de `Ubuntu Software`

Pour les débutants, la façon la plus simple d'installer des logiciels sur Ubuntu est d'utiliser le dialogue "**`Ubuntu Software`**". Pour l'ouvrir, vous pouvez soit:

- Utiliser la souris et cliquer sur l'icône appropriée dans le jeu de la partie gauche du bureau.
- Ouvrir le terminal et utiliser **`snap-store`**.

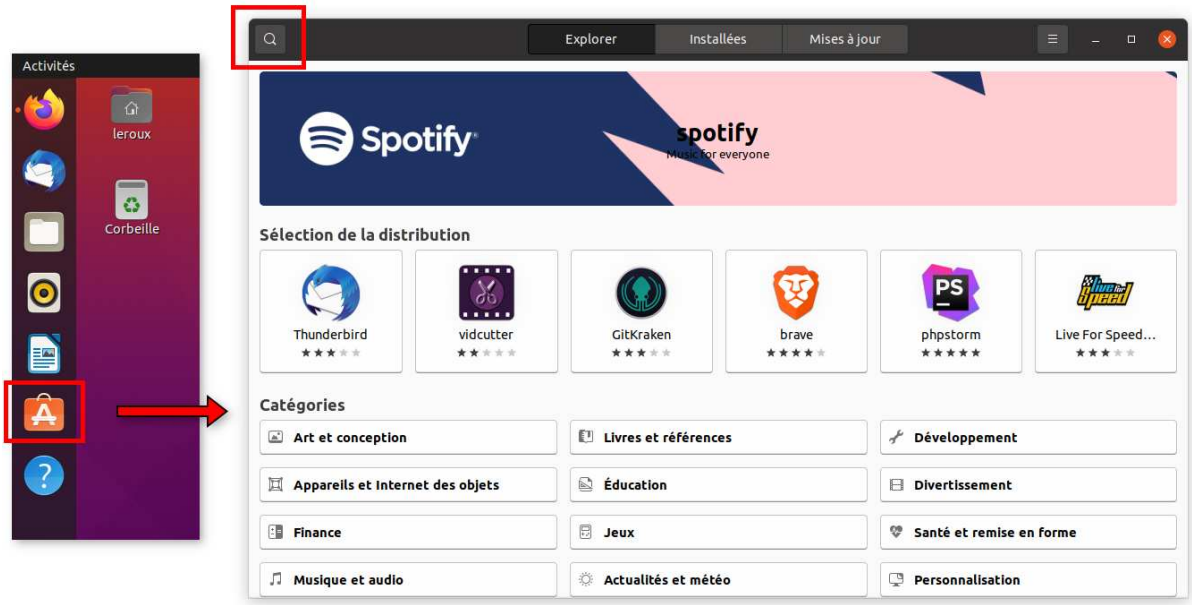


Figure 5.11: Le dialogue "**`Ubuntu Software`**" dans Ubuntu.

A partir de là, il est très facile de parcourir la bibliothèque de logiciels et de choisir quelque chose à installer. Les menus du bas offrent de parcourir la bibliothèque de logiciels par catégories, si vous ne trouvez pas ce que vous cherchez, utilisez le moteur de recherche en cliquant sur l'icône de recherche en haut à gauche.

5.2.4.3 Utilisation de `Synaptic`

"**`Synaptic`**" est un installateur de logiciels très avancé et populaire dans le monde Linux, il offre un moteur de recherche beaucoup plus complet et détaillé que "**`Ubuntu Software`**". Rappelez-vous que Ubuntu "Noble Numbat" 24.04 LTS peut offrir jusqu'à 20 000 programmes à installer et qu'il peut être difficile de se retrouver dans une bibliothèque aussi vaste.
Le prérequis pour pouvoir utiliser **`Synaptic`**, est de l'installer en utilisant **`Ubuntu Software`**, ou **`apt`**:

```
user@localhost:~$ sudo apt install synaptic
```

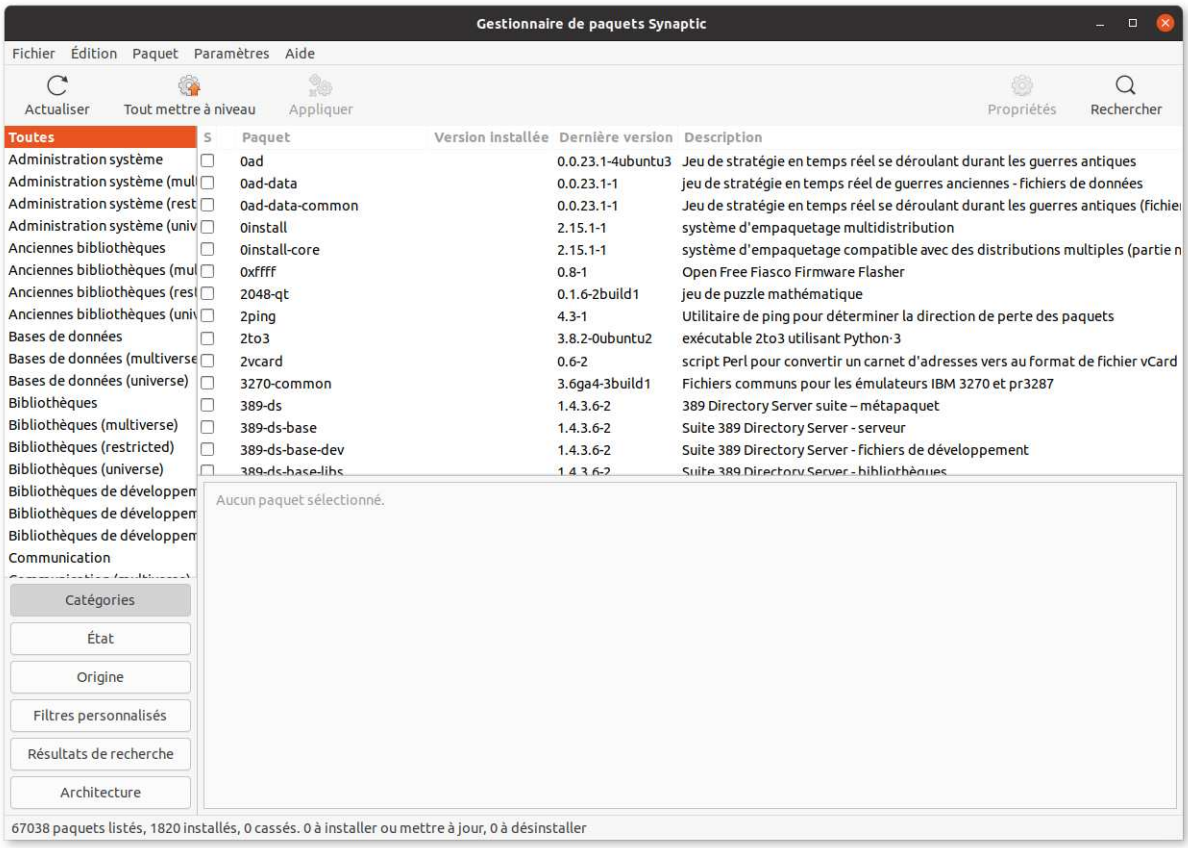


Figure 5.12: Le logiciel "**`Synaptic`**" dans Ubuntu.

5.2.4.4 Que faire si vous ne trouvez pas ce que vous cherchez ?

Si vous voulez plus de dépôts en ligne pour choisir et installer des logiciels, vous pouvez parcourir la page Web dédiée à Ubuntu

https://doc.ubuntu-fr.org/depots_focal

Notez que certains éditeurs de logiciels ne diffusent pas leurs logiciels via le système de distribution standard d'Ubuntu, vous devrez donc vérifier leur site Web et en particulier la section dédiée au téléchargement et/ou à l'installation pour Linux. Ils guident généralement la procédure du début à la fin.

5.2.5 Conseils et astuces Ubuntu

Maintenant que vous essayez ou utilisez Ubuntu, vous pourriez vouloir personnaliser son apparence et profiter pleinement du potentiel du bureau GNOME.
Vous verrez que le bureau GNOME est souvent appelé GNOME shell, c'est-à-dire GNOME et

toutes les extensions, les greffons, qui peuvent être installés pour améliorer l'expérience utilisateur. GNOME shell a été préconfiguré par l'équipe Ubuntu de la manière qu'ils pensaient appropriée pour la plupart des utilisateurs, en particulier les débutants dans le monde Linux. L'interface est très facile à gérer, mais laisse peu de place à une configuration personnalisée. Parce qu'Ubuntu a décidé de simplifier l'expérience utilisateur, ils ont également caché certains des outils nécessaires pour administrer/personnaliser GNOME shell.
Dans la suite, je vais vous montrer comment installer les outils qui amélioreront votre expérience avec GNOME shell.

5.2.5.1 Tweaks - Ajustements

Le panneau de contrôle d'Ubuntu présenté dans la section [Sec. 5.2.2.1] est l'outil par défaut fourni pour configurer GNOME shell. Cela fonctionne parfaitement, mais fournit une expérience utilisateur limitée de ces possibilités, pour améliorer cela, vous devrez installer le programme "**Tweaks**" ("**Ajustements**" en français). Pour ce faire, vous pouvez utiliser "**Ubuntu Software**", "**Synaptic**" (rechercher "**gnome-tweaks**") ou la commande:

```
user@localhost:~$ sudo apt install gnome-tweaks
```

Tweaks fournit une interface complète pour personnaliser le bureau GNOME (thèmes utilisateur, aspect visuel, système de fenêtres ...), vous pouvez lancer **Tweaks** via les menus d'applications, vous trouverez généralement le lanceur approprié dans le sous-menu "Utilitaires" (voir figure [Fig. 5.13]) Alternativement, vous pouvez entrer la commande **gnome-tweaks** dans le terminal:

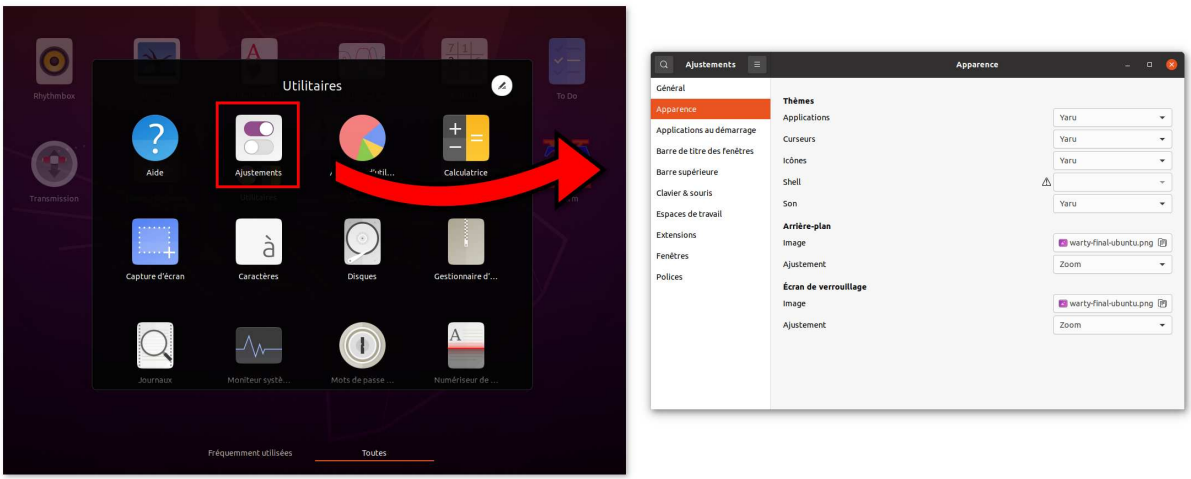


Figure 5.13: "**Tweaks**" ("**Ajustements**") dans Ubuntu "Jammy Jellyfish" 22.04 LTS.

Tweaks permet d'installer des thèmes utilisateur qui peuvent être trouvés sur le site:

https://www.gnome-look.org/

5.2.5.2 Extensions

Dans Ubuntu, les extensions de GNOME shell ne peuvent pas être gérées en utilisant le panneau de contrôle d'Ubuntu ni en utilisant **gnome-tweaks**, pour ce faire, il est nécessaire d'installer:

- Ubuntu "Jammy Jellyfish" 22.04 LTS: **gnome-shell-extension-prefs**

```
user@localhost:~$ sudo apt install gnome-shell-extension-prefs
```

Extensions fournit une interface pour activer/désactiver toutes les extensions de GNOME shell installées, pour lancer **Extensions** ouvrez le menu d'applications et recherchez "**extensions**", puis cliquez sur le lanceur approprié. Alternativement, vous pouvez entrer la commande **gnome-shell-extension-prefs** dans le terminal:

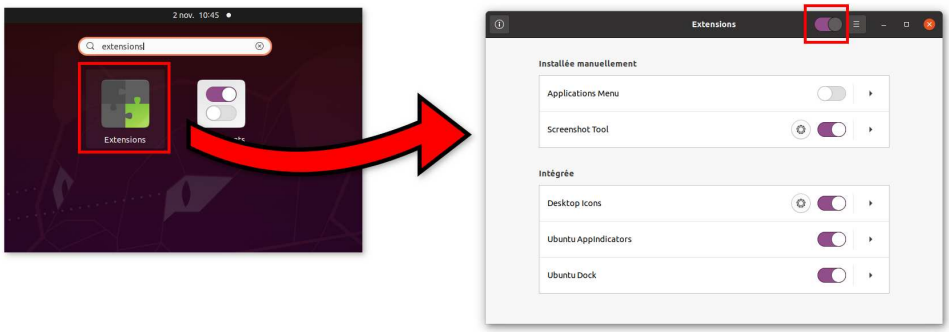


Figure 5.14: Le dialogue "**Extensions**" dans Ubuntu "Jammy Jellyfish" 22.04 LTS.

Extensions permet d'activer/désactiver chaque extension de GNOME shell, mais également toutes les extensions simultanément en utilisant le bouton situé dans la barre de titre de la fenêtre de l'application (voir figure [Fig. 5.14]). Lorsque cela est possible, **Extensions** offre également une interface de configuration pour l'extension.

- Ubuntu "Noble Numbat" 24.04 LTS: **gnome-shell-extension-manager**

```
user@localhost:~$ sudo apt install gnome-shell-extension-manager
```

Le programme **Gestionnaire d'extension** fournit une interface pour activer/désactiver toutes les extensions de GNOME shell installées, Pour lancer **Gestionnaire d'extension** ouvrez le menu d'applications et recherchez "**extension**", puis cliquez sur le lanceur approprié. Alternativement, vous pouvez entrer la commande **extension-manager** dans le terminal (voir figure [Fig. 5.15]):
Le programme "Gestionnaire d'extension" permet d'activer/désactiver chaque extension de GNOME shell, mais également toutes les extensions simultanément en utilisant le bouton situé dans la barre de titre de la fenêtre de l'application (voir figure [Fig. 5.15]).

Lorsque cela est possible, le logiciel "Gestionnaire d'extension" offre également une interface de configuration pour l'extension.

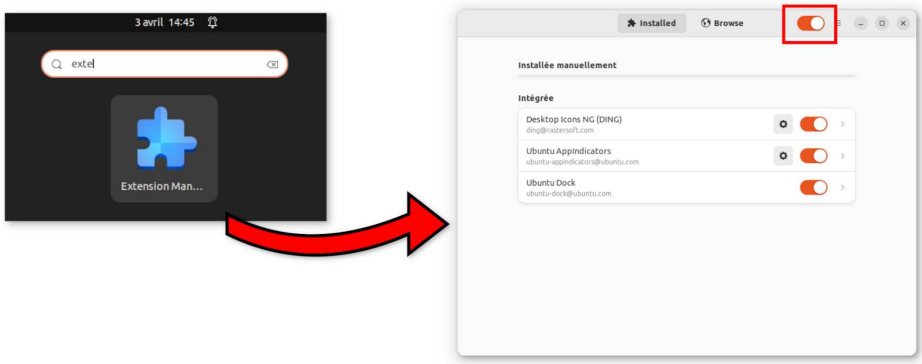


Figure 5.15: "**`Gestionnaire d'extension`**" dans Ubuntu "Noble Numbat" 24.04 LTS.

Vous trouverez aussi un onglet "Parcourir" pour rechercher et installer plus d'extensions:

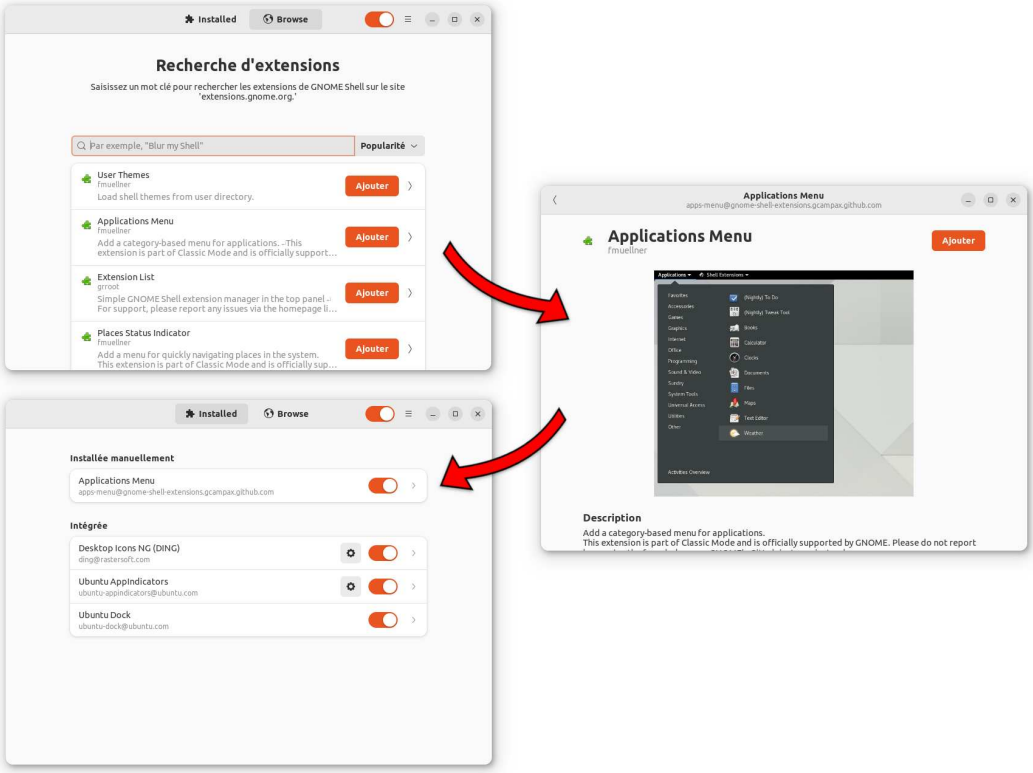


Figure 5.16: Ajouter une extension avec le programme "**`Gestionnaire d'extension`**".

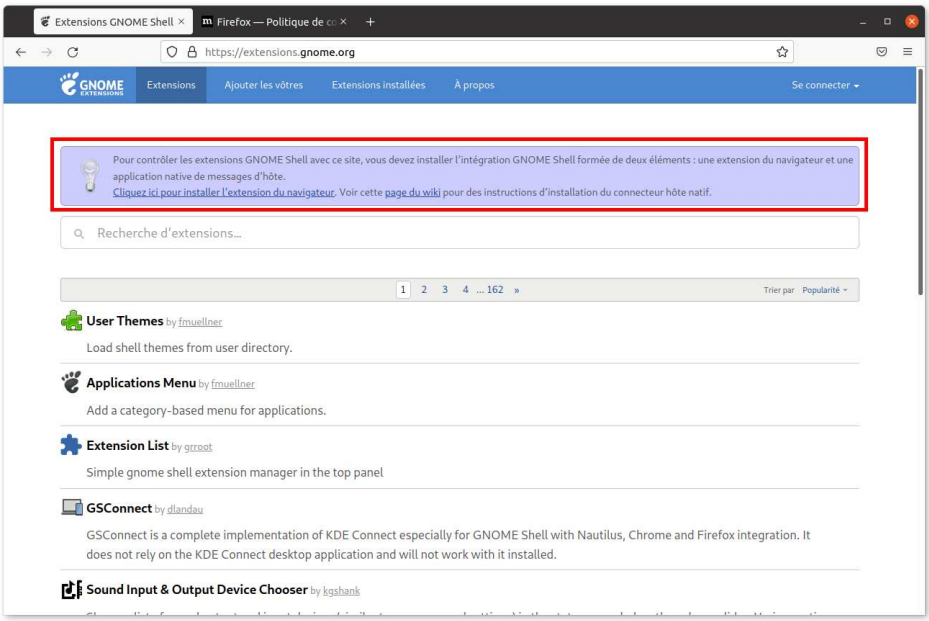
5.2.5.3 Ubuntu "Focal Fossa" 20.04 LTS uniquement: "https://extensions.gnome.org/"

Même si vous pouvez encore le faire dans Ubuntu "Noble Numbat" 24.04 LTS, la fonctionnalité pour installer facilement des extensions en ligne a été ajoutée de manière native au "**`Gestionnaire d'extension`**" de Ubuntu "Jammy Jellyfish" 22.04 LTS, donc le contenu de cette section n'est valable que pour Ubuntu "Focal Fossa" 20.04 LTS.

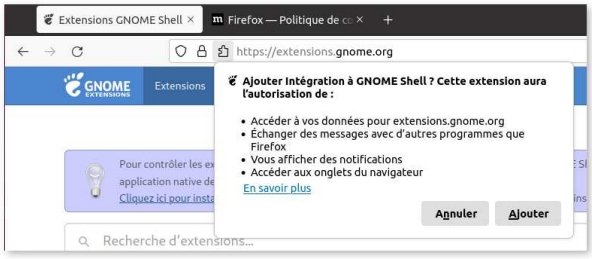
Ce site Web est la façon la plus simple de parcourir et d'installer des extensions pour le bureau GNOME:

https://extensions.gnome.org/

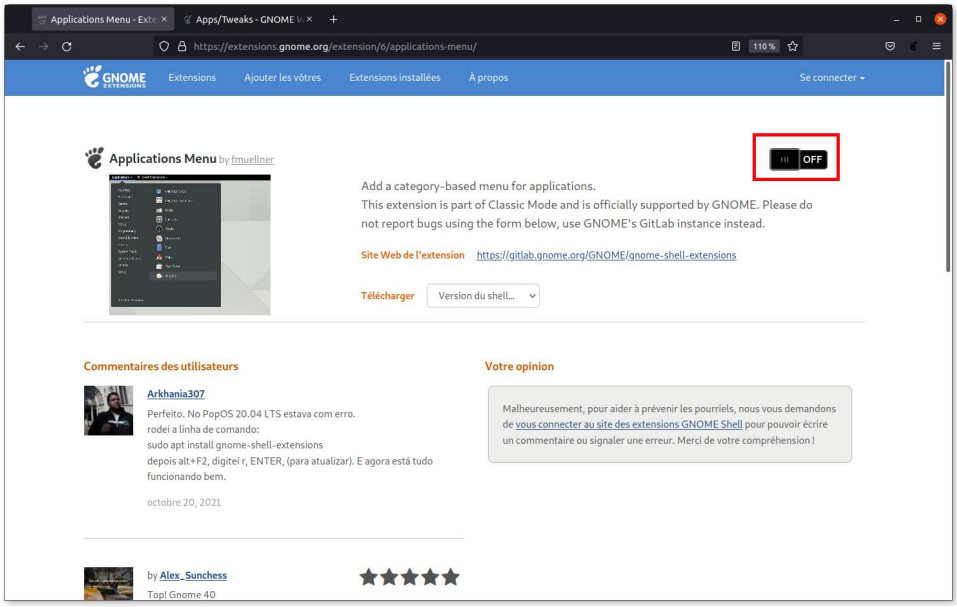
1. Lors de votre première visite, le site Web présentera un message inséré dans une bande bleue en haut:



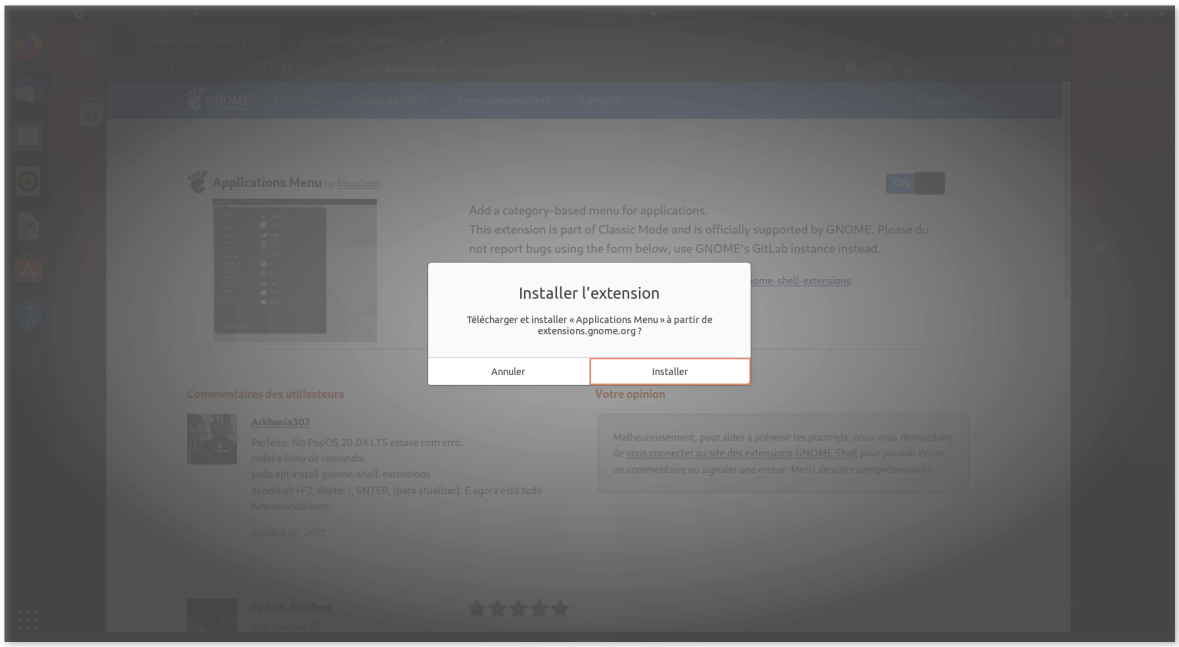
Le message invite à installer une extension de navigateur Web dédiée au site Web:



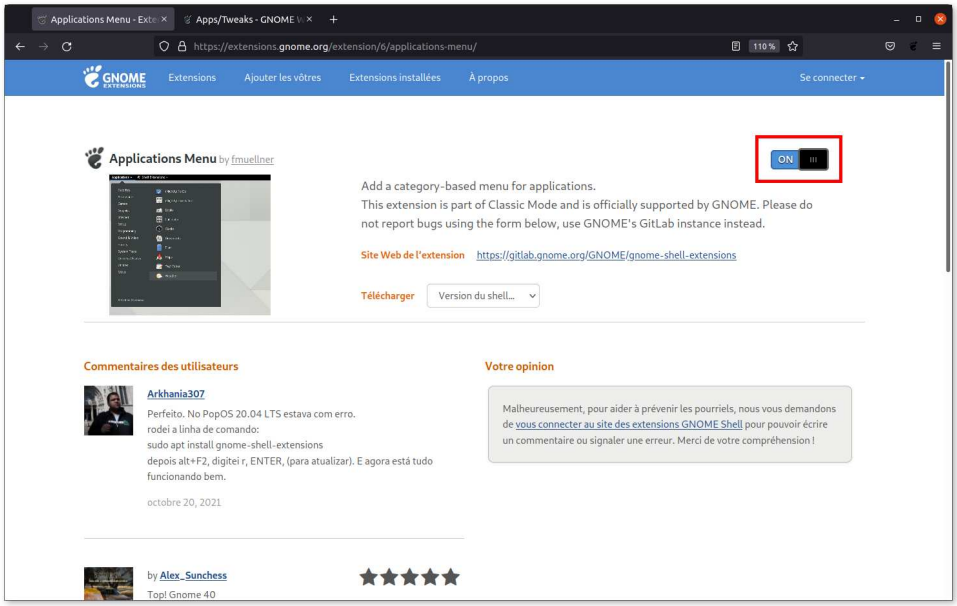
2. Suivez la procédure pour installer l'extension, puis la disposition des pages Web du site ressemblera à:



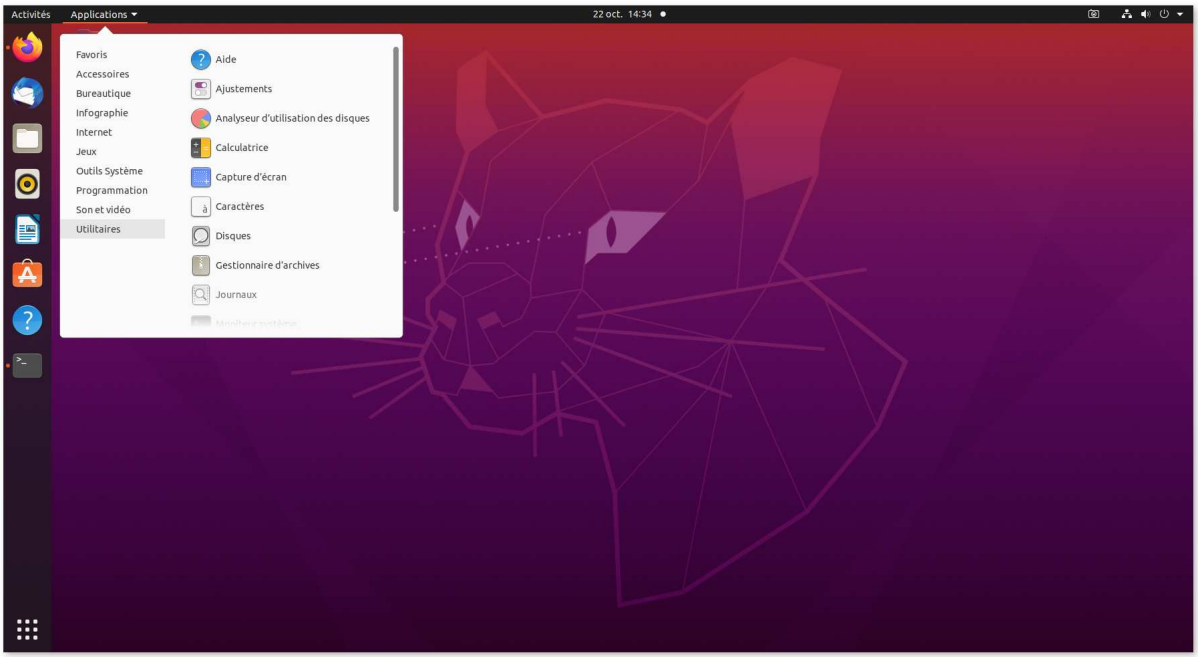
L'interrupteur offre un raccourci pour installer/activer/désactiver directement les extensions de GNOME présentées sur la page Web:



3. Si vous êtes intéressé par une extension, alors activez l'interrupteur et suivez la procédure. Lorsque la procédure d'installation est terminée, l'interrupteur reste actif et vous pouvez activer et désactiver l'extension pour GNOME shell



4. L'interrupteur s'allume, après l'installation de l'extension "Menu des applications":



Utilisation de la ligne de commande

Ce chapitre présente les bases de l'utilisation de la ligne de commande.
Les prochaines sections se concentreront sur la présentation des éléments requis pour obtenir un niveau de débutant en utilisant cette interface texte avec votre ordinateur.
Les étapes de ce processus peuvent être résumées en répondant aux questions suivantes:

- Qu'est-ce qu'un interpréteur de commande ?
- Qu'est-ce qu'une commande ?
- Où trouver des commande(s) ?
- Comment exécuter une commande ?
- Comment utiliser une commande ?
- Comment obtenir de l'aide à l'utilisation des commande(s) ?
- Quelles sont les commandes de base ?
- Qu'est-ce qu'un filtre ?
- Qu'est-ce qu'une redirection ?
- Qu'est-ce que l'écriture de scripts ?

En répondant à ces questions, quelques exemples seront fournis pour illustrer quelques utilisations possibles de la ligne de commande.

6.1 Qu'est-ce qu'un interpréteur de commande ?

Un interpréteur de commande, également appelé shell, est un programme qui permet aux utilisateurs d'interagir avec le système en saisissant des commandes. L'interpréteur lit les commandes saisies par l'utilisateur, les interprète et les exécute. Il fournit une interface en ligne de commande pour accéder au système d'exploitation et à ses utilitaires.
Il existe plusieurs interpréteurs disponibles sous Linux, notamment BASH, TCSH, ZSH, KSH ... (le nom se termine généralement par SH pour SHell), chacun avec son propre ensemble de fonctionnalités et de capacités.
Comprendre comment utiliser un interpréteur de commande est essentiel pour quiconque souhaite travailler avec Linux.

6.1.1 Le BASH "Bourne-Again" SHell

BASH, ou GNU BASH, est le shell, ou interpréteur de commande, qui apparaît par défaut dans la plupart des systèmes d'exploitation GNU.
BASH propose entre autre les fonctions suivantes:

- Édition de ligne de commande
- Historique de commande de taille illimitée
- Gestion des processu
- Fonctions et alias de shell
- Tableaux indexés de taille illimitée
- Calcul arithmétique en toute base de deux à soixante-quatre

Le manuel est disponible en ligne à http://www.gnu.org/software/bash/manual/.
Vous pouvez également consulter mon Tutoriel de base pour la programmation BASH.

Scripts de démarrage BASH

Lorsque BASH démarre, il exécute des commandes présentes dans divers scripts.

- Lorsque BASH est invoqué en tant qu'interpréteur de commande à la de connexion, il:
  1. Lit et exécute d'abord les commandes du fichier `/etc/profile`, s'il existe.
  2. Recherche ensuite `~/.bash_profile` (en language Unix/Linux `~= $HOME`), `~/.bash_login` et `~/.profile`, dans cet ordre, et lit et exécute les commandes du premier qui existe et est lisible.
- Lorsqu'un shell interactif qui n'est pas un shell de connexion est lancé, BASH lit et exécute les commandes de `~/.bashrc`, s'il existe.
- Lorsqu'un shell de connexion se termine, BASH lit et exécute les commandes du fichier `~/.bash_logout`, s'il existe.

Commandes natives de BASH

Les commandes suivantes sont les plus importantes à connaître pour commencer à utiliser BASH:

- Pour afficher quelque chose, la commande **echo**:
  - Pour afficher du texte:

```
user@localhost:~$ echo "Ceci est le texte à afficher !"
Ceci est le texte à afficher !
user@localhost:~$
```

  - Pour afficher la valeur d'une variable d'environnement, en utilisant le symbole "**$**":

```
user@localhost:~$ echo $HOME
/home/user
user@localhost:~$
```

    Sans le symbole "**$**":

```
user@localhost:~$ echo HOME
HOME
user@localhost:~$
```

- Pour imprimer le répertoire de travail actif, la commande **pwd**, voir section [Sec. 6.7.1.2].
- Pour imprimer l'historique des commandes, la commande **history**:

```
user@localhost:~$ history
   48  cd ..
   49  ls
   50  cp Ethanol.xyz ~/Documents
   52  ls -l ~/Documents
   53  cd Documents
user@localhost:~$
```

- Pour créer des alias, la commande **alias**, voir section [Sec. 6.10].
- Pour rendre des variables d'environnement héritables, la commande **export**, voir section [Sec. 6.10].

Caractères spéciaux et protection de caractères

Voici quelques-uns des caractères spéciaux dans BASH, c'est-à-dire des caractères qui ont déjà un sens, raccourci de commande ou autre, qui prendront priorité sur l'affichage du caractère lui-même:

- L'espace " ", caractère d'espace: à éviter dans les noms de fichiers et de répertoires. Les espaces sont les séparateurs de champ par défaut sur la ligne de commande, il est donc préférable d'éviter les espaces dans les noms de fichiers et de répertoires.
- Le caractère `$` dollar: ce qui suit est une variable.
- Le caractère `*` étoile: représente n'importe quel nombre de caractères = tout.
- Le caractère `?` point d'interrogation
- Les caractères de parenthèses `{}` `()` `[]`: pour englober des expressions.
- Le caractère de barre oblique `/`: pour définir les chemins de fichiers et de répertoires.
- Les caractères de guillemets inversés `` ` `` `` ` ``: pour remplacer des commandes englobées
- Les caractères de guillemets simples `' '`: pour englober des commandes.
- Les caractères de guillemets doubles `" "`: pour englober des commandes avec expansion de variables.
- Le caractère dièse `#`: pour commencer un commentaire dans BASH.
- Les caractères `&` `<` `>`: pour les redirections, voir [Sec. 6.9].
- Les caractères `.` et `..`: pour la navigation dans l'arborescence des répertoires.
- Le caractère de barre oblique inverse `\`: pour protéger les autres caractères spéciaux.

La liste n'est pas exhaustive, et d'autres caractères peuvent nécessiter d'être protégés, ce qui signifie précédés d'un `\` sur la ligne de commande. Pour protéger un caractère, cela signifie indiquer à BASH de l'afficher tel quel, au lieu d'utiliser son comportement par défaut décrit ci-dessus.

Par exemple, si le répertoire "`Documents`" a été renommé "`Doc uments`" par erreur:

```
user@localhost:~$ ls Doc*
'Doc uments'
```

Pour changer de répertoire vers "`Doc uments`":

```
user@localhost:~$ cd Doc\ uments
user@localhost:~/Doc uments$
```

6.1.2 Fichiers Linux et terminologie de la ligne de commande

On peut distinguer 2 types de fichiers dans tout système informatique:

- Les fichiers texte.
- Les autres types de fichiers = Fichiers binaires = fichiers qui ne sont pas des fichiers texte.

Il est facile de travailler sur les fichiers texte en utilisant la ligne de commande, cela est plus compliqué pour les fichiers binaires.

Sur un système Linux, on peut également distinguer entre 2 catégories de fichiers et/ou de répertoires:

- Les fichiers et répertoires standards
- Les fichiers et répertoires cachés dont les noms, par convention, commencent par un point `.`, par exemple: `~/.bashrc` pour plus d'informations sur ce fichier voir section [Sec. 6.10].

Enfin, certains termes sont fréquemment utilisés pour décrire l'interaction avec les ordinateurs, en particulier lors de l'utilisation de la ligne de commande:

- Entrée standard (`STDIN`): le périphérique à partir duquel l'entrée du système est récupérée, généralement votre clavier ou votre souris.
- Sortie standard (`STDOUT`): la fenêtre du terminal qui affiche les informations de l'interpréteur de commande.
- Erreur standard (`STDERR`): les messages d'erreur potentiels des commandes.

Ces éléments peuvent apparaître lors de la recherche d'aide ou si vous souhaitez aller au delà de ce manuel.

6.2 Commande ? Est exécutable !

Puisque l'utilisation de la ligne de commande consiste à donner des commandes à l'interpréteur, la prochaine question légitime à se poser est: qu'est-ce qu'une commande sous Linux ?
La réponse à cette question est simple:

Une commande est un fichier avec la permission d'exécution = un fichier exécutable !

6.3 Où trouver des commande(s) ?

La prochaine question qui vient naturellement à l'esprit est alors: où trouver des fichier(s) qui ont la permission d'exécution ? Où trouver les commandes ?
Et il y a deux réponses à cette question:

- Dans le **PATH**
- N'importe où vous le souhaitez, car vous pouvez créer des commande(s) vous-même.

6.3.1 PATH

PATH est la variable d'environnement [EV] (voir [Sec. 4.2.5] pour plus d'informations) qui liste tous les répertoires qui contiennent potentiellement des commandes:

- Pour afficher toutes les [EV] et leurs valeurs, utilisez la commande **env**:

```
user@localhost:~$ env
```

- Pour afficher le contenu de n'importe quelle variable [EV] **VAR**, utilisez la commande **echo** + **$** symbol:

```
user@localhost:~$ echo $VAR
```

Pour exemple, pour afficher le contenu de **PATH**:

```
user@localhost:~$ echo $PATH
/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:
/usr/games:/usr/local/games:/snap/bin
user@localhost:~$
```

Cette commande affiche une liste de répertoires séparés par des symboles **:**.
Lorsque l'interpréteur de commande recherche des commande(s) saisies via la ligne de commande, il recherche successivement dans tous les répertoires listés dans le **PATH** pour cette commande particulière. Selon l'exemple précédent, il recherche:

- Tout d'abord dans: `/usr/local/sbin`
- Ensuite dans: `/usr/local/bin`
- Puis dans: `/usr/sbin`
- Et ainsi de suite ...

La recherche s'arrête à la première occurrence de la commande dans n'importe quel répertoire.

Vous pouvez lister le contenu de n'importe quel répertoire listé dans le **PATH** pour voir quelles commandes sont disponibles à l'intérieur.

6.3.2 Créer une commande

Pour ce faire, il suffit de donner le droit d'exécution au fichier que vous souhaitez exécuter, c'est-à-dire utiliser comme commande, pour ce faire, utilisez la commande **chmod** (voir section [Sec. 4.2.4.1] pour plus d'informations):

```
user@localhost:~$ chmod 700 MaCommande
```

Maintenant que **MaCommande** a les permissions de fichier appropriées, il est possible de tenter de l'exécuter et de l'utiliser comme une commande.

6.3.3 Localiser une commande

Pour savoir où est situé une commande dans l'arborescence du système de fichiers, utilisez la commande **which**:

```
user@localhost:~$ which ls
/usr/bin/ls
user@localhost:~$
```

Cela localisera la commande que vous souhaitez utiliser, à condition que cette commande soit dans le **PATH**.

6.3.4 Ajouter une commande au PATH

Voir section [Sec. 6.10].

6.4 Comment exécuter une commande ?

La réponse à cette question dépend de la manière de localiser la commande pour l'interpréteur:

- La commande est dans le **PATH**, alors utilisez simplement son nom:

```
user@localhost:~$ MaCommande
```

- La commande n'est pas dans le **PATH**:
  - Utilisez le chemin de répertoire complet suivi du nom de l'exécutable:

```
user@localhost:~$ ~/Documents/MaCommande
```

ou

```
user@localhost:~$ /home/user/Documents/MaCommande
```

ou

```
user@localhost:~$ Documents/MaCommande
```

  - Changez de répertoire vers le répertoire de l'exécutable et utilisez les symboles `./`:

```
user@localhost:~$ cd Documents
user@localhost:~/Documents$ ./MaCommande
```

`./` signifie simplement: regardez dans le répertoire actif.

Notez que la commande est exécutée et aura donc un effet dans le répertoire de travail actif.

Si vous avez l'intention d'exécuter une commande dans un autre répertoire, alors:

- Changez de répertoire vers le répertoire cible de la commande, voir section [Sec. 6.7.1.3].
- Si possible, fournissez le répertoire cible de la commande comme argument.

6.5 Comment utiliser une commande ?

Les commandes peuvent recevoir à la fois:

- Des arguments, des données pour la commande à traiter:

```
user@localhost:~$ cd Documents
```

  La commande **cd** changera le répertoire de travail vers le répertoire spécifié.

- Des options, qui spécifieront comment la commande opère:

```
user@localhost:~/Documents$ ls -la
```

  La commande **ls** affichera le contenu du répertoire de travail actif, les résultats seront fournis suivant les instructions des options **-la**.

Cependant, très souvent les commandes peuvent être utilisées sans arguments ni options:

```
user@localhost:~/Documents$ ls
```

De nombreuses commandes acceptent des options.
Par convention, les options suivent le nom de la commande et commencent par:

- **-lettre(s)** où **lettre(s)** est une ou plusieurs lettres uniques, chaque lettre étant utilisée pour spécifier une option particulière, ex:

```
user@localhost:~/Documents$ ls -lh
```

  -lettre(s) options peuvent être fournies ensemble, comme précédemment, ou séparément:

```
user@localhost:~/Documents$ ls -l -h
```

- **--mot** où **mot** est un mot entier, utilisé pour spécifier une option particulière, ex:

```
user@localhost:~/Documents$ ls --human-readable
```

- Soit **-lettre(s)** et/ou **--mot(s)** :

```
user@localhost:~/Documents$ ls -lt --human-readable --all
```

Très souvent, l'option **-lettre** a un équivalent **--mot**.

```
user@localhost:~/Documents$ ls -ltah
```

est équivalent à:

```
user@localhost:~/Documents$ ls -l --all -t --human-readable
```

Chaque commande a son propre ensemble d'options et il n'y a pas d'écriture automatique pour vous aider à déterminer quoi utiliser et/ou comment l'utiliser.

6.6 Comment obtenir de l'aide ?

Comme illustré dans la section précédente, obtenir de l'aide sur la commande que vous souhaitez utiliser peut être important. Pour ce faire, vous pouvez vérifier soit:

- L'aide courte de la commande: en utilisant les options **-h** ou **--help**
- Le manuel détaillé de la commande: en utilisant la commande **man**.

6.6.1 Les options -h ou --help

Par convention, la plupart des commandes acceptent les options **-h** ou **--help**, parfois les deux.
Alors la commande affichera simplement un message d'aide court:

```
user@localhost:~$ mkdir --help
Utilisation: mkdir [OPTION]... RÉPERTOIRE...
Créer le ou les RÉPERTOIREs s'ils n'existent pas.

Les arguments obligatoires pour les options longues le sont aussi pour les
options courtes.
  -m, --mode=MODE   set file mode (as in chmod), not a=rwx - umask
  -p, --parents     no error if existing, make parent directories as needed,
                    with their file modes unaffected by any -m option.
  -v, --verbose     print a message for each created directory
  -Z                   définir le contexte de sécurité SELinux de tous les
                         répertoires créés au type par défaut
      --context[=CTX]  comme -Z ou, si CTX est indiqué, définir le contexte de
                         sécurité SELinux ou SMACK à CTX
      --help     afficher l'aide et quitter
      --version  afficher des informations de version et quitter

Aide en ligne de GNU coreutils: <https://www.gnu.org/software/coreutils/>
Signalez les problèmes de traduction à: <traduc@traduc.org>
Documentation complète <https://www.gnu.org/software/coreutils/mkdir>
ou disponible localement via: info '(coreutils) mkdir invocation'
```

6.6.2 La commande man

La commande **man** permet d'accéder au manuel de la commande, avec généralement plus d'informations détaillées.
Les pages de manuel peuvent être longues et il peut parfois être un peu compliqué d'obtenir les informations que vous recherchez. Pas de souci ici, voici un truc pour vous aider, en utilisant comme exemple les pages de manuel de la commande **grep** (voir [Sec. 6.8]):

- Accédez aux pages de manuel de la commande **grep**:

```
user@localhost:~$ man grep
```

- La première page devrait ressembler à:

```
GREP(1)                                Commandes de l'utilisateur                                GREP(1)

NOM
       grep, egrep, fgrep - Afficher les lignes correspondant à un motif donné

SYNOPSIS
       grep [OPTION...] MOTIF [FICHIER...]
       grep [OPTION...] -e MOTIF ... [FICHIER...]
       grep [OPTION...] -f MOTIF_FICHIER ... [FICHIER...]

DESCRIPTION
       grep  cherche  un  MOTIF  dans  chaque  FICHIER.  MOTIF  est  un  ou  plusieurs  motifs  séparés
       par un retour à la ligne et grep affiche chaque ligne correspondant à un motif.
       Généralement, MOTIF devrait être entre guillemets lorsque grep est utilisé dans un interpréteur
       de commandes.

       Un fichier « - » signifie l'entrée standard. Si aucun FICHIER n'est donné, les recherches
       récursives explorent le répertoire de travail et celles non récursives lisent l'entrée
       standard.

       En  outre,  les  variantes  egrep  et  fgrep  sont respectivement les mêmes que les programmes
       grep -E et grep -F. Ces variantes sont obsolètes mais sont fournies pour la rétro
       compatibilité.

OPTIONS
   Informations générales sur le programme
       --help Afficher un message d'utilisation et quitter.

       -V, --version
              Afficher le numéro de version de grep et quitter.

   Syntaxe du motif
       -E, --extended-regexp
              Interpréter le MOTIF comme une expression rationnelle étendue (ERE, voir ci-dessous).

       -F, --fixed-strings
              Interpréter le MOTIF comme étant une chaîne figée, pas une expression rationnelle.
```

- Avec une invite sur la dernière ligne pour saisir des instructions:

```
Manual page grep(1) line 1 (press h for help or q to quit)
```

- Vous pouvez utiliser cette invite pour rechercher dans ces pages de manuel en utilisant la commande **/** suivie d'un motif ou d'un mot clé que vous recherchez:

```
/mot
```

- Appuyez sur Enter pour initier la recherche de ce motif dans le document, ici **mot**, et l'écran sera rafraîchi pour afficher le texte contenant la première occurrence trouvée.
- Après cela, appuyez sur n pour sauter à la prochaine occurrence, et ainsi de suite.

6.7 Commandes de base

Cette section présente les commandes de base et les options associées.
J'ai décidé de regrouper les commandes comme suit:

- Gestion du système de fichiers: **ls**, **pwd**, **cd**, **touch**, **mkdir**, **mv**, **cp**, **rmdir**, **rm**
- Affichage de fichiers: **wc**, **cat**, **tac**, **more**, **tail**, **cut**
- Gestion de fichiers: **chown**, **chmod**, **diff**, **ln**

6.7.1 Options de gestion du système de fichiers

6.7.1.1 La commande `ls`

Pour lister le contenu d'un répertoire:

- **ls** *"lister le contenu"*
  - usage: **ls** [OPTION] ... [ARG]
  - ex: **user@localhost:~$ ls**
  - options: **-l** (liste détaillée), **-t** (tri par date), **-a** (fichiers cachés), **-h** (lisible par l'humain)

Utilisation de la commande **ls**:

```
user@localhost:~$ ls
Bureau Documents Images Modèles Musique Public snap Téléchargements Vidéos
user@localhost:~$
```

Les options peuvent être utilisées pour modifier la sortie de la commande **ls**:

- Liste détaillée, en utilisant: **-l**

```
user@localhost:~$ ls -l
total 36
drwxr-xr-x.  2 user ipcms 4096 avril 12 11:04 Bureau
drwxr-xr-x.  2 user ipcms 4096 avril 12 11:04 Documents
drwxr-xr-x.  2 user ipcms 4096 avril 12 11:04 Images
drwxr-xr-x.  2 user ipcms 4096 avril 12 11:04 Modèles
drwxr-xr-x.  2 user ipcms 4096 avril 12 11:04 Musique
drwxr-xr-x.  2 user ipcms 4096 avril 12 11:04 Public
drwx------.  2 user ipcms 4096 avril 12 11:04 snap
drwxr-xr-x.  2 user ipcms 4096 avril 12 11:04 Téléchargements
drwxr-xr-x.  2 user ipcms 4096 avril 12 11:04 Vidéos
user@localhost:~$
```

La syntaxe de la ligne de sortie de la commande "**ls -l**" est la suivante:

- "-" signifie "fichier", alternativement "d" signifie "répertoire" et "l" signifie "lien symbolique" (voir section [Sec. 6.7.3.4]).
- "rwxr-xr-x" sont les permissions sur le fichier.
- "2" est le nombre de liens physiques du fichier avec le disque dur.
- "user" est le nom du propriétaire du fichier.
- "ipcms" est le nom du groupe auquel appartient le propriétaire du fichier.
- "4096" est la taille du fichier sur le disque dur.
- "avril 12 11:04" est la date et l'heure de dernière modification du fichier.

- Liste des fichiers, y compris les fichiers cachés, en utilisant: **-a** ou **--all**

```
user@localhost:~$ ls -a
.              .bash_logout   .cache     Images   Musique  snap
..             .bashrc        .config    .local   .profile  Téléchargements
.bash_history  Bureau      Documents  Modèles  Public   Vidéos
user@localhost:~$
```

- Liste du contenu de l'arborescence du répertoire, en utilisant: **-R**
  Affichage récursif de tous les fichiers et répertoires de l'arborescence du système de fichiers, voir section [Sec. 6.7.1.4].

Par défaut, la commande **ls** liste le contenu du répertoire actif.
Cependant, les arguments peuvent être utilisés pour spécifier l'élément, fichier ou répertoire à lister:

```
user@localhost:~$ ls /home
user  lost+found
user@localhost:~$
```

ou de manière similaire:

```
user@localhost:~$ ls -l /home
total 20
drwx------. 53 user   ipcms   4096 14 avril 16:25 user
drwx------.  2 root   root   16384 14 oct.   2020 lost+found
user@localhost:~$
```

La commande précédente liste le contenu du répertoire **/home**, qui contient:

- Le répertoire personnel de l'utilisateur **user** ou **/home/user** ou **~**
- Un répertoire **lost+found** qui est une construction automatique du système de fichiers. Vous trouverez un répertoire **lost+found** à la racine de chaque partition Linux.

6.7.1.2 La commande `pwd`

Pour afficher le chemin complet du répertoire actif dans le système de fichiers:

- **`pwd`** *"afficher le répertoire actif"*
  - usage: `pwd`
  - ex: `user@localhost:~$ pwd`

Utilisation de la commande `pwd`:

```
user@localhost:~$ pwd
/home/user
user@localhost:~$
```

La commande `pwd` indique simplement où vous êtes.

6.7.1.3 La commande `cd`

Pour changer de répertoire:

- **`cd`** *"changer de répertoire"*
  - usage: `cd [ARG]`
  - ex: `user@localhost:~$ cd MonDossier`

Utilisation de la commande `cd`:

```
user@localhost:~$ cd Documents
user@localhost:~/Documents$
user@localhost:~/Documents$ pwd
/home/user/Documents
user@localhost:~/Documents$
```

Les conventions d'écriture existent pour vous aider à naviguer dans l'arborescence du système de fichiers:

- Changer de répertoire pour le répertoire personnel de l'utilisateur **`$HOME`** ou **`~`**:

  ```
  user@localhost:~$ cd
  ```

- Revenir au répertoire précédent, en utilisant: `-`

  ```
  user@localhost:~$ cd -
  ```

- Changer de répertoire pour le n^{ime} répertoire parent, en utilisant: `../` × (n-1) + `..`
  - Un répertoire vers le haut, en utilisant: `..`

```
user@localhost:~$ cd ..
```

  - Deux répertoires vers le haut, en utilisant: `../..`

```
user@localhost:~$ cd ../..
```

  - Et ainsi de suite, avec `../` autant de fois que nécessaire.

6.7.1.4 La commande `mkdir`

Pour créer de nouveaux répertoires:

- **`mkdir`** *"créer un répertoire"*
  - usage: `mkdir [OPTION] ... [ARG]`
  - ex: `user@localhost:~$ mkdir MonNouveauDossier`
  - options: `-p` (avec parents)

Utilisation de la commande `mkdir`:

```
user@localhost:~/Documents$ mkdir NouveauDossier
user@localhost:~/Documents$ ls
NouveauDossier
user@localhost:~/Documents$
```

Vous pouvez également créer une arborescence complète, en utilisant: `-p`

```
user@localhost:~/Documents$ mkdir -p NouveauDossier/Level-1/Level-2/Level-3
user@localhost:~/Documents$ ls -R NouveauDossier
NouveauDossier:
Level-1
NouveauDossier/Level-1:
Level-2
NouveauDossier/Level-1/Level-2:
Level-3
NouveauDossier/Level-1/Level-2/Level-3:
user@localhost:~$ cd NouveauDossier/Level-1/Level-2/Level-3
user@localhost:~/Documents/NouveauDossier/Level-1/Level-2/Level-3$ pwd
/home/user/Documents/NouveauDossier/Level-1/Level-2/Level-3
user@localhost:~/Documents/NouveauDossier/Level-1/Level-2/Level-3$ cd
user@localhost:~$
```

6.7.1.5 La commande `touch`

La commande **`touch`** met à jour la date et l'heure de dernière modification d'un fichier ou d'un répertoire, si le nom de fichier spécifié n'existe pas, alors la commande **`touch`** créera un fichier vide avec ce nom:

- **`touch`** *"mettre à jour la date et l'heure"*
  - usage: **`touch [ARG]`**
  - ex: **`user@localhost:~$ touch MonFichier`**

Utilisation de la commande **`touch`**:

```
user@localhost:~$ touch NouveauFichier
user@localhost:~$ ls -l NouveauFichier
drwxr-xr-x.  1 user ipcms    0 avril 12 14:25 NouveauFichier
user@localhost:~$
```

Dans ce cas, la commande **`touch`** a simplement créé un fichier vide nommé `NouveauFichier`.

6.7.1.6 La commande mv

Pour déplacer des fichiers ou des répertoires, la commande **mv** peut être appliquée à la fois aux fichiers et aux répertoires, mais déplacer un répertoire déplacera également tous les fichiers que le répertoire contient avec lui:

- **mv** *"déplacer"*
  - usage: `mv [ARG1] [ARG2]`
  - ex: `user@localhost:~$ mv MonFichier MonNouveauFichier`

Utilisation de la commande **mv**:

- Déplacer des fichiers d'un emplacement à un autre:

```
user@localhost:~$ mv NouveauFichier Documents/
user@localhost:~$ ls Documents
NouveauDossier  NouveauFichier
user@localhost:~$
```

- Déplacer des répertoires d'un emplacement à un autre:

```
user@localhost:~$ mv Documents/NouveauDossier/Level-1 Documents/
user@localhost:~$ ls Documents
Level-1  NouveauDossier  NouveauFichier
user@localhost:~$ ls Documents/Level-1
Level-2
user@localhost:~$
```

- Déplacer un objet vers le répertoire actif en utilisant: `.`

```
user@localhost:~$ cd Documents
user@localhost:~/Documents$ mv Level-1/Level-2 .
user@localhost:~/Documents$ ls
Level-1  Level-2  NouveauDossier  NouveauFichier
user@localhost:~/Documents$
```

- Déplacer un objet vers le répertoire parent en utilisant: `..`

```
user@localhost:~/Documents$ cd Level-2
user@localhost:~/Documents/Level-2$ ls
Level-3
user@localhost:~/Documents/Level-2$ mv Level-3 ..
user@localhost:~/Documents/Level-2$ ls ..
Level-1  Level-2  Level-3  NouveauDossier  NouveauFichier
user@localhost:~/Documents/Level-2$
```

6.7.1.7 La commande `cp`

Pour copier des fichiers et des répertoires:

- **`cp`** *"copier"*
  - usage: `cp [ARG1] [ARG2]`
  - ex: `user@localhost:~$ cp MonFichier MonNouveauFichier`

Utilisation de la commande `cp`:

- Copier des fichiers:

```
user@localhost:~/Documents/Level-2$ cp ../NouveauFichier .
user@localhost:~/Documents/Level-2$ ls
NouveauFichier
user@localhost:~/Documents/Level-2$
```

- Copier des répertoires de manière récursive, en utilisant: `-r`

```
user@localhost:~/Documents/Level-2$ cd ..
user@localhost:~/Documents$ cp -r Level-2 Level-4
user@localhost:~/Documents$ ls
Level-1  Level-2  Level-4  NouveauDossier  NouveauFichier
user@localhost:~/Documents$ cp -r Level-2 Level-4
user@localhost:~/Documents$ cp -r Level-1 Level-4/level-2
user@localhost:~/Documents$
```

Si le répertoire de destination existe déjà, alors le contenu du répertoire initial sera ajouté au contenu du répertoire de destination existant.

6.7.1.8 La commande `rmdir`

Pour supprimer des répertoires vides:

- **`rmdir`** *"supprimer un répertoire vide"*
  - usage: `rmdir [OPTION] ... [ARG]`
  - ex: `user@localhost:~$ rmdir MonDossier`
  - options: `-p` (avec parents), `-r` (récursif), `-f` (forcer)

Utilisation de la commande `rmdir`:

- Supprimer un répertoire vide:

```
user@localhost:~/Documents$ rmdir Level-3
user@localhost:~/Documents$ ls
Level-1  Level-2  Level-4  NouveauDossier  NouveauFichier
user@localhost:~/Documents$
```

- Supprimer les parents de répertoires vides:

```
user@localhost:~/Documents$ rmdir -p Level-4/Level-2/Level-1
user@localhost:~/Documents$ ls
Level-1  Level-2  NouveauDossier  NouveauFichier
user@localhost:~/Documents$
```

6.7.1.9 La commande `rm`

Pour supprimer des fichiers et des répertoires:

- **`rm`** *"supprimer"*
  - usage: `rm [OPTION] ... [ARG]...`
  - ex: `user@localhost:~$ rm MonFichier`

Utilisation de la commande **`rm`**:

Sur Ubuntu "Noble Numbat" 24.04 LTS la commande **`rm`** est dangereuse telle quelle, en particulier parce que l'option **`-i`** n'est pas une option par défaut. Je vous conseille vivement de modifier votre fichier `~/.bashrc` pour modifier le comportement par défaut de la commande **`rm`**, voir section [Sec. 6.10].

- Supprimer des fichiers:

```
user@localhost:~/Documents$ rm NouveauFichier
user@localhost:~/Documents$ ls
Level-1  Level-2  Level-4  NouveauDossier
user@localhost:~/Documents$
```

  Notez que la commande **`rm`** **SUPPRIME DÉFINITIVEMENT** le fichier.

- Demander confirmation avant chaque suppression, en utilisant: **`-i`**

```
user@localhost:~/Documents$ rm -i Level-2/NouveauFichier
rm: supprimer 'Level-2/NouveauFichier' ? n
user@localhost:~/Documents$
```

  Pour continuer, il faut saisir o (Oui) ou n (Non).
  Pour rendre cela automatique, éditez votre fichier `~/.bashrc`, voir section [Sec. 6.10].

- Supprimer des répertoires, en utilisant: **`-r`**

```
user@localhost:~/Documents$ rm -r Level-2
user@localhost:~/Documents$
```

6.7.2 Options d'affichage de fichiers

Pour les besoins de cette section, nous considérerons que nous avons deux fichiers nommés `C60.xyz` et `Ethanol.xyz` dans le répertoire **`$HOME`**`/Documents`:

```
user@localhost:~$ ls Documents
C60.xyz  Ethanol.xyz
user@localhost:~$
```

`C60.xyz` et `Ethanol.xyz` sont des fichiers texte simples, qui contiennent respectivement les coordonnées atomiques du fullerène C_{60} et de la molécule d'éthanol, tous deux au format de fichier XYZ [1].

6.7.2.1 La commande `wc`

Pour compter les caractères, les mots et les lignes dans un fichier.

- **`wc`** *"compter les mots et les lignes"*
  - usage: **`wc`** `[OPTION]` ... `[ARG]`
  - ex: **`user@localhost:~$ wc -l MonFichier`**
  - options: `-l` (nombre de lignes)

Utilisation de la commande **`wc`**:

- Compter les caractères, en utilisant: `-m`

```
user@localhost:~/Documents$ wc -m Ethanol.xyz
525 Ethanol.xyz
user@localhost:~/Documents$
```

- Compter les mots, en utilisant: `-w`

```
user@localhost:~/Documents$ wc -w Ethanol.xyz
37 Ethanol.xyz
user@localhost:~/Documents$
```

- Compter les lignes, en utilisant: `-l`

```
user@localhost:~/Documents$ wc -l Ethanol.xyz
11 Ethanol.xyz
user@localhost:~/Documents$
```

6.7.2.2 La commande `cat`

Pour afficher le contenu d'un fichier:

- **`cat`** *"afficher le contenu"*
  - usage: `cat [ARG]`
  - ex: `user@localhost:~$ cat MonFichier`

Utilisation de la commande **`cat`**:

```
user@localhost:~/Documents$ cat Ethanol.xyz
         9

C       1.0111998889     -0.0452918889     -0.0626048889
C      -0.4620761111      0.0306281111      0.2946991111
H       1.6265438889     -0.0376928889      0.8456121111
H       1.3252608889      0.8030881111     -0.6846978889
H       1.2501238889     -0.9611748889     -0.6188868889
H      -0.7580021111     -0.8263228889      0.9315601111
H      -0.6822251111      0.9536901111      0.8665561111
H      -2.1126961111      0.0649821111     -0.6649928889
O      -1.1981291111      0.0180941111     -0.9072448889
user@localhost:~/Documents$
```

6.7.2.3 La commande `tac`

Pour afficher le contenu d'un fichier à l'envers:

- **`tac`** *"afficher à l'envers"*
  - usage: `tac [ARG]`
  - ex: `user@localhost:~$ tac MonFichier`

Utilisation de la commande **`tac`**:

```
user@localhost:~/Documents$ tac Ethanol.xyz
O      -1.1981291111      0.0180941111     -0.9072448889
H      -2.1126961111      0.0649821111     -0.6649928889
H      -0.6822251111      0.9536901111      0.8665561111
H      -0.7580021111     -0.8263228889      0.9315601111
H       1.2501238889     -0.9611748889     -0.6188868889
H       1.3252608889      0.8030881111     -0.6846978889
H       1.6265438889     -0.0376928889      0.8456121111
C      -0.4620761111      0.0306281111      0.2946991111
C       1.0111998889     -0.0452918889     -0.0626048889

         9
user@localhost:~/Documents$
```

6.7.2.4 La commande `more`

Pour afficher le contenu d'un fichier page par page:

- **`more`** *"afficher page par page"*
  - usage: **`more [ARG]`**
  - ex: **`user@localhost:~$ more MonFichier`**

Utilisation de la commande **`more`**:

```
user@localhost:~/Documents$ more C60.xyz
60

C        -1.168000       -3.284000       -0.100000
C        -0.588000       -3.148000       -1.380000
C         0.808000       -3.164000       -1.228000
C         1.096000       -3.308000        0.140000
C        -0.124000       -3.384000        0.836000
C        -2.340000       -2.576000        0.216000
C        -2.472000       -1.968000        1.480000
C        -1.428000       -2.064000        2.420000
C        -0.256000       -2.776000        2.096000
C         0.836000       -2.088000        2.664000
C         2.056000       -2.012000        1.972000
C         2.184000       -2.624000        0.708000
C         2.992000       -1.792000       -0.096000
C         2.704000       -1.648000       -1.464000
C         1.612000       -2.332000       -2.032000
C         1.016000       -1.488000       -2.988000
C        -0.380000       -1.472000       -3.140000
C        -1.184000       -2.304000       -2.336000
C        -2.360000       -1.596000       -2.012000
C        -2.936000       -1.732000       -0.736000
C         1.740000       -0.276000       -3.012000
C         2.784000       -0.376000       -2.068000
C         3.360000       -0.668000        0.668000
--Plus--(38%)
```

Le fichier `C60.xyz` est beaucoup plus long que le fichier `Ethanol.xyz`, et il n'est pas possible d'afficher l'intégralité du contenu du fichier sur une seule fenêtre de terminal. La commande **`more`** est comme une commande **`cat`** qui s'arrête à chaque page, pour continuer et afficher le contenu du fichier il suffit de presser  .

6.7.2.5 La commande `tail`

Pour afficher les dernières lignes d'un fichier:

- **`tail`** *"afficher les dernières lignes"*
  - usage: `tail [OPTION] ... [ARG]`
  - ex: `user@localhost:~$ tail -f MonFichier`
  - options: `-f` (dix dernières lignes avec mise à jour), `--lines=n` (n dernières lignes)

Utilisation de la commande **`tail`**:

- Par défaut, afficher les dix dernières lignes du fichier:

```
user@localhost:~/Documents$ tail C60.xyz
C        -1.612000        2.332000        2.032000
C         1.184000        2.304000        2.336000
C         2.340000        2.576000       -0.216000
C         0.256000        2.776000       -2.096000
C        -2.188000        2.624000       -0.708000
C        -0.808000        3.164000        1.228000
C         0.588000        3.148000        1.380000
C         1.168000        3.284000        0.100000
C         0.124000        3.384000       -0.836000
C        -1.096000        3.308000       -0.140000
user@localhost:~/Documents$
```

- Afficher les dix dernières lignes du fichier et les mettre à jour, en utilisant: `-f`

```
user@localhost:~/Documents$ tail -f C60.xyz
C        -1.612000        2.332000        2.032000
C         1.184000        2.304000        2.336000
C         2.340000        2.576000       -0.216000
C         0.256000        2.776000       -2.096000
C        -2.188000        2.624000       -0.708000
C        -0.808000        3.164000        1.228000
C         0.588000        3.148000        1.380000
C         1.168000        3.284000        0.100000
C         0.124000        3.384000       -0.836000
C        -1.096000        3.308000       -0.140000
```

  Ceci est utile pour suivre un travail en cours: un programme effectuant des calculs est en train de produire des données dans un fichier, la commande "**`tail -f`**" permet de suivre les résultats sans avoir à ouvrir le fichier dans un éditeur et à rafraîchir son contenu. Notez qu'avec l'option `-f` l'invite reste bloquée pour mettre à jour la sortie standard avec les données, pour terminer la tâche et récupérer l'invite, appuyez sur Ctrl + c .

- Afficher les *n* dernières lignes d'un fichier, en utilisant: `--lines=`*n*

```
user@localhost:~/Documents$ tail --lines=2 C60.xyz
C         0.124000        3.384000       -0.836000
C        -1.096000        3.308000       -0.140000
user@localhost:~/Documents$
```

6.7.2.6 La commande `cut`

Pour couper, extraire et afficher des colonnes d'un fichier:

- **`cut`** *"couper, extraire des colonnes"*
  - usage: `cut [OPTION] ... [ARG]`
  - ex: `user@localhost:~$ cut -c5-10 MonFichier`
  - options: `-c` (numéros de caractères: `-cA-B,C-D,E-F...`)

La commande **`cut`** extrait des colonnes en fonction des numéros des caractères dans la ligne, pensez au fichier comme une grille de L lignes $\times$ C caractères, la commande **`cut`** permet alors d'extraire des colonnes en spécifiant des sélections de caractères de 1 à C:

- Couper des caractères 1 à 3:

```
user@localhost:~/Documents$ cut -c 1-3 Ethanol.xyz


C
C
H
H
H
H
H
H
O
user@localhost:~/Documents$
```

- Utiliser une liste séparée par des virgules pour spécifier plusieurs colonnes:

```
user@localhost:~/Documents$ cut -c 1-3,6-13,24-31,42-49 Ethanol.xyz
       9

C     1.011  -0.045  -0.062
C    -0.462   0.030   0.294
H     1.626  -0.037   0.845
H     1.325   0.803  -0.684
H     1.250  -0.961  -0.618
H    -0.758  -0.826   0.931
H    -0.682   0.953   0.866
H    -2.112   0.064  -0.664
O    -1.198   0.018  -0.907
user@localhost:~/Documents$
```

6.7.3 Options de gestion de fichiers

Pour les besoins de cette section, nous considérerons qu'il y a un autre fichier nommé `Ethanol-mod.xyz` dans le répertoire **`$HOME`**`/Documents`:

```
user@localhost:~$ ls Documents
C60.xyz  Ethanol.xyz Ethanol-mod.xyz
user@localhost:~$
```

`Ethanol-mod.xyz` contient les coordonnées atomiques d'une molécule d'éthanol modifiée, avec deux atomes d'hydrogène substitués par des atomes d'azote.

6.7.3.1 La commande `chown`

Pour changer le propriétaire d'un fichier ou d'un répertoire:

- **`chown`** *"changer le propriétaire"*
  - usage: `chown [OPTION] ... [ARG1] [ARG2]`
  - ex: `user@localhost:~$ chown user.group MonFichier`
  - options: `-R` (récursif)

La commande **`chown`** permet de changer le propriétaire et/ou le groupe d'un fichier ou d'un répertoire:

```
user@localhost:~$ chown -R leroux.ipcms Documents
user@localhost:~$
```

Cela changera récursivement la propriété de tous les répertoires et fichiers situés dans `~/Documents` y compris ce répertoire.

6.7.3.2 La commande `chmod`

Pour changer les permissions d'un fichier ou d'un répertoire:

- **`chmod`** *"changer les permissions"*
  - usage: `chmod [OPTION] ... [ARG]`
  - ex: `user@localhost:~$ chmod 755 MonFichier`
  - options: `xyz` avec `x`, `y` et `z` entre 0 et 7, `-R` (récursif)

La commande **`chmod`** permet de définir les permissions d'accès, d'utilisation ou de modification d'un fichier en utilisant trois chiffres:

(**0**	=	aucune permission	=	`---`)
1	=	exécuter	=	`--x`
2	=	écrire	=	`-w-`
4	=	lire	=	`r-`

Ainsi que leurs combinaisons:

3	=	**1 + 2**	=	exécuter + lire	=	`r-x`
5	=	**1 + 4**	=	exécuter + écrire	=	`rw-`
6	=	**2 + 4**	=	écrire + read	=	`rw-`
7	=	**1 + 2 + 4**	=	exécuter + écrire + lire	=	`rwx`

Linux fait la distinction entre 3 types d'utilisateurs pour lequels il est possible d'ajuster les permissions:

- Le propriétaire du fichier.
- Le groupe d'utilisateur auquel le propriétaire appartient.
- Tous les autres utilisateurs reconnus par le système.

Les permissions pour un fichier étant données pour chaque catégorie d'utilisateur par une simple combinaison des chiffres 1, 2 et 4.

Par exemple il est possible de donner les permissions 640 au fichier `~/Documents/Ethanol.xyz`:

```
user@localhost:~$ chmod 640 Documents/Ethanol.xyz
user@localhost:~$
```

Cela donnera les permissions de lecture et d'écriture au propriétaire, et supprimera la possibilité d'exécuter le fichier. Cela donnera les permissions de lecture aux membres du groupe auquel appartient le propriétaire, et supprimera la possibilité de modifier et d'exécuter le fichier. Enfin, cela ne donnera aucune permission aux autres utilisateurs du système.

6.7.3.3 La commande `diff`

Pour afficher les différences entre deux fichiers:

- **`diff`** *"différences entre deux fichiers"*
  - usage: **`diff [OPTION] ... [ARG]`**
  - ex: **`user@localhost:~$ diff Fichier-1.dat Fichier-2.dat`**
  - options: **`--color=always`** (colorer la sortie)

La commande **`diff`** compare les fichiers ligne par ligne pour afficher les différences:

```
user@localhost:~/Documents$ cat Ethanol-mod.xyz
         9

C       1.0111998889     -0.0452918889     -0.0626048889
C      -0.4620761111      0.0306281111      0.2946991111
H       1.6265438889     -0.0376928889      0.8456121111
N       1.3252608889      0.8030881111     -0.6846978889
H       1.2501238889     -0.9611748889     -0.6188868889
N      -0.7580021111     -0.8263228889      0.9315601111
H      -0.6822251111      0.9536901111      0.8665561111
H      -2.1126961111      0.0649821111     -0.6649928889
O      -1.1981291111      0.0180941111     -0.9072448889
user@localhost:~/Documents$ diff --color=always Ethanol.xyz Ethanol-mod.xyz
6c6
< H       1.3252608889      0.8030881111     -0.6846978889
---
> N       1.3252608889      0.8030881111     -0.6846978889
8c8
< H      -0.7580021111     -0.8263228889      0.9315601111
---
> N      -0.7580021111     -0.8263228889      0.9315601111
user@localhost:~/Documents$
```

Lorsqu'une différence est trouvée, la commande **`diff`** indique d'abord où elle se situe, puis quelle est cette différence en utilisant une combinaison de "`ligne-1`+`lettre`+`ligne-2`":

- `ligne-1` est le numéro de la ligne où se trouve la différence dans le premier fichier.
- `lettre` est une lettre-clé qui décrit le changement dans la ligne:
  - `a` = ajouter la ligne.
  - `c` = changer la ligne.
  - `d` = supprimer la ligne.
- `ligne-2` est le numéro de la ligne où se trouve la différence dans le deuxième fichier.

Ensuite, les différences sont affichées, pour le premier fichier (ici `Ethanol.xyz`) les lignes de sortie commencent par le symbole `<`, alors que pour le deuxième fichier (ici `Ethanol-mod.xyz`) les lignes de sortie commencent par le symbole `>`.

6.7.3.4 La commande `ln`

Pour créer des liens symboliques, c'est à dire des raccourcis dans le monde Linux:

- **`ln`** *"créer un lien"*
  - usage: `ln [OPTION] ... [ARG1] [ARG2]`
  - ex: `user@localhost:~$ ln -s MonFichier MonLien`
  - options: `-s` (créer un lien symbolique)

Utilisation de la commande **`ln`**:

Pour créer un lien symbolique, il est nécessaire de fournir le chemin complet, c'est à dire l'emplacement dans le système de fichiers, de l'objet que vous souhaitez lier.

- Avec un fichier:

```
user@localhost:~$ ln -s /home/user/Documents/Ethanol.xyz ../Lien-Ethanol.xyz
user@localhost:~$ cd ..
user@localhost:~$ ls -l Link*
lrwxrwxrwx. 1 user ipcms   11 14 avril 12:48 Lien-Ethanol.xyz -> /home/user/Documents/Ethanol.xyz
user@localhost:~$
```

Le lien symbolique **`Lien-Ethanol.xyz`** agit comme un raccourci:

```
user@localhost:~$ cat Lien-etanol.xyz
         9

C       1.0111998889     -0.0452918889     -0.0626048889
C      -0.4620761111      0.0306281111      0.2946991111
H       1.6265438889     -0.0376928889      0.8456121111
H       1.3252608889      0.8030881111     -0.6846978889
H       1.2501238889     -0.9611748889     -0.6188868889
H      -0.7580021111     -0.8263228889      0.9315601111
H      -0.6822251111      0.9536901111      0.8665561111
H      -2.1126961111      0.0649821111     -0.6649928889
O      -1.1981291111      0.0180941111     -0.9072448889
user@localhost:~$
```

- Pour créer un lien symbolique avec un répertoire:

```
user@localhost:~$ ln -s /home/user/Documents Dcs
user@localhost:~$ ls -l Dc*
lrwxrwxrwx. 1 user ipcms   11 14 avril 12:49 Dcs -> /home/user/Documents
user@localhost:~$
```

Le lien symbolique **`Dcs`** agit comme un raccourci vers le répertoire `~/Documents`.

```
user@localhost:~$ cp Link-Ethanol.xyz Dcs/Ethanol-copie.xyz
user@localhost:~$ ls Documents
C60.xyz  Ethanol.xyz  Ethanol-copie.xyz  Ethanol-mod.xyz
user@localhost:~$
```

6.8 Filtres

Les filtres sont un type d'utilitaire de ligne de commande conçu pour manipuler et traiter des données texte.
Les filtres les plus courants sont **awk**, **sed** et **grep**.

- **grep** est utilisé pour rechercher et filtrer des données texte. Il recherche dans un fichier spécifié des lignes qui correspondent à un motif ou à une expression rationnelle donnée, et affiche les lignes correspondantes.
- **sed** est utilisé pour effectuer des opérations sur des données texte, telles que la recherche et le remplacement, la suppression ou la substitution de lignes, et l'ajout ou l'insertion de lignes.
- **awk** est un outil de traitement de texte polyvalent qui peut effectuer diverses tâches, telles que la sélection et la transformation de colonnes de données, le filtrage de données indésirables, et l'exécution de calculs.

Les filtres sont couramment utilisés en combinaison avec d'autres utilitaires et commandes Linux pour effectuer des tâches de traitement de données plus complexes.
Comme les autres commandes, les filtres utilisent des options, mais ils nécessitent également l'utilisation d'**expressions rationnelles** ou **regexp** (pour regular expression en Anglais).
Les expressions rationnelles, regexp, sont un ensemble de règles et de modèles utilisés pour rechercher et manipuler des données texte, elles sont très souvent spécifiques à un filtre particulier.
La forme générale de la synthaxe reste cependant la même:

```
user@localhost:~$ filter option(s) 'expression rationnelle' fichier
```

ou:

```
user@localhost:~$ filter option(s) "expression rationnelle" fichier
```

L'expression rationnelle apparaît entre guillemets, simples "'" ou doubles """, l'un ou l'autre.
Les prochaines sections présenteront quelques exemples d'expressions rationnelles, dans l'espoir que celles-ci puissent vous être utiles.

6.8.1 La commande grep

Trouver les lignes correspondantes:

- **grep** *"afficher les lignes correspondant à un motif"*
  - usage: `grep [OPTION] ... [EXPRESSION RATIONNELLE] [ARG]`
  - ex: `user@localhost:~$ grep "TOTAL ENERGY =" cpmd.out`
  - options: `-n` (afficher le numéro de ligne), `-A NUM` (afficher `NUM` lignes après le motif), `-B NUM` (afficher `NUM` lignes avant le motif), `-v` (inverser la correspondance = lignes non correspondantes)

La commande **grep** est utilisée pour trouver des ligne(s) qui contiennent des motif(s) dans des fichiers texte, ces motifs étant décrits par la regexp. Ainsi, comme pour les autres filtres, l'expression rationnelle, regexp, est la partie la plus importante des instructions données à la ligne de commande.
Voici quelques exemples d'expressions rationnelles pour le filtre **grep**:

- Pour rechercher un motif:

```
grep 'linux' Fichier
```

  Pour simplement chercher toutes les ligne(s) qui contiennent le motif `linux`

- Pour rechercher toutes les lignes qui commencent par un motif, en utilisant: `^`

```
grep '^linux' Fichier
```

  Pour chercher toutes les ligne(s) qui commencent par le motif `linux`

Voici quelques exemples d'options pour le filtre **grep**:

- Toujours colorer la sortie, en utilisant: `--color=always`

```
user@localhost:~/Documents$ grep --color=always 'C' Ethanol.xyz
C       1.0111998889     -0.0452918889     -0.0626048889
C      -0.4620761111      0.0306281111      0.2946991111
user@localhost:~/Documents$
```

  Pour chercher toutes les ligne(s) qui contiennent le motif `C`

- Afficher le numéro de ligne(s) où le motif est trouvé, en utilisant: `-n`

```
user@localhost:~/Documents$ grep --color=always -n '^O' Ethanol.xyz
11:O      -1.1981291111      0.0180941111     -0.9072448889
user@localhost:~/Documents$
```

  Pour chercher toutes les ligne(s) qui commencent par le motif `O`, et afficher le numéro de ligne(s).

- Afficher *n* lignes après chaque motif trouvé, en utilisant: **-A**

```
user@localhost:~/Documents$ grep --color=always -A 5 '^C' Ethanol.xyz
C       1.0111998889     -0.0452918889     -0.0626048889
C      -0.4620761111      0.0306281111      0.2946991111
H       1.6265438889     -0.0376928889      0.8456121111
H       1.3252608889      0.8030881111     -0.6846978889
H       1.2501238889     -0.9611748889     -0.6188868889
H      -0.7580021111     -0.8263228889      0.9315601111
H      -0.6822251111      0.9536901111      0.8665561111
user@localhost:~/Documents$
```

Pour chercher toutes les ligne(s) qui commencent par le motif `C`, et afficher 5 lignes après chaque occurrence du motif.

- Afficher *n* lignes avant chaque motif trouvé, en utilisant: **-B**

```
user@localhost:~/Documents$ grep --color=always -B 5 '^O' Ethanol.xyz
H       1.3252608889      0.8030881111     -0.6846978889
H       1.2501238889     -0.9611748889     -0.6188868889
H      -0.7580021111     -0.8263228889      0.9315601111
H      -0.6822251111      0.9536901111      0.8665561111
H      -2.1126961111      0.0649821111     -0.6649928889
O      -1.1981291111      0.0180941111     -0.9072448889
user@localhost:~/Documents$
```

Pour chercher toutes les ligne(s) qui commencent par le motif `O`, et afficher 5 lignes avant chaque occurrence du motif.

- Inverser les résultats, afficher toutes les lignes sauf celles qui contiennent le motif, en utilisant: **-v**

```
user@localhost:~/Documents$ grep --color=always -v 'H' Ethanol.xyz
         9

C       1.0111998889     -0.0452918889     -0.0626048889
C      -0.4620761111      0.0306281111      0.2946991111
O      -1.1981291111      0.0180941111     -0.9072448889
user@localhost:~/Documents$
```

Pour chercher toutes les ligne(s) qui ne contiennent pas le motif `H`

6.8.2 La commande `sed`

Filtrer et transformer du texte:

- **`sed`** *"éditeur de flux pour filtrer et transformer du texte"*
  - usage: `sed [OPTION] ... [EXPRESSION RATIONNELLE] [ARG]`
  - ex: `user@localhost:~$ sed 's/MYPATH/$HOME/g'`

`sed` est utilisée pour rechercher, rempalcer, supprimer des motif(s) dans des lignes de texte. Voici quelques exemples d'expressions rationnelles pour le filtre `sed`:

- Remplacer des motif(s), en utilisant: `'s/OLD/NEW/VAL'`
  - Remplacer tous les motif(s) sur chaque ligne(s), en utilisant: `'s/OLD/NEW/g'`

```
user@localhost:~/Documents$ sed 's/1/?/g' Ethanol.xyz
        9

C       ?.0???998889     -0.0452?8889     -0.0626048889
C      -0.462076????      0.030628????      0.294699????
H       ?.6265438889     -0.0376928889      0.8456?2????
H       ?.3252608889      0.803088????     -0.6846978889
H       ?.250?238889     -0.96??748889     -0.6?88868889
H      -0.758002????     -0.8263228889      0.93?560????
H      -0.682225????      0.953690????      0.866556????
H      -2.???696????      0.064982????     -0.6649928889
O      -?.?98?29????      0.0?8094????     -0.9072448889
user@localhost:~/Documents$
```

  - Remplacer le $n^{\text{ième}}$ motif sur chaque ligne, en utilisant: `'s/OLD/NEW/n'`

```
user@localhost:~/Documents$ sed 's/C /Fe/1' Ethanol.xyz
        9

Fe      1.0111998889     -0.0452918889     -0.0626048889
Fe     -0.4620761111      0.0306281111      0.2946991111
H       1.6265438889     -0.0376928889      0.8456121111
H       1.3252608889      0.8030881111     -0.6846978889
H       1.2501238889     -0.9611748889     -0.6188868889
H      -0.7580021111     -0.8263228889      0.9315601111
H      -0.6822251111      0.9536901111      0.8665561111
H      -2.1126961111      0.0649821111     -0.6649928889
O      -1.1981291111      0.0180941111     -0.9072448889
user@localhost:~/Documents$
```

  - Remplacer les motif(s) à partir du $n^{\text{ième}}$ sur chaque ligne: `'s/OLD/NEW/gn'`

```
user@localhost:~/Documents$ sed 's/8/ /g3' Ethanol.xyz
        9

C       1.01119988 9     -0.045291   9     -0.062604   9
C      -0.4620761111      0.0306281111      0.2946991111
H       1.62654388 9     -0.037692   9      0. 456121111
H       1.32526088 9      0. 030  1111     -0.6 4697   9
H       1.25012388 9     -0.961174   9     -0.61   6   9
H      -0.7580021111     -0.826322   9      0.9315601111
H      -0.6822251111      0.9536901111      0.8665561111
H      -2.1126961111      0.0649821111     -0.6649928  9
O      -1.1981291111      0.0180941111     -0.907244   9
user@localhost:~/Documents$
```

- Ajouter des motif(s), en utilisant: `'s/\(OLD\)/MOD\1THIS/VAL'`
  - Autour de tous les motif(s) sur chaque ligne: `'s/\(OLD\)/MOD\1THIS/g'`

```
user@localhost:~/Documents$ sed 's/\(\.\)/ \1 /g' Ethanol.xyz
         9

C       1 . 0111998889     -0 . 0452918889     -0 . 0626048889
C      -0 . 4620761111      0 . 0306281111      0 . 2946991111
H       1 . 6265438889     -0 . 0376928889      0 . 8456121111
H       1 . 3252608889      0 . 8030881111     -0 . 6846978889
H       1 . 2501238889     -0 . 9611748889     -0 . 6188868889
H      -0 . 7580021111     -0 . 8263228889      0 . 9315601111
H      -0 . 6822251111      0 . 9536901111      0 . 8665561111
H      -2 . 1126961111      0 . 0649821111     -0 . 6649928889
O      -1 . 1981291111      0 . 0180941111     -0 . 9072448889
user@localhost:~/Documents$
```

Autour du premier `.`, insérer des espaces autour de chaque `.`, notez que le symbole de point `.` est un symbole spécial et doit donc être protégé en utilisant `\`, l'écriture correcte étant alors `\.`, voir section [Sec. 6.1.1]

  - Autour du $n^{\text{ième}}$ motif sur chaque ligne: `'s/\(OLD\)/MOD\1THIS/n'`

```
user@localhost:~/Documents$ sed 's/\(0\)/\*\1\*/2' Ethanol.xyz
         9

C       1.0111998889     -*0*.0452918889     -0.0626048889
C      -0.462*0*761111      0.0306281111      0.2946991111
H       1.6265438889     -0.*0*376928889      0.8456121111
H       1.3252608889      _0_.8030881111     -0.6846978889
H       1.2501238889     -*0*.9611748889     -0.6188868889
H      -0.758*0*021111     -0.8263228889      0.9315601111
H      -0.6822251111      _0_.9536901111      0.8665561111
H      -2.1126961111      0.*0*649821111     -0.6649928889
O      -1.1981291111      0.*0*180941111     -0.9072448889
user@localhost:~/Documents$
```

Autour du deuxième caractère `0` sur chaque ligne, insérer des caractères `*`, notez que le symbole d'étoile `*` est un symbole spécial et doit donc être protégé en utilisant `\`, l'écriture correcte étant alors `\*`, voir section [Sec. 6.1.1]

  - Autour de tous les motif(s) à partir du $n^{\text{ième}}$ sur chaque ligne: `'s/\(OLD\)/MOD\1THIS/gn'`

```
user@localhost:~/Documents$ sed 's/\(0\)/ \1 /g2' Ethanol.xyz
         9

C       1.0111998889     -0.0452918889     -0.0626048889
C      -0.462 0 761111      0.0306281111      0.2946991111
H       1.6265438889     -0.0376928889      0.8456121111
H       1.3252608889      0.8 3 881111     - 0 .6846978889
H       1.2501238889     -0.9611748889     - 0 .6188868889
H      -0.758 0  0 21111     - 0 .8263228889       0 .93156 0 1111
H      -0.6822251111      0.95369 0 1111       0 .8665561111
H      -2.1126961111      0.0649821111     -0.6649928889
O      -1.1981291111      0.0180941111     -0.9072448889
user@localhost:~/Documents$
```

- Réaliser des action(s) sur des ligne(s):
  - Sur une ligne cible, en utilisant: `'VAL'`

```
user@localhost:~/Documents$ sed '3 s/C/F/1' Ethanol.xyz
        9

F       1.0111998889     -0.0452918889     -0.0626048889
C      -0.4620761111      0.0306281111      0.2946991111
H       1.6265438889     -0.0376928889      0.8456121111
H       1.3252608889      0.8030881111     -0.6846978889
H       1.2501238889     -0.9611748889     -0.6188868889
H      -0.7580021111     -0.8263228889      0.9315601111
H      -0.6822251111      0.9536901111      0.8665561111
H      -2.1126961111      0.0649821111     -0.6649928889
O      -1.1981291111      0.0180941111     -0.9072448889
user@localhost:~/Documents$
```

  - Sur la dernière ligne, en utilisant: `'$`

```
user@localhost:~/Documents$ sed '$ s/O /Cl/1' Ethanol.xyz
        9

C       1.0111998889     -0.0452918889     -0.0626048889
C      -0.4620761111      0.0306281111      0.2946991111
H       1.6265438889     -0.0376928889      0.8456121111
H       1.3252608889      0.8030881111     -0.6846978889
H       1.2501238889     -0.9611748889     -0.6188868889
H      -0.7580021111     -0.8263228889      0.9315601111
H      -0.6822251111      0.9536901111      0.8665561111
H      -2.1126961111      0.0649821111     -0.6649928889
Cl     -1.1981291111      0.0180941111     -0.9072448889
user@localhost:~/Documents$
```

  - Sur une plage de lignes, en utilisant: `'VAL,VBL'`, VBL $\in$ [VAL+1, $]

```
user@localhost:~/Documents$ sed '6,9 s/0/ /g3' Ethanol.xyz
        9

C       1.0111998889     -0.0452918889     -0.0626048889
H       1.6265438889     -0.0376928889      0.8456121111
H       1.3252608889      0.8 3 881111     - .6846978889
H       1.2501238889     -0.9611748889     - .6188868889
H      -0.7580 21111     - .8263228889       .93156 1111
H      -0.6822251111      0.95369 1111       .8665561111
H      -2.1126961111      0.0649821111     -0.6649928889
O      -1.1981291111      0.0180941111     -0.9072448889
user@localhost:~/Documents$
```

- Supprimer des ligne(s):
  - Supprimer la $n^{\text{ième}}$ ligne, en utilisant: `'VALd'`

```
user@localhost:~/Documents$ sed '3d' Ethanol.xyz
         9

C       1.0111998889     -0.0452918889     -0.0626048889
C      -0.4620761111      0.0306281111      0.2946991111
H       1.6265438889     -0.0376928889      0.8456121111
H       1.3252608889      0.8 3 881111     - .6846978889
H       1.2501238889     -0.9611748889     - .6188868889
H      -0.7580 21111     - .8263228889       .93156 1111
H      -0.6822251111      0.95369 1111       .8665561111
H      -2.1126961111      0.0649821111     -0.6649928889
O      -1.1981291111      0.0180941111     -0.9072448889
user@localhost:~/Documents$
```

    Pour supprimer la troisième ligne.

  - Pour supprimer la dernière ligne, en utilisant: `'$d'`

```
user@localhost:~/Documents$ sed '4,$d' Ethanol.xyz
         9

C       1.0111998889     -0.0452918889     -0.0626048889
user@localhost:~/Documents$
```

  - Supprimer une plage de lignes, en utilisant: `'VAL,VBLd'`, VBL $\in$ [VAL+1, $]

```
user@localhost:~/Documents$ sed '4,8d' Ethanol.xyz
         9

C       1.0111998889     -0.0452918889     -0.0626048889
H      -0.6822251111      0.9536901111      0.8665561111
H      -2.1126961111      0.0649821111     -0.6649928889
O      -1.1981291111      0.0180941111     -0.9072448889
user@localhost:~/Documents$
```

    Pour supprimer les lignes 4 à 8.

- Supprimer des ligne(s) contenat des motif(s): `'/PATTERN/d'`

```
user@localhost:~/Documents$ sed '/C/d' Ethanol.xyz
         9

H       1.6265438889     -0.0376928889      0.8456121111
H       1.3252608889      0.8030881111     -0.6846978889
H       1.2501238889     -0.9611748889     -0.6188868889
H      -0.7580021111     -0.8263228889      0.9315601111
H      -0.6822251111      0.9536901111      0.8665561111
H      -2.1126961111      0.0649821111     -0.6649928889
O      -1.1981291111      0.0180941111     -0.9072448889
user@localhost:~/Documents$
```

6.8.3 La commande awk

Trouver des motif(s) et traiter les ligne(s) correspondantes:

- **awk** *"langage de recherche et de traitement de motif"*
  - usage: `awk [OPTION] ... [EXPRESSION RATIONNELLE] [ARG]`
  - ex: `user@localhost:~$ awk '{printf $NF\n}' MyFile`

La commande **awk** est utilisée pour trouver des motifs et traiter les lignes dans lesquelles les motifs sont trouvés, la façon de sortir les résultats étant décrite par la regexp.
Voici quelques exemples d'expressions rationnelles pour le filtre **awk**:

- Afficher la ligne entière, le n^{th} mot de la ligne, le dernier mot de la ligne:
  - Afficher la ligne entière qui a été traitée, en utilisant: **$0**

```
user@localhost:~/Documents$ awk '{print $0}' Ethanol.xyz
        9

C       1.0111998889     -0.0452918889     -0.0626048889
C      -0.4620761111      0.0306281111      0.2946991111
H       1.6265438889     -0.0376928889      0.8456121111
H       1.3252608889      0.8030881111     -0.6846978889
H       1.2501238889     -0.9611748889     -0.6188868889
H      -0.7580021111     -0.8263228889      0.9315601111
H      -0.6822251111      0.9536901111      0.8665561111
H      -2.1126961111      0.0649821111     -0.6649928889
O      -1.1981291111      0.0180941111     -0.9072448889
user@localhost:~/Documents$
```

  - Afficher le dernier mot de la ligne, en utilisant: **$NF**

```
user@localhost:~/Documents$ awk '{print $NF}' Ethanol.xyz
9

-0.0626048889
0.2946991111
0.8456121111
-0.6846978889
-0.6188868889
0.9315601111
0.8665561111
-0.6649928889
-0.9072448889
user@localhost:~/Documents$
```

  - Afficher le n^{th} mot de la ligne, $n \in$ [1-NF], en utilisant: **$*n***

```
user@localhost:~/Documents$ awk '{print $1}' Ethanol.xyz
9

C
C
H
H
H
H
H
H
O
user@localhost:~/Documents$
```

- Afficher et ajouter une nouvelle ligne, ou, afficher:
  - Afficher et ajouter une nouvelle ligne, en utilisant: **`print`**

```
user@localhost:~/Documents$ awk '{print $1}' Ethanol.xyz
9

C
C
H
H
H
H
H
H
O
user@localhost:~/Documents$
```

  - Afficher uniquement, en utilisant: **`printf`**

```
user@localhost:~/Documents$ awk '{printf $1}' Ethanol.xyz
9CCHHHHHHOuser@localhost:~/Documents$
```

- Afficher plus: texte, tabulation, nouvelle ligne, ou autre, en utilisant: **`" "`**
  - Afficher un texte (entre guillemets):

```
user@localhost:~/Documents$ awk '{printf $1" "}' Ethanol.xyz
9  C C H H H H H H Ouser@localhost:~/Documents$
```

    Pour afficher les premiers mots de chaque ligne, chaque mot étant séparé par des espaces.

  - Afficher une tabulation, en utilisant: **`"\t"`**

```
user@localhost:~/Documents$ awk '{printf $1"\t"}' Ethanol.xyz
9       C       C       H       H       H       H       H       H       O       user@localhost:~/Documents$
```

    Pour afficher les premiers mots de chaque ligne, chaque mot étant séparé par des tabulations.

  - Afficher une nouvelle ligne, en utilisant: **`"\n"`**

```
user@localhost:~/Documents$ awk '{printf $1"\n"}' Ethanol.xyz
9

C
C
H
H
H
H
H
H
O
user@localhost:~/Documents$
```

    Pour afficher les premiers mots de chaque ligne séparés par des nouvelles lignes, comme **`'{print $1}'`**.

- Pour rechercher des motif(s): `'/motif/'`

```
user@localhost:~/Documents$ awk '/C/' Ethanol.xyz
C       1.0111998889     -0.0452918889     -0.0626048889
C      -0.4620761111      0.0306281111      0.2946991111
user@localhost:~/Documents$
```

Cela peut être combiné avec des commandes d'affichage:

```
user@localhost:~/Documents$ awk '/C/ {print $2}' Ethanol.xyz
1.0111998889
-0.4620761111
user@localhost:~/Documents$
```

- Compter les lignes contenant des motif(s): `'/motif/{++var} END'`

```
user@localhost:~/Documents$ awk '/C/{++count} END {print "Compteur = ", count}' Ethanol.xyz
Compteur = 2
user@localhost:~/Documents$
```

Pour afficher le nombre de lignes qui contiennent le motif `C`.

- Recherche conditionnelle, basée sur le nombre de caractères ou de mots, en utilisant:

  1) `'length($N)'`, $\mathbf{N} \in [0 - \mathrm{NF}]$ ou `'NF'`
  2) Opérateurs de test: `<`, `>`, `<=`, `>=`, `==`

  – Rechercher des lignes qui contiennent *n* caractères: `'length($0)'`

```
user@localhost:~/Documents$ awk 'length($0) < 5' Ethanol.xyz
9

user@localhost:~/Documents$
```

Pour rechercher des lignes qui contiennent moins de 5 caractères.

  – Rechercher des mots qui contiennent *n* caractères: `'length($n)'`

```
user@localhost:~/Documents$ awk 'length($2) == 12' Ethanol.xyz
C       1.0111998889     -0.0452918889     -0.0626048889
H       1.6265438889     -0.0376928889      0.8456121111
H       1.3252608889      0.8030881111     -0.6846978889
H       1.2501238889     -0.9611748889     -0.6188868889
user@localhost:~/Documents$
```

Pour rechercher des lignes où le deuxième mot contient 12 caractères.

  – Pour rechercher des lignes en fonction du nombre de mots, en utilisant: `'NF'`

```
user@localhost:~/Documents$ awk 'NF == 4' Ethanol.xyz
C       1.0111998889     -0.0452918889     -0.0626048889
C      -0.4620761111      0.0306281111      0.2946991111
H       1.6265438889     -0.0376928889      0.8456121111
H       1.3252608889      0.8030881111     -0.6846978889
H       1.2501238889     -0.9611748889     -0.6188868889
H      -0.7580021111     -0.8263228889      0.9315601111
H      -0.6822251111      0.9536901111      0.8665561111
H      -2.1126961111      0.0649821111     -0.6649928889
O      -1.1981291111      0.0180941111     -0.9072448889
user@localhost:~/Documents$
```

Pour rechercher des lignes qui contiennent 4 mots.

- Substitutions, en utilisant: **'{$N="Nouveau"; print $0}'**, **N** $\in [1 - \text{NF}]$

```
user@localhost:~/Documents$ awk 'NF == 4 {$1="X"; print $0}' Ethanol.xyz
X 1.0111998889 -0.0452918889 -0.0626048889
X -0.4620761111 0.0306281111 0.2946991111
X 1.6265438889 -0.0376928889 0.8456121111
X 1.3252608889 0.8030881111 -0.6846978889
X 1.2501238889 -0.9611748889 -0.6188868889
X -0.7580021111 -0.8263228889 0.9315601111
X -0.6822251111 0.9536901111 0.8665561111
X -2.1126961111 0.0649821111 -0.6649928889
X -1.1981291111 0.0180941111 -0.9072448889
user@localhost:~/Documents$
```

Si la ligne a 4 mots, substituer le premier mot par X, puis afficher la ligne.

Voici un exemple d'option pour le filtre **awk**:

- Spécifier les séparateurs de champs, en utilisant: **-F**

```
user@localhost:~/Documents$ awk -F "." '{print $1}' Ethanol.xyz
         9

C       1
C      -0
H       1
H       1
H       1
H      -0
H      -0
H      -2
O      -1
user@localhost:~/Documents$
```

Les espaces sont les séparateurs de champs par défaut, l'utilisation de l'option **-F** permet d'ajuster ce paramètre.
Dans l'exemple précédent, l'option **-F** est utilisée pour définir les points comme nouveaux séparateurs de champs.

6.9 Redirections

6.9.1 Envoyer un travail en arrière-plan

Travail en avant-plan vers l'arrière-plan

Pour récupérer l'accès à l'invite bloquée par la dernière commande:

- Ctrl + z , puis bg *"envoyer le travail en avant-plan vers l'arrière-plan"*
  - usage: Ctrl + z dans le terminal, puis: `user@localhost:~$ bg`
  - ex: `user@localhost:~$ bg`

En utilisant les Ctrl + z +`bg`:

```
user@localhost:~$ MonScript
```

Le script `MonScript` bloque l'invite, il devrait être nécessaire de l'arrêter avant de saisir de nouvelles commandes.

Si vous appuyez sur Ctrl + z , alors le `MonScript` sera gelé, et vous obtiendrez l'accès à l'invite, puis entrez la commande `bg`:

```
user@localhost:~$
user@localhost:~$ bg
```

Travail directement en arrière-plan

Pour envoyer directement un travail en arrière-plan, afin que l'invite reste accessible:

- `&` *"envoyer un travail en arrière-plan"*
  - usage: `[CMD] [OPTION] ... [ARG] &`
  - ex: `user@localhost:~$ gedit &`

En utilisant la redirection `&`:

```
user@localhost:~$ MyScript &
user@localhost:~$
```

6.9.2 Plusieurs commandes sur la même ligne

Pour saisir plusieurs commandes sur la même ligne, utilisez les symboles **&&**:

- **&&** *"plusieurs commandes sur la même ligne (une après l'autre)"*
  - usage: `[CMD] [OPTION] ... [ARG] && [CMD] [OPTION] ... [ARG]`
  - ex: `user@localhost:~$ ls -l && cd`

En utilisant **&&** les commandes sont exécutées dans leur odre d'apparition sur la ligne:

```
user@localhost:~$ cd Documents && mkdir test && ls && cd
test
user@localhost:~$
```

6.9.3 Redirection de la sortie standard dans un fichier

Redirection dans un nouveau fichier

Pour rediriger la sortie d'une commande dans un nouveau fichier (efface l'existant):

- **>** *"rediriger la sortie standard dans un fichier (efface l'existant)"*
  - usage: `[CMD] [OPTION] ... [ARG] > MonFichier`
  - ex: `user@localhost:~$ ls -l > MonFichier`

En utilisant la redirection **>**:

```
user@localhost:~$ cat ~/Documents/Ethanol.xyz > cat-ethanol.xyz
user@localhost:~$ ls cat-*
cat-ethanol.xyz
user@localhost:~$ cat cat-ethanol.xyz
         9

C       1.0111998889     -0.0452918889     -0.0626048889
C      -0.4620761111      0.0306281111      0.2946991111
H       1.6265438889     -0.0376928889      0.8456121111
H       1.3252608889      0.8030881111     -0.6846978889
H       1.2501238889     -0.9611748889     -0.6188868889
H      -0.7580021111     -0.8263228889      0.9315601111
H      -0.6822251111      0.9536901111      0.8665561111
H      -2.1126961111      0.0649821111     -0.6649928889
O      -1.1981291111      0.0180941111     -0.9072448889
user@localhost:~$
```

Redirection dans un fichier existant

Pour ajouter la sortie d'une commande à la fin d'un fichier:

- **>>** *"rediriger la sortie standard et ajouter à la fin du fichier"*
  - usage: `[CMD] [OPTION] ... [ARG] >> MonFichier`
  - ex: `user@localhost:~$ ls -l >> MonFichier`

En utilisant la redirection >>:

```
user@localhost:~$ cat ~/Documents/Ethanol.xyz >> cat-ethanol.xyz
user@localhost:~$ ls cat-*
cat-ethanol.xyz
user@localhost:~$ cat cat-ethanol.xyz
         9

C       1.0111998889     -0.0452918889     -0.0626048889
C      -0.4620761111      0.0306281111      0.2946991111
H       1.6265438889     -0.0376928889      0.8456121111
H       1.3252608889      0.8030881111     -0.6846978889
H       1.2501238889     -0.9611748889     -0.6188868889
H      -0.7580021111     -0.8263228889      0.9315601111
H      -0.6822251111      0.9536901111      0.8665561111
H      -2.1126961111      0.0649821111     -0.6649928889
O      -1.1981291111      0.0180941111     -0.9072448889
         9

C       1.0111998889     -0.0452918889     -0.0626048889
C      -0.4620761111      0.0306281111      0.2946991111
H       1.6265438889     -0.0376928889      0.8456121111
H       1.3252608889      0.8030881111     -0.6846978889
H       1.2501238889     -0.9611748889     -0.6188868889
H      -0.7580021111     -0.8263228889      0.9315601111
H      -0.6822251111      0.9536901111      0.8665561111
H      -2.1126961111      0.0649821111     -0.6649928889
O      -1.1981291111      0.0180941111     -0.9072448889
user@localhost:~$
```

Redirection dans un nouveau fichier, incluant les messages d'erreur

Pour rediriger la sortie et les messages d'erreur potentiels d'une commande dans un nouveau fichier (effacer l'existant):

- **`>& (ou &>)`** *"rediriger la sortie et l'erreur standard dans un fichier (efface l'existant)"*
  - usage: `[CMD] [OPTION] ... [ARG] >& MonFichier`
  - ex: `user@localhost:~$ ls -l >& MonFichier`

Redirection dans un fichier existant, incluant les messages d'erreur

Pour ajouter la sortie et les messages d'erreur potentiels d'une commande à la fin d'un fichier:

- **`&>>`** *"rediriger la sortie et l'erreur standard et ajouter à la fin du fichier"*
  - usage: `[CMD] [OPTION] ... [ARG] &>> MonFichier`
  - ex: `user@localhost:~$ ls -l &>> MonFichier`

Redirection dans un fichier et envoi en arrière-plan

Il est possible d'envoyer à la fois la commande en arrière-plan et de rediriger la sortie dans un fichier:

```
user@localhost:~$ MonScript >& result &
user@localhost:~$
```

Dans cet exemple, la sortie et les messages d'erreur de **`MonScript`** sont redirigés dans le fichier `result` et la commande est envoyée en arrière-plan afin que l'invite reste accessible.

6.9.4 Redirection vers une autre commande: le pipe

Pour rediriger la sortie d'une commande vers une autre commande:

- **| (appelé pipe ou pipeline)** *"rediriger vers une autre commande"*
  - usage: `[CMD] [OPTION] ... [ARG] | [CMD] [OPTION] ... [ARG]`
  - ex: `user@localhost:~$ ls -l | more`
  - ex: `user@localhost:~$ ls -l | grep MyMotif`
  - ex: `user@localhost:~$ ls -l | awk '{printf $NF" "}'`

Le pipe est extrêmement important pour utiliser la ligne de commande, il permet de combiner efficacement des commandes sur une seule ligne.

Utilisations de la redirection `|`:

- Redirection unique:
  - Exemple 1:

```
user@localhost:~$ cat ~/Documents/Ethanol.xyz | wc -l
11
user@localhost:~$
```

    La sortie de **cat** est redirigée dans "**wc -l**" pour compter les lignes.

  - Exemple 2:

```
user@localhost:~$ ls -l | grep 'Doc*'
drwxr-xr-x.  2 user ipcms 4096 avril 13 21:23 Documents
user@localhost:~$
```

    La sortie de "**ls -l**" est redirigée dans le filtre **grep** qui sélectionne les lignes qui contiennent le mot décrit par l'expression rationnelle, dans ce cas: **Doc***.
    Lorsque vous utilisez des pipes, les filtres travaillent sur le flux de sortie.

  - Exemple 3:

```
user@localhost:~$ ls -l | grep '^d'
drwxr-xr-x.  2 user ipcms 4096 avril 12 18:44 Bureau
drwxr-xr-x.  3 user ipcms 4096 avril 13 21:23 Documents
drwxr-xr-x.  2 user ipcms 4096 avril 12 18:44 Images
drwxr-xr-x.  2 user ipcms 4096 avril 12 18:44 Modèles
drwxr-xr-x.  2 user ipcms 4096 avril 12 18:44 Musique
drwxr-xr-x.  2 user ipcms 4096 avril 12 18:44 Public
drwx------.  3 user ipcms 4096 avril 12 18:44 snap
drwxr-xr-x.  2 user ipcms 4096 avril 12 18:44 Téléchargements
drwxr-xr-x.  2 user ipcms 4096 avril 12 18:44 Vidéos
user@localhost:~$
```

    La sortie de "**ls -l**" est redirigée dans le filtre **grep** qui sélectionne les lignes qui **^d** = qui commencent par **d**

- Redirection multiple:
  - Exemple 1:

```
user@localhost:~$ ls -l | grep '^d' | awk '{printf $NF" "}'
Bureau Documents Images Modèles Musique Public snap Téléchargements Vidéos
user@localhost:~$
```

La sortie de "`ls -l`" est redirigée dans **grep** qui sélectionne les lignes commençant par **d**, puis dans **awk** qui affiche le dernier mot de chaque ligne séparé par des espaces, créant ainsi la liste de tous les dossiers dans le répertoire actif.

  - Exemple 2:

```
user@localhost:~$ ls -l | awk '/^d/ {printf $NF" "}' | sed '/snap/d'
Bureau
Documents
Images
Modèles
Musique
Public
Téléchargements
Vidéos
user@localhost:~$
```

La sortie de la commande "`ls -l`" est redirigée dans **awk** qui sélectionne les lignes qui commencent par **d**, puis dans **sed** qui supprime le motif `snap`.

  - Exemple 3:

```
user@localhost:~$ cat ~/Documents/Ethanol.xyz | sed '1,2d' |
awk '{printf "At=\""$1"\", x=\""$2"\", y=\""$3"\", z=\""$4"\"\n"}'
At="C", x="1.0111998889", y="-0.0452918889", z="-0.0626048889"
At="C", x="-0.4620761111", y="0.0306281111", z="0.2946991111"
At="H", x="1.6265438889", y="-0.0376928889", z="0.8456121111"
At="H", x="1.3252608889", y="0.8030881111", z="-0.6846978889"
At="H", x="1.2501238889", y="-0.9611748889", z="-0.6188868889"
At="H", x="-0.7580021111", y="-0.8263228889", z="0.9315601111"
At="H", x="-0.6822251111", y="0.9536901111", z="0.8665561111"
At="H", x="-2.1126961111", y="0.0649821111", z="-0.6649928889"
At="O", x="-1.1981291111", y="0.0180941111", z="-0.9072448889"
user@localhost:~$
```

La sortie de la commande **cat** est redirigée dans le filtre **sed** qui supprime les deux premières lignes, puis dans le filtre **awk** qui imprime le contenu modifié. Notez que dans la dernière séquence, les guillemets doubles doivent être protégés pour être imprimés, en utilisant **\"**.

Il n'y a pas de limite au nombre de pipes que vous pouvez utilisier pour rediriger vos commandes.

Pipe vers la sortie standard et un fichier

Pour rediriger la sortie d'une commande vers la sortie standard et un fichier:

- **tee** *"lire la sortie standard et rediriger vers la sortie et un fichier (efface l'existant)"*
  Invocation spéciale à partir d'un pipe |
  - usage: **[CMD] [OPTION] ... [ARG] | tee [OPTION] ... [ARG]**
  - ex: **user@localhost:~$ ls -l | tee MonFichier**
  - ex: **user@localhost:~$ cpmd file.inp | tee file.out**

La commande **tee** après un pipe redirige la sortie de la commande qui précède le pipe vers la sortie standard et un fichier dont le nom est spécifié comme argument de la commande **tee**.

Utilisations de la redirection **tee**:

```
user@localhost:~$ ls -l | tee ls.out
drwxr-xr-x. 2 user ipcms 4096 avril 12 18:44 Bureau
drwxr-xr-x. 3 user ipcms 4096 avril 13 21:23 Documents
drwxr-xr-x. 2 user ipcms 4096 avril 12 18:44 Images
-rw-rw-r--  2 user ipcms 4049 avril 13 21:40 ls.out
drwxr-xr-x. 2 user ipcms 4096 avril 12 18:44 Modèles
drwxr-xr-x. 2 user ipcms 4096 avril 12 18:44 Musique
drwxr-xr-x. 2 user ipcms 4096 avril 12 18:44 Public
drwx------. 3 user ipcms 4096 avril 12 18:44 snap
drwxr-xr-x. 2 user ipcms 4096 avril 12 18:44 Téléchargements
drwxr-xr-x. 2 user ipcms 4096 avril 12 18:44 Vidéos
user@localhost:~$ cat ls.out
drwxr-xr-x. 2 user ipcms 4096 avril 12 18:44 Bureau
drwxr-xr-x. 3 user ipcms 4096 avril 13 21:23 Documents
drwxr-xr-x. 2 user ipcms 4096 avril 12 18:44 Images
-rw-rw-r--  2 user ipcms 4049 avril 13 21:40 ls.out
drwxr-xr-x. 2 user ipcms 4096 avril 12 18:44 Modèles
drwxr-xr-x. 2 user ipcms 4096 avril 12 18:44 Musique
drwxr-xr-x. 2 user ipcms 4096 avril 12 18:44 Public
drwx------. 3 user ipcms 4096 avril 12 18:44 snap
drwxr-xr-x. 2 user ipcms 4096 avril 12 18:44 Téléchargements
drwxr-xr-x. 2 user ipcms 4096 avril 12 18:44 Vidéos
user@localhost:~$
```

6.10 Scripting

Le scripting, c'est-à-dire l'écriture et l'utilisation de scripts, signifie programmer une liste de commande(s) et d'action(s) à effectuer par l'interpréteur de commandes dans un fichier. L'interpréteur de commandes pourra traiter ce fichier et les commandes qu'il contient plus tard.

Un script BASH est un fichier texte basique composé d'au moins 2 lignes, et ressemble à:

```
#!/bin/bash

# Cet exemple illustre comment dire 'Hello' en programmation BASH
echo "Bonjour"
```

La première ligne `#!/bin/bash` indique au système quel programme utiliser pour exécuter le script, ici l'interpréteur de commandes BASH ; les autres lignes décrivent les commandes à effectuer.
Les commentaires peuvent être insérés en tout lieu en utilisant le symbole `#` suivi d'un espace.

Dans BASH, comme dans tout autre langage de programmation, vous trouverez de nombreuses possibilités pour atteindre le même but. Les exemples fournis dans les sections précédentes illustrent cela facilement.

Une fois le script écrit, vous avez 2 options pour l'utiliser comme commande:

- Utiliser la commande **bash** et le nom du fichier en argument, par exemple:

```
user@localhost:~$ bash MonScript
```

- Changer les permissions du fichier script et le rendre exécutable, puis l'utiliser comme commande:

```
user@localhost:~$ chmod 755 MonScript
user@localhost:~$ ./MonScript
```

6.10.1 Numérotation sur la ligne de commande

Sur la ligne de commande, **commande(s)**, **option(s)**, **argument(s)** **redirection(s)** et **expression(s)** **rationnelles** sont séparés par des espaces vides et sont numérotés selon leur ordre d'apparition sur la ligne, en commençant par 0 pour le premier élément:

```
user@localhost:~$ cat MonFichier | grep -v 'MonMotClé'
```

Sur la ligne précédente:

Objet	Numéro
La commande **cat**	0
L'argument "**MonFichier**"	1
La redirection "**|**" (voir section [Sec. 6.9.4])	2
La commande **grep**	3
L'option "**-v**"	4
L'expression rationnelle **'MonMotClé'** (voir section [Sec. 6.8])	5

6.10.2 Option(s) et argument(s)

Il est possible de passer des option(s) et/ou des argument(s) à un script en se référant au numéro qu'ils apparaîtront sur la ligne de commande (voir section [Sec. 6.10.1]), par exemple:

```
user@localhost:~$ MonScript MonOption1 MonOption2 MonArgument
```

Structures correspondantes possibles du fichier **MonScript**:

```
#!/bin/bash

ls -$1 -$2 $3
```

ou

```
#!/bin/bash

# Déclarations de variables en utilisant VAR=VAL

# Déclarer une variable OPT1 pour stocker la 1ère option
OPT1=$1
# Déclarer une variable OPT2 pour stocker la 2ème option
OPT2=$2
# Déclarer une variable ARG pour stocker l'argument
ARG=$3

ls -$OPT1$OPT2 $ARG
```

Si **MonOption1**, **MonOption2** et **MonArgument** sont respectivement **l**, **t** et **MonFichier**, alors le script **MonScript** sera équivalent à:

```
user@localhost:~$ ls -lt MonFichier
```

6.10.3 Exemples de scripts

Exemple 1:

```
#!/bin/bash

# Créer une variable DIRS qui contient la liste, séparée par des espaces,
# des répertoires dans le répertoire actif:
DIRS=`ls -l | grep '^d' | awk '{printf $NF" "}'`
# Pour rediriger les résultats de commande(s) dans une variable
# il faut mettre l'expression entre des symboles ` `

# Puis pour afficher la liste:
echo $DIRS
```

Exemple 2:

```
#!/bin/bash

# Ce script reçoit un argument, un répertoire dans lequel aller:
cd $1

# Créer une variable DIRS avec la liste des dossiers dans ce répertoire:
DIRS=`ls -l | awk '/^d/ {printf $NF" "}'`

# Afficher chaque élément de la liste:
for DIR in $DIRS
do
  echo $DIR
done
```

Pour plus d'exemples, consultez mon tutoriel de base sur la programmation BASH: Tutoriel de base sur la programmation BASH.

6.10.4 Le fichier de configuration "~/.bashrc"

Comme déjà mentionné dans la section [sec. 6.1.1], ce fichier texte est lu par BASH chaque fois qu'un shell démarre. Ensuite, BASH exécute les commande(s) dans ~/.bashrc pour préparer, configurer, la prochaine session de ligne de commande.
Par conséquent, ~/.bashrc est comme un fichier de script qui peut être utilisé pour affiner le comportement de la ligne de commande.
Parmi les choses intéressantes à configurer:

Modifier la variable PATH

Supposons que vous souhaitez inclure un nouveau répertoire dans la variable **PATH**, afin que les commandes dans ce répertoire puissent être facilement localisées par l'interpréteur de commandes.

```
user@localhost:~$ ls bin
MonScript  MaCommande  MonApplication
user@localhost:~$
```

Le répertoire ~/bin contient 3 fichiers, et ces fichiers ont les permissions d'exécution, ce qui signifie qu'ils peuvent être utilisés comme commandes.
Cependant pour utiliser l'une des ces commandes, par exemple **MonScript**, il est nécessaire de faire soit:

```
user@localhost:~$ ./bin/MonScript
```

ou

```
user@localhost:~$ /home/user/bin/MonScript
```

ou

```
user@localhost:~$ cd bin
user@localhost:~$ ./MonScript
```

Vous pouvez simplifier cela en éditant le fichier ~/.bashrc comme suit:

```
# Créer une nouvelle variable PATH qui contient:
# - 1) le contenu préservé de la variable PATH existante "$PATH"
# - 2) en ajoutant le nouveau répertoire ~/bin à la variable PATH
PATH=$PATH:~/bin
```

Créer des alias

Les alias sont des commandes créées par l'utilisateur insérées dans le fichier `~/.bashrc`.
Les alias se comportent comme des commandes dans la variable **`PATH`** et sont donc disponibles directement une fois le shell démarré. Ils ont également la priorité sur les commandes existantes et seront donc exécutés à la place. Cette dernière fonctionnalité est particulièrement intéressante pour modifier le comportement des commandes existantes.

Pour créer un alias, utilisez la syntaxe suivante:

```
alias NOM='CE QUE FAIRE'
```

Des alias pratiques à insérer dans votre fichier `~/.bashrc`:

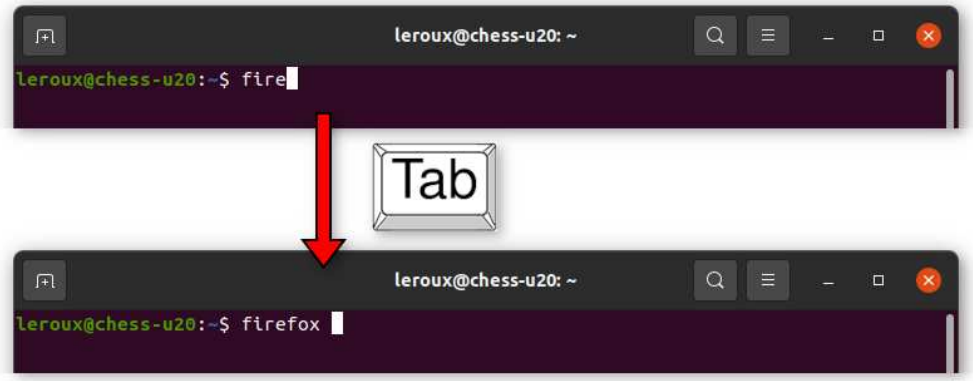
```
# Raccourcis pour les options "ls"
alias ll='ls -lh'
alias la='ls -ah'
alias lt='ls -th'
alias lla='ls -lah'
alias llt='ls -lth'
alias llta='ls -ltah'

# Confirmation systématique avant de supprimer quelque chose "rm -i"
alias rm='rm -i'

# Utiliser toujours des couleurs avec "diff"
alias diff='diff --color=always'

# Utiliser toujours des couleurs avec "grep"
alias grep='grep --color=always'
```

6.11 Conseils et astuces pour bien utiliser ligne de commande

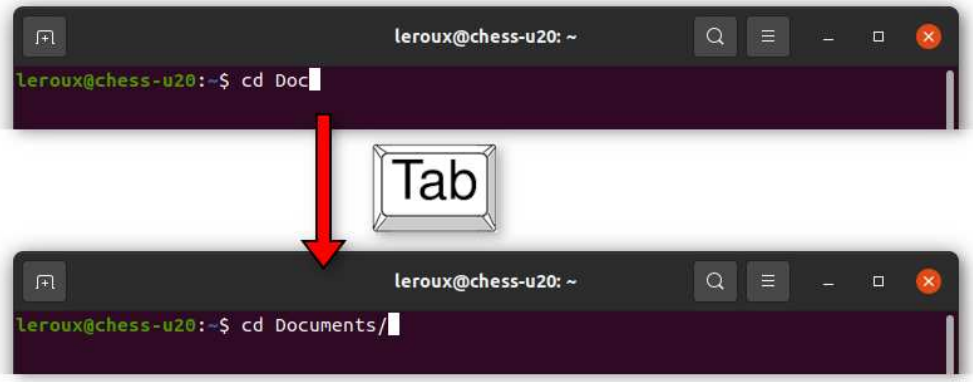
6.11.1 Auto-complétion

L'auto-complétion, ou complétion, est une astuce de ligne de commande. Lorsque vous saisissez des commandes, si le nom de la commande est partiellement saisi (écrit dans le terminal), alors si vous appuyez sur la touche ⇆ / tab du clavier, Linux essayera de compléter la ligne pour vous:

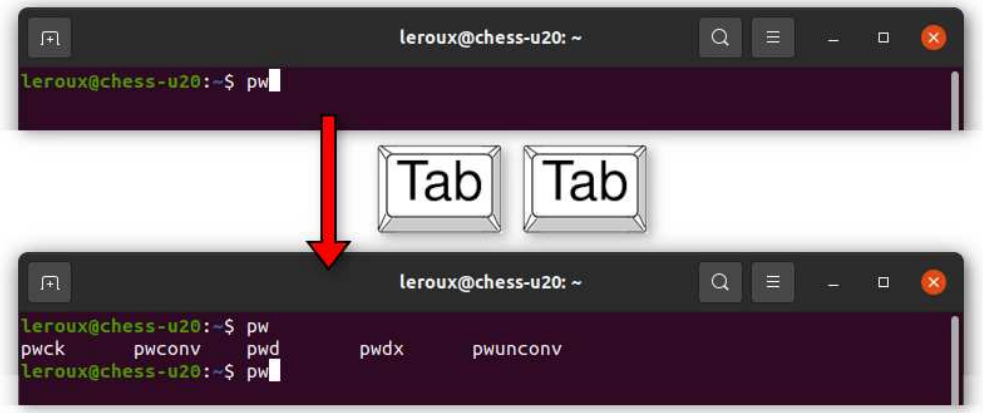
- Si il n'y a qu'une seule possibilité, le système complétera la commande / la ligne:
  - Pour une commande:



  - Pour un fichier ou un chemin:



- Sinon, si il y a plusieurs options, appuyez une seconde fois sur la touche ⟺ / tab et le système affichera les options disponibles pour compléter la commande / la ligne:
  - Pour une commande:



  - Pour un fichier ou un chemin:

6.11.2 Trucs et astuces

Vous trouverez ci-dessous quelques astuces bien utiles pour utiliser la ligne de commande:

Se déplacer dans le répertoire personnel de l'utilisateur:	`cd`
Se déplacer dans le répertoire précédent dont vous venez:	`cd -`
Se déplacer dans le répertoire parent dans l'arborescence:	`cd ..`
Se déplacer au début de la ligne:	Ctrl + a
Se déplacer à la fin de la ligne:	Ctrl + e
Tuer une tâche qui bloque la ligne de commande:	Ctrl + c
Afficher la dernière commande:	Ctrl + p
Utiliser la dernière commande saisie:	`!!`, puis Enter
Utilsier la $n^{\text{ième}}$ dernière commande saisie:	`!`$-n$, puis Enter
Commandes précédentes:	
- Voir :	Page ↑
- Naviguer :	Page ↓
- Exécuter :	Enter

Table 6.1: Trucs et astuces pour la ligne de commande.

Linux: avantages et inconvénients

7.1 Inconvénients

Aucun, le seul problème que vous pourriez rencontrer avec Linux concerne les jeux vidéo.
Vous pouvez jouer avec Linux, mais les jeux les plus récents ne seront probablement pas disponibles.

7.2 Avantages

Il y a tellement d'avantages à utiliser Linux:

- **C'est gratuit !**

  Oui, cela semble facile, mais les fabricants d'ordinateurs vendent des ordinateurs portables ou des stations de travail équipés de Linux. La différence par rapport au même équipement avec MS Windows: c'est 100 € moins cher !

- **Pas besoin de changer votre ordinateur pour utiliser la dernière version de Linux !**

  La méthode habituelle que les GAFAM (Google, Apple, Facebook, Amazon, Microsoft) utilisent pour justifier que vous changiez votre équipement: une nouvelle version de leur SE.
  Les matériaux sont de plus en plus coûteux, financièrement, mais encore plus écologiquement, alors pourquoi changer votre équipement, pourquoi consommer plus ?
  Pourquoi acheter un nouvel ordinateur, juste pour obtenir la nouvelle version de MS Windows ?
  Juste pour vous faire réfléchir, combien de temps devriez-vous utiliser votre ordinateur pour que l'empreinte carbone de son utilisation (de la production d'énergie, entre 1,8 kg CO_2/an en France et 22 kg CO_2/an en Afrique ou en Asie [2]) devienne aussi importante que l'empreinte carbone de sa production, entre 250 et 500 kg CO_2) ?

  En France au mieux: $\frac{250}{1,8} = 139$ **ans !!!**

- **Sécurité élevée: pas de virus ... pas de virus du tout !** /E
- **Stabilité élevée:** les meilleurs serveurs du monde utilisent Linux, vous utilisez Linux tous les jours sur votre téléphone portable pour vérifier votre compte en banque, pourquoi ne pas l'utiliser sur votre ordinateur ?
- **Facilité d'utilisation:** cela est particulièrement vrai avec la distribution Ubuntu préparée pour les débutants ... sans oublier que la plupart d'entre vous utilisent déjà Linux sur leur téléphone portable tous les jours ;-)
- **C'est Libre = Open Source: le code source est disponible !!!**

  C'est probablement la raison la plus importante pour laquelle vous devriez utiliser Linux, le code source est disponible. L'écosystème Linux repose principalement sur des logiciels libres, donc presque tous les codes sources sont disponibles. Cela signifie que le code peut être vérifié, lu, par n'importe quel développeur, en garantissant que le programme fait ce qu'il doit faire.
  À l'ère numérique, préférez-vous confier votre vie privée à un logiciel propriétaire (souvent propriété d'un des GAFAM) ou à un logiciel libre ?

Glossaire des commandes

8.1 Commandes standard

- **man** *"appel à l'aide - pages de manuel"*
  - usage: `man [ARG]`
  - ex: `user@localhost:~$ man ls`

- **ls** *"lister le contenu"*
  - usage: `ls [OPTION] ... [ARG]`
  - ex: `user@localhost:~$ ls`
  - options: `-l` (liste détaillée), `-t` (tri par date), `-a` (fichiers cachés), `-h` (lisible par l'humain)

- **pwd** *"imprimer le répertoire de travail"*
  - usage: `pwd`
  - ex: `user@localhost:~$ pwd`

- **cd** *"changer de répertoire"*
  - usage: `cd [ARG]`
  - ex: `user@localhost:~$ cd MonRépertoire`

- **touch** *"changer la date"*
  - usage: `touch [ARG]`
  - ex: `user@localhost:~$ touch MonFichier`

- **mv** *"déplacer"*
  - usage: `mv [ARG1] [ARG2]`
  - ex: `user@localhost:~$ mv MonFichier MonNouveauFichier`

- **cp** *"copier"*
  - usage: `cp [ARG1] [ARG2]`
  - ex: `user@localhost:~$ cp MonFichier MonNouveauFichier`

- **mkdir** *"créer un répertoire"*
  - usage: `mkdir [OPTION] ... [ARG]`
  - ex: `user@localhost:~$ mkdir MonNouveauRépertoire`
  - options: `-p` (avec parents)

- **rmdir** *"supprimer un répertoire vide"*
  - usage: `rmdir [OPTION] ... [ARG]`
  - ex: `user@localhost:~$ rmdir MonRépertoire`
  - options: `-p` (avec parents), `-r` (récurcif), `-f` (forcer)

- **rm** *"supprimer"*
  - usage: `rm [OPTION] ... [ARG]...`
  - ex: `user@localhost:~$ rm MonFichier`

- **df** *"afficher des informations sur le système de fichiers"*
  - usage: `df [OPTION] ...`
  - ex: `user@localhost:~$ df -h`
  - options: `-h` (lisible par l'humain)

- **du** *"utilisation du disque"*
  - usage: `du [OPTION] ... [ARG]...`
  - ex: `user@localhost:~$ du -sh MonFichier`
  - options: `-s` (taille totale), `-h` (lisible par l'humain)

- **cat** *"afficher dans la sortie standard"*
  - usage: `cat [ARG]`
  - ex: `user@localhost:~$ cat MonFichier`

- **tac** *"opposé de* `cat`*"*
  - usage: `tac [ARG]`
  - ex: `user@localhost:~$ tac MonFichier`

- **more** *"afficher dans la sortie standard page par page"*
  - usage: `more [ARG]`
  - ex: `user@localhost:~$ more MonFichier`

- **less** *"*`more` *avancé, permet de naviguer et de rechercher"*
  - usage: `less [ARG]`
  - ex: `user@localhost:~$ less MonFichier`

- **clear** *"nettoyer le terminal"*
  - usage: `clear`
  - ex: `user@localhost:~$ clear`

- **echo** *"afficher quelque chose"*
  - usage: `echo [ARG]`
  - ex: `user@localhost:~$ echo`

- **cut** *"couper, extraire des colonnes"*
  - usage: `cut [OPTION] ... [ARG]`
  - ex: `user@localhost:~$ cut -c5-10 MonFichier`
  - options: `-c` (numéros de caractères: `-cA-B,C-D,E-F...`)

- **tail** *"afficher la fin"*
  - usage: `tail [OPTION] ... [ARG]`
  - ex: `user@localhost:~$ tail -f MonFichier`
  - options: `-f` (dix dernières lignes avec mise à jour), `-lines=n` (n dernières lignes)

- **wc** *"compter les mots et les lignes"*
  - usage: `wc [OPTION] ... [ARG]`
  - ex: `user@localhost:~$ wc -l MonFichier`
  - options: `-l` (nombre de lignes)

- **ln** *"créer un lien"*
  - usage: `ln [OPTION] ... [ARG1] [ARG2]`
  - ex: `user@localhost:~$ ln -s MonFichier MonLien`
  - options: `-s` (créer un lien symbolique)

- **chmod** *"changer les permissions"*
  - usage: `chmod [OPTION] ... [ARG]`
  - ex: `user@localhost:~$ chmod 755 MonFichier`
  - options: `xyz` avec `x`, `y` et `z` entre 0 et 7, `-R` (récurcif)

- **chown** *"changer le propriétaire"*
  - usage: `chown [OPTION] ... [ARG1] [ARG2]`
  - ex: `user@localhost:~$ chown user.group MonFichier`
  - options: `-R` (récurcif)

- **ps** *"informations sur les processus"*
  - usage: `ps [OPTION] ...`
  - ex: `user@localhost:~$ ps -auwx`
  - options: `-ax` (tous les processus), `-u` (format orienté utilisateur), `-w` (sortie large)

- **top** *"utilisation du système"*
  - usage: `top [OPTION] ...`
  - ex: `user@localhost:~$ top`

- **kill** *"terminer un processus"*
  - usage: `kill [OPTION] ... [ARG] ...`
  - ex: `user@localhost:~$ kill -9 PID`
  - options: `-9` (envoyer un signal de fin)

- **su** *"changer d'utilisateur"*
  - usage: `su [OPTION] ... [ARG]`
  - ex: `user@localhost:~$ su - utilisateur`
  - options: `-` (utiliser la connexion shell)

- **sudo** *"exécuter une commande en tant qu'administrateur"*
  - usage: `sudo [OPTION] ... [CMD] [OPTION] ... [ARG]`
  - ex: `user@localhost:~$ sudo ls -l MonRépertoire`

- **env** *"connaître votre environnement"*
  - usage: `env`
  - ex: `user@localhost:~$ env`

- **diff** *"différences entre deux fichiers"*
  - usage: `diff [OPTION] ... [ARG]`
  - ex: `user@localhost:~$ diff Fichier-1.dat Fichier-2.dat`
  - options: `-color=always` (pour colorer la sortie)

- **sleep** *"attendre un peu s'il vous plaît"*
  - usage: `sleep [TEMPS]`
  - ex: `user@localhost:~$ sleep 3600`

- **which** *"vérifier $PATH pour savoir quel binaire est utilisé"*
  - usage: `which [ARG]`
  - ex: `user@localhost:~$ which cpmd`

- **whereis** *"trouver l'emplacement des binaires, bibliothèques et pages de manuel"*
  - usage: `whereis [ARG]`
  - ex: `user@localhost:~$ whereis gimp`

- **tar** *"créer une archive"*
  - usage: `tar [OPTION] ... [ARG1] [ARG2] ...`
  - ex: `user@localhost:~$ tar -jcf Fichier.tar.bz2 Fichier`
  - options: `-c` (créer une archive), `-x` (extraire une archive), `-f` (nom de fichier), `-j` (bzip2), `-z` (gzip), `-t` (liste de fichiers dans l'archive)

- **find** *"chercher un fichier"*
  - usage: `find [EMPLACEMENT] [OPTION] ... [ARG]`
  - ex: `user@localhost:~$ find /home -name "MonFichier"`
  - options: `-name` (nom est)

- **passwd** *"changer le mot de passe"*
  - usage: `passwd [OPTION] ... [NOM D'UTILISATEUR]`
  - ex: `user@localhost:~$ passwd`

8.2 Commandes de redirection

- **&** *"envoyer un travail en arrière-plan"*
  - usage: **[CMD] [OPTION] ... [ARG] &**
  - ex: **user@localhost:~$ gedit &**

- **Ctrl + z , puis bg** *"envoyer un travail de l'avant-plan vers arrière-plan"*
  - usage: **Ctrl + z** dans le terminal, suivi de: **user@localhost:~$ bg**
  - ex: **user@localhost:~$ bg**

- **&&** *"plusieurs commandes sur la même ligne (une après l'autre)"*
  - usage: **[CMD] [OPTION] ... [ARG] && [CMD] [OPTION] ... [ARG]**
  - ex: **user@localhost:~$ ls -l && cd**

- **>>** *"rediriger la sortie standard et ajouter à la fin du fichier"*
  - usage: **[CMD] [OPTION] ... [ARG] >> MonFichier**
  - ex: **user@localhost:~$ ls -l >> MonFichier**

- **>& (ou &>)** *"rediriger la sortie et l'erreur standard dans un fichier (efface l'existant)"*
  - usage: **[CMD] [OPTION] ... [ARG] >& MonFichier**
  - ex: **user@localhost:~$ ls -l >& MonFichier**

- **&>>** *"rediriger la sortie et l'erreur standard et ajouter à la fin du fichier"*
  - usage: **[CMD] [OPTION] ... [ARG] &>> MonFichier**
  - ex: **user@localhost:~$ ls -l &>> MonFichier**

- **| (appelé pipe ou pipeline)** *"rediriger dans une autre commande"*
  - usage: **[CMD] [OPTION] ... [ARG] | [CMD] [OPTION] ... [ARG]**
  - ex: **user@localhost:~$ ls -l | more**
  - ex: **user@localhost:~$ ls -l | grep MonMotif**
  - ex: **user@localhost:~$ ls -l | awk '{printf $NF" "}'** MonFichier

- **tee** *"lire la sortie standard et rediriger vers la sortie et un fichier (efface l'existant)"*
  Appel spécial à partir d'un pipe |
  - usage: **[CMD] [OPTION] ... [ARG] | tee [OPTION] ... [ARG]**
  - ex: **user@localhost:~$ ls -l | tee MonFichier**
  - ex: **user@localhost:~$ cpmd file.inp | tee file.out**

8.3 Commandes Bash

- **alias** *"créer un alias de commande"*
  - usage: `alias NOUVELLE_COMMANDE='[CMD] [OPTION]... [ARG]'`
  - ex: `user@localhost:~$ alias ll='ls -lth'`

- **VARIABLE_ENV=** *"créer une variable d'environnement"*
  - usage: `VARIABLE_ENV=[VALEUR]`
  - ex: `user@localhost:~$ MAVAR=100`

- **export VARIABLE_ENV=** *"créer une variable d'environnement héritable"*
  - usage: `export VARIABLE_ENV=[VALEUR]`
  - ex: `user@localhost:~$ export MAVAR=100`

8.4 Commandes de filtres

- **grep** *"imprimer les lignes correspondant à un motif"*
  - usage: `grep [OPTION] ... [EXPRESSION RATIONNELLE] [ARG]`
  - ex: `user@localhost:~$ grep "TOTAL ENERGY =" cpmd.out`
  - options: `-n` (imprimer le numéro de ligne), `-A NUM` (imprimer `NUM` lignes après le motif), `-B NUM` (imprimer `NUM` lignes avant le motif), `-v` (inverser la correspondance = lignes non correspondantes)

- **sed** *"éditeur de flux pour filtrer et transformer du texte"*
  - usage: `sed [OPTION] ... [EXPRESSION RATIONNELLE] [ARG]`
  - ex: `user@localhost:~$ sed 's/MYPATH/$HOME/g'`

- **awk** *"langage de recherche et de traitement de motifs"*
  - usage: `awk [OPTION] ... [EXPRESSION RATIONNELLE] [ARG]`
  - ex: `user@localhost:~$ awk '{printf $NF" "}' MonFichier`

Bibliography

[1] Minnesota Supercomputer Center (1993). (Cité page 79.)

[2] The Shift Project. Déployer la sobriété numérique (2020). (Cité page 115.)

Ce document a été préparé en utilisant le système d'exploitation Linux et des logiciels libres:

L'éditeur de texte	"gVim"
L'éditeur de graphiques vectoriels	Inkscape
Le programme de manipulation d'images GNU	"GIMP"
Et le système de préparation de documents	"LaTeX 2ε"