

Linux

Prof d'info

Lycée Thiers

- 1 Système d'exploitation
- 2 Présentation de Linux
- 3 Système de fichiers
 - Vue logique
 - Les principaux répertoires systèmes
- 4 Le shell bash

Avant-propos

Ces transparents constituent une toute petite introduction forcément non exhaustive au système Linux. Ils sont présentés pour faciliter le quotidien des étudiants de MP2I dans l'élaboration de leurs programmes en cours d'année. On y donne notamment

- des précisions sur les principaux répertoires sous Linux

On n'y trouve pas d'explication sur ce qu'est un disque logique ni sur la notion d'inode. Un futur document traitera des droits et redirections.

Avant-propos

Ces transparents constituent une toute petite introduction forcément non exhaustive au système Linux. Ils sont présentés pour faciliter le quotidien des étudiants de MP2I dans l'élaboration de leurs programmes en cours d'année. On y donne notamment

- des précisions sur les principaux répertoires sous Linux
- les commandes de survie dans le shell bash.

On n'y trouve pas d'explication sur ce qu'est un disque logique ni sur la notion d'inode. Un futur document traitera des droits et redirections.

- Un **historique** de Linux

- Un [historique](#) de Linux
- Un cours sur Linux très complet : [Linux France](#) et un autre : [Telecom Paris](#)

- Un [historique](#) de Linux
- Un cours sur Linux très complet : [Linux France](#) et un autre : [Telecom Paris](#)
- Des précisions sur l'architecture des fichiers sous Linux [ici](#) et la nomenclature des systèmes de fichiers [là](#).

- Un [historique](#) de Linux
- Un cours sur Linux très complet : [Linux France](#) et un autre : [Telecom Paris](#)
- Des précisions sur l'architecture des fichiers sous Linux [ici](#) et la nomenclature des systèmes de fichiers [là](#).
- Un cours du [MIT](#)

- 1 Système d'exploitation
- 2 Présentation de Linux
- 3 Système de fichiers
 - Vue logique
 - Les principaux répertoires systèmes
- 4 Le shell bash

Présentation

- Un *système d'exploitation* (en anglais operating system (OS)) est un ensemble de programmes qui dirige l'utilisation des ressources d'un ordinateur par des logiciels applicatifs.

Présentation

- Un *système d'exploitation* (en anglais operating system (OS)) est un ensemble de programmes qui dirige l'utilisation des ressources d'un ordinateur par des logiciels applicatifs.
- L'OS fournit aux programmes utilisateurs un accès unifié aux ressources matérielles et logicielles (périphériques, mémoire, autres programmes). Il offre un ensemble de fonctions « primitives » permettant d'interagir avec le matériel.



FIGURE – Le système d'exploitation est un intermédiaire entre les logiciels d'application et le matériel (WIKIPEDIA).

Composants logiciels d'un OS

On trouve notamment :

- l'*ordonnanceur* : décide quel programme s'exécute à un moment donné sur le processeur ;

Composants logiciels d'un OS

On trouve notamment :

- l'*ordonnanceur* : décide quel programme s'exécute à un moment donné sur le processeur ;
- le *gestionnaire de mémoire*, qui répartit la mémoire vive entre les différents programmes en cours d'exécution ;

Composants logiciels d'un OS

On trouve notamment :

- l'*ordonnanceur* : décide quel programme s'exécute à un moment donné sur le processeur ;
- le *gestionnaire de mémoire*, qui répartit la mémoire vive entre les différents programmes en cours d'exécution ;
- les différents *systèmes de fichiers* qui définissent la manière de stocker les fichiers sur les supports physiques (disque dur, clé USB, disque optique etc.) ;

Composants logiciels d'un OS

On trouve notamment :

- l'*ordonnanceur* : décide quel programme s'exécute à un moment donné sur le processeur ;
- le *gestionnaire de mémoire*, qui répartit la mémoire vive entre les différents programmes en cours d'exécution ;
- les différents *systèmes de fichiers* qui définissent la manière de stocker les fichiers sur les supports physiques (disque dur, clé USB, disque optique etc.) ;
- la *pile réseau* qui implémente des protocoles de communication comme TCP/IP ;

Composants logiciels d'un OS

On trouve notamment :

- l'*ordonnanceur* : décide quel programme s'exécute à un moment donné sur le processeur ;
- le *gestionnaire de mémoire*, qui répartit la mémoire vive entre les différents programmes en cours d'exécution ;
- les différents *systèmes de fichiers* qui définissent la manière de stocker les fichiers sur les supports physiques (disque dur, clé USB, disque optique etc.) ;
- la *pile réseau* qui implémente des protocoles de communication comme TCP/IP ;
- les *pilotes de périphériques* (driver) qui gèrent les périphériques matériels (souris, clavier, carte 3D etc.)

Standard POSIX

- La palette des services offerts et la manière de s'en servir diffèrent d'un système d'exploitation à l'autre.

Standard POSIX

- La palette des services offerts et la manière de s'en servir diffèrent d'un système d'exploitation à l'autre.
- Le standard industriel POSIX du IEEE définit une suite d'appels systèmes standard. POSIX est largement inspiré de UNIX.

Standard POSIX

- La palette des services offerts et la manière de s'en servir diffèrent d'un système d'exploitation à l'autre.
- Le standard industriel POSIX du IEEE définit une suite d'appels systèmes standard. POSIX est largement inspiré de UNIX.
- Un logiciel applicatif qui effectue des appels système selon POSIX pourra être utilisé sur tous les systèmes d'exploitation conformes à ce standard.

Standard POSIX

- La palette des services offerts et la manière de s'en servir diffèrent d'un système d'exploitation à l'autre.
- Le standard industriel POSIX du IEEE définit une suite d'appels systèmes standard. POSIX est largement inspiré de UNIX.
- Un logiciel applicatif qui effectue des appels système selon POSIX pourra être utilisé sur tous les systèmes d'exploitation conformes à ce standard.
- Tous les systèmes de type LINUX sont compatibles POSIX de même que Android, macOS, iOS mais pas Windows : alors que les premiers dérivent de UNIX, Windows dérive de MS-DOS dont la philosophie est différente.

Extension des noms de fichiers

- Dans les systèmes POSIX, l'extension du nom de fichier (tous les caractères à droite du « · » dans le nom du fichier) n'a pas de signification particulière pour le système.

Extension des noms de fichiers

- Dans les systèmes POSIX, l'extension du nom de fichier (tous les caractères à droite du « · » dans le nom du fichier) n'a pas de signification particulière pour le système.
- Mais elle est pratique pour l'utilisateur car il voit bien à quel type de fichier il a à faire.

Extension des noms de fichiers

- Dans les systèmes POSIX, l'extension du nom de fichier (tous les caractères à droite du « · » dans le nom du fichier) n'a pas de signification particulière pour le système.
- Mais elle est pratique pour l'utilisateur car il voit bien à quel type de fichier il a à faire.
- Un fichier exécutable n'a pas nécessairement un nom se terminant par **.exe**

Extension des noms de fichiers

- Dans les systèmes POSIX, l'extension du nom de fichier (tous les caractères à droite du « · » dans le nom du fichier) n'a pas de signification particulière pour le système.
- Mais elle est pratique pour l'utilisateur car il voit bien à quel type de fichier il a à faire.
- Un fichier exécutable n'a pas nécessairement un nom se terminant par **.exe**
- L'utilisation de l'extension pour déterminer le type de fichier est une caractéristique de Windows

- 1 Système d'exploitation
- 2 **Présentation de Linux**
- 3 Système de fichiers
 - Vue logique
 - Les principaux répertoires systèmes
- 4 Le shell bash

Historique (PI)

- Linux ou GNU/Linux : famille de systèmes d'exploitation open source de type Unix fondé sur le noyau Linux, créé en 1991 par **Linus Torvalds**.

Historique (PI)

- Linux ou GNU/Linux : famille de systèmes d'exploitation open source de type Unix fondé sur le noyau Linux, créé en 1991 par **Linus Torvalds**.
- GNU : projet informatique créé en janvier 1984 par **Richard Stallman** pour développer le système d'exploitation GNU. Chaque brique du projet est un logiciel libre utilisable en tant que tel mais dont l'objectif est de s'inscrire dans une logique cohérente. GNU/Linux c'est donc le noyau Linux plus les composantes de GNU.

Historique (PI)

- Linux ou GNU/Linux : famille de systèmes d'exploitation open source de type Unix fondé sur le noyau Linux, créé en 1991 par **Linus Torvalds**.
- GNU : projet informatique créé en janvier 1984 par **Richard Stallman** pour développer le système d'exploitation GNU. Chaque brique du projet est un logiciel libre utilisable en tant que tel mais dont l'objectif est de s'inscrire dans une logique cohérente. GNU/Linux c'est donc le noyau Linux plus les composantes de GNU.
- Linux équipe une faible part des ordinateurs PC mais beaucoup de serveurs, téléphones portables, systèmes embarqués ou encore superordinateurs.

Historique (PI)

- Linux ou GNU/Linux : famille de systèmes d'exploitation open source de type Unix fondé sur le noyau Linux, créé en 1991 par **Linus Torvalds**.
- GNU : projet informatique créé en janvier 1984 par **Richard Stallman** pour développer le système d'exploitation GNU. Chaque brique du projet est un logiciel libre utilisable en tant que tel mais dont l'objectif est de s'inscrire dans une logique cohérente. GNU/Linux c'est donc le noyau Linux plus les composantes de GNU.
- Linux équipe une faible part des ordinateurs PC mais beaucoup de serveurs, téléphones portables, systèmes embarqués ou encore superordinateurs.
- Android : système d'exploitation pour téléphones portables. Utilise le noyau Linux mais pas GNU. Equipe 85 % des tablettes tactiles et smartphones.

Logiciel libre (PI)

- Distributions Linux (Ubuntu, Debian, Linux Mint, CentOS ...) : systèmes d'exploitation libres.

Logiciel libre (PI)

- Distributions Linux (Ubuntu, Debian, Linux Mint, CentOS ...) : systèmes d'exploitation libres.
- Les 4 libertés d'un logiciels libres telles que définies par la Free Software Foundation. On peut :

le non respect de ces règles peut conduire à des condamnations.

Logiciel libre (PI)

- Distributions Linux (Ubuntu, Debian, Linux Mint, CentOS ...) : systèmes d'exploitation libres.
- Les 4 libertés d'un logiciels libres telles que définies par la Free Software Foundation. On peut :
 - utiliser le logiciel sans restriction,

le non respect de ces règles peut conduire à des condamnations.

Logiciel libre (PI)

- Distributions Linux (Ubuntu, Debian, Linux Mint, CentOS ...) : systèmes d'exploitation libres.
- Les 4 libertés d'un logiciels libres telles que définies par la Free Software Foundation. On peut :
 - utiliser le logiciel sans restriction,
 - étudier le logiciel

le non respect de ces règles peut conduire à des condamnations.

Logiciel libre (PI)

- Distributions Linux (Ubuntu, Debian, Linux Mint, CentOS ...) : systèmes d'exploitation libres.
- Les 4 libertés d'un logiciels libres telles que définies par la Free Software Foundation. On peut :
 - utiliser le logiciel sans restriction,
 - étudier le logiciel
 - le modifier pour l'adapter à ses besoins

le non respect de ces règles peut conduire à des condamnations.

Logiciel libre (PI)

- Distributions Linux (Ubuntu, Debian, Linux Mint, CentOS ...) : systèmes d'exploitation libres.
- Les 4 libertés d'un logiciels libres telles que définies par la Free Software Foundation. On peut :
 - utiliser le logiciel sans restriction,
 - étudier le logiciel
 - le modifier pour l'adapter à ses besoins
 - le redistribuer sous certaines conditions précises

le non respect de ces règles peut conduire à des condamnations.

Logiciel libre (PI)

- Distributions Linux (Ubuntu, Debian, Linux Mint, CentOS ...) : systèmes d'exploitation libres.
- Les 4 libertés d'un logiciels libres telles que définies par la Free Software Foundation. On peut :
 - utiliser le logiciel sans restriction,
 - étudier le logiciel
 - le modifier pour l'adapter à ses besoins
 - le redistribuer sous certaines conditions précises

le non respect de ces règles peut conduire à des condamnations.

- Certaines licences sont conçues selon le principe du *copyleft* : une œuvre dérivée d'un logiciel sous copyleft doit à son tour être libre.

Logiciel libre (PI)

- Distributions Linux (Ubuntu, Debian, Linux Mint, CentOS ...) : systèmes d'exploitation libres.
- Les 4 libertés d'un logiciels libres telles que définies par la Free Software Foundation. On peut :
 - utiliser le logiciel sans restriction,
 - étudier le logiciel
 - le modifier pour l'adapter à ses besoins
 - le redistribuer sous certaines conditions précises

le non respect de ces règles peut conduire à des condamnations.

- Certaines licences sont conçues selon le principe du *copyleft* : une œuvre dérivée d'un logiciel sous copyleft doit à son tour être libre.
- Licence GNU GPL : logiciel libre + copyleft. Le noyau Linux utilise cette licence.

Logiciel libre (PI)

- Distributions Linux (Ubuntu, Debian, Linux Mint, CentOS ...) : systèmes d'exploitation libres.
- Les 4 libertés d'un logiciels libres telles que définies par la Free Software Foundation. On peut :
 - utiliser le logiciel sans restriction,
 - étudier le logiciel
 - le modifier pour l'adapter à ses besoins
 - le redistribuer sous certaines conditions précises

le non respect de ces règles peut conduire à des condamnations.

- Certaines licences sont conçues selon le principe du *copyleft* : une œuvre dérivée d'un logiciel sous copyleft doit à son tour être libre.
- Licence GNU GPL : logiciel libre + copyleft. Le noyau Linux utilise cette licence.
- Un logiciel libre n'est pas nécessairement gratuit, et inversement un logiciel gratuit n'est pas forcément libre.

Caractéristiques (PI)

- Linux est *multi-tâches* : plusieurs processus peuvent s'exécuter « simultanément ».

Caractéristiques (PI)

- Linux est *multi-tâches* : plusieurs processus peuvent s'exécuter « simultanément ».
- Linux est *multi-utilisateur* : Chaque personne utilisant le système dispose d'un *compte*, qui peut être vu comme une certaine zone qui lui est allouée, accessible par un nom et un mot de passe. Un mécanisme de droits un peu contraignant empêche un utilisateur d'accéder à des données dont il n'a pas les droits.

Caractéristiques (PI)

- Linux est *multi-tâches* : plusieurs processus peuvent s'exécuter « simultanément ».
- Linux est *multi-utilisateur* : Chaque personne utilisant le système dispose d'un *compte*, qui peut être vu comme une certaine zone qui lui est allouée, accessible par un nom et un mot de passe. Un mécanisme de droits un peu contraignant empêche un utilisateur d'accéder à des données dont il n'a pas les droits.
- Les droits peuvent être modifiés.

Caractéristiques (PI)

- Linux est *multi-tâches* : plusieurs processus peuvent s'exécuter « simultanément ».
- Linux est *multi-utilisateur* : Chaque personne utilisant le système dispose d'un *compte*, qui peut être vu comme une certaine zone qui lui est allouée, accessible par un nom et un mot de passe. Un mécanisme de droits un peu contraignant empêche un utilisateur d'accéder à des données dont il n'a pas les droits.
- Les droits peuvent être modifiés.
- Un utilisateur spécial a tous les droits le *super-utilisateur* ou *administrateur* (super user en anglais).

Utilisateur

- Chaque utilisateur possède un *identifiant de connexion* (login). On lui associe un mot de passe.

Utilisateur

- Chaque utilisateur possède un *identifiant de connexion* (login). On lui associe un mot de passe.
- L'ensemble des données de l'utilisateur (identifiant, mot de passe et autres métadonnées), ainsi que ses fichiers personnels constituent le *compte* de l'utilisateur.

Utilisateur

- Chaque utilisateur possède un *identifiant de connexion* (login). On lui associe un mot de passe.
- L'ensemble des données de l'utilisateur (identifiant, mot de passe et autres métadonnées), ainsi que ses fichiers personnels constituent le *compte* de l'utilisateur.
- L'ensemble des interactions de l'utilisateur authentifié avec le système est appelé une *session*. Quand l'utilisateur se déconnecte, on dit qu'il *ferme sa session*.

Utilisateur

- A l'identifiant de connexion correspond un numéro d'utilisateur nommé UID (User IDentifier). L'OS n'utilise que l'UID, l'identifiant n'est là que par soucis de convivialité.

Utilisateur

- A l'identifiant de connexion correspond un numéro d'utilisateur nommé UID (User IDentifier). L'OS n'utilise que l'UID, l'identifiant n'est là que par soucis de convivialité.
- Les utilisateurs peuvent être réunis en *groupes*. Ces derniers ont un nom symbolique et un identifiant numérique GID (Group IDentifier).

Utilisateur

- A l'identifiant de connexion correspond un numéro d'utilisateur nommé UID (User IDentifier). L'OS n'utilise que l'UID, l'identifiant n'est là que par soucis de convivialité.
- Les utilisateurs peuvent être réunis en *groupes*. Ces derniers ont un nom symbolique et un identifiant numérique GID (Group IDentifier).
- Chaque utilisateur a un groupe principal et éventuellement des groupes secondaires. La commande **id** affiche les identifiants numériques et les groupes de l'utilisateur courant.

```
$ id
uid=1000(ivan) gid=1000(ivan) groupes=1000(ivan),
4(adm),24(cdrom),27(sudo),30(dip),46(plugdev),
111(lxd),116(lpadmin),131(libvirt),132(sambashare)
```


Noyau

- *Noyau de système d'exploitation* (ou noyau, ou kernel en anglais) : partie fondamentale de certains systèmes d'exploitation. Gère les ressources de l'ordinateur et permet aux différents composants (matériels et logiciels) de communiquer entre eux.

Noyau

- *Noyau de système d'exploitation* (ou noyau, ou kernel en anglais) : partie fondamentale de certains systèmes d'exploitation. Gère les ressources de l'ordinateur et permet aux différents composants (matériels et logiciels) de communiquer entre eux.
- Le noyau assure :

Noyau

- *Noyau de système d'exploitation* (ou noyau, ou kernel en anglais) : partie fondamentale de certains systèmes d'exploitation. Gère les ressources de l'ordinateur et permet aux différents composants (matériels et logiciels) de communiquer entre eux.
- Le noyau assure :
 - la communication entre les logiciels et le matériel

Noyau

- *Noyau de système d'exploitation* (ou noyau, ou kernel en anglais) : partie fondamentale de certains systèmes d'exploitation. Gère les ressources de l'ordinateur et permet aux différents composants (matériels et logiciels) de communiquer entre eux.
- Le noyau assure :
 - la communication entre les logiciels et le matériel
 - la gestion des divers logiciels (tâches) d'une machine (lancement des programmes, ordonnancement...)

Noyau

- *Noyau de système d'exploitation* (ou noyau, ou kernel en anglais) : partie fondamentale de certains systèmes d'exploitation. Gère les ressources de l'ordinateur et permet aux différents composants (matériels et logiciels) de communiquer entre eux.
- Le noyau assure :
 - la communication entre les logiciels et le matériel
 - la gestion des divers logiciels (tâches) d'une machine (lancement des programmes, ordonnancement...)
 - la gestion du matériel (mémoire, processeur, périphérique, stockage...).

Noyau

- *Noyau de système d'exploitation* (ou noyau, ou kernel en anglais) : partie fondamentale de certains systèmes d'exploitation. Gère les ressources de l'ordinateur et permet aux différents composants (matériels et logiciels) de communiquer entre eux.
- Le noyau assure :
 - la communication entre les logiciels et le matériel
 - la gestion des divers logiciels (tâches) d'une machine (lancement des programmes, ordonnancement...)
 - la gestion du matériel (mémoire, processeur, périphérique, stockage...).
- Le noyau offre ses fonctions (l'accès aux ressources qu'il gère) au travers des *appels système*. Il transmet ou interprète les informations du matériel via des interruptions (appelées les entrées/sorties).

Processus

Processus : programme en cours d'exécution par un ordinateur. Régulièrement, il a besoin d'accéder à des ressources protégées (comme une écriture en mémoire). Le noyau prend alors le relai du processus pour rendre le service demandé et lui rend le contrôle lorsque les actions voulues ont été réalisées.

- 1 Système d'exploitation
- 2 Présentation de Linux
- 3 Système de fichiers**
 - Vue logique
 - Les principaux répertoires systèmes
- 4 Le shell bash

- 1 Système d'exploitation
- 2 Présentation de Linux
- 3 Système de fichiers**
 - Vue logique
 - Les principaux répertoires systèmes
- 4 Le shell bash

Système de fichier

- *Système de fichier* : ensemble de conventions et structures permettant de stocker des données sur un support physique ♡

Système de fichier

- *Système de fichier* : ensemble de conventions et structures permettant de stocker des données sur un support physique ♥
- Chaque système de fichier implémente (à sa façon) les primitives de manipulation de fichiers génériques offertes par le système d'exploitation :

Système de fichier

- *Système de fichier* : ensemble de conventions et structures permettant de stocker des données sur un support physique ♥
- Chaque système de fichier implémente (à sa façon) les primitives de manipulation de fichiers génériques offertes par le système d'exploitation :
 - Création d'un fichier de nom donné ;

Système de fichier

- *Système de fichier* : ensemble de conventions et structures permettant de stocker des données sur un support physique ♥
- Chaque système de fichier implémente (à sa façon) les primitives de manipulation de fichiers génériques offertes par le système d'exploitation :
 - Création d'un fichier de nom donné ;
 - Ouverture d'un fichier en lecture ou écriture ;

Système de fichier

- *Système de fichier* : ensemble de conventions et structures permettant de stocker des données sur un support physique ♥
- Chaque système de fichier implémente (à sa façon) les primitives de manipulation de fichiers génériques offertes par le système d'exploitation :
 - Création d'un fichier de nom donné ;
 - Ouverture d'un fichier en lecture ou écriture ;
 - Lecture ou écriture de portion d'un fichier ;

Système de fichier

- *Système de fichier* : ensemble de conventions et structures permettant de stocker des données sur un support physique ♥
- Chaque système de fichier implémente (à sa façon) les primitives de manipulation de fichiers génériques offertes par le système d'exploitation :
 - Création d'un fichier de nom donné ;
 - Ouverture d'un fichier en lecture ou écriture ;
 - Lecture ou écriture de portion d'un fichier ;
 - Suppression d'un fichier ;

Système de fichier

- *Système de fichier* : ensemble de conventions et structures permettant de stocker des données sur un support physique ♥
- Chaque système de fichier implémente (à sa façon) les primitives de manipulation de fichiers génériques offertes par le système d'exploitation :
 - Création d'un fichier de nom donné ;
 - Ouverture d'un fichier en lecture ou écriture ;
 - Lecture ou écriture de portion d'un fichier ;
 - Suppression d'un fichier ;
 - Copie ou renommage d'un fichier.

Systèmes de fichier

- Chaque système de fichier possède ses propres caractéristiques et objectifs. Certains (Samba ou NFS) sont par exemple destinés à exposer à l'utilisateur des fichiers et répertoires se trouvant physiquement sur une/des machine(s) distante(s).

Systèmes de fichier

- Chaque système de fichier possède ses propres caractéristiques et objectifs. Certains (Samba ou NFS) sont par exemple destinés à exposer à l'utilisateur des fichiers et répertoires se trouvant physiquement sur une/des machine(s) distante(s).
- Quelques exemples :

OS	Système de fichier
MSDOS	FAT
Windows 95/98	FAT32 (File Allocation Table 32bits)
Windows NT	NTFS (New Technology File System) standard pour
MAC OS	HFS+ (High Performance File System)
Linux	ext4 (Extended File System)

Système de fichier

- Fichier : suite d'octets constituant un ensemble cohérent (en principe) d'information. Sous Linux, *tout* est fichier (même le clavier est considéré comme un fichier).

Système de fichier

- Fichier : suite d'octets constituant un ensemble cohérent (en principe) d'information. Sous Linux, *tout* est fichier (même le clavier est considéré comme un fichier).
- Règles de nommage Linux : 255 caractères au plus, principalement des lettres et des chiffres mais `.` `-` `_` `~` `+` `%` sont aussi autorisés.

Système de fichier

- Fichier : suite d'octets constituant un ensemble cohérent (en principe) d'information. Sous Linux, *tout* est fichier (même le clavier est considéré comme un fichier).
- Règles de nommage Linux : 255 caractères au plus, principalement des lettres et des chiffres mais `.` `-` `_` `~` `+` `%` sont aussi autorisés.
- Les espaces ne sont pas interdits, mais je les déconseille de même que les accents.

Répertoires ou dossiers

- Si le système de fichier était une commode, les répertoires seraient des tiroirs contenant des fichiers ou des sous-tiroirs.

Répertoires ou dossiers

- Si le système de fichier était une commode, les répertoires seraient des tiroirs contenant des fichiers ou des sous-tiroirs.
- Un *répertoire* est un fichier dans lequel se trouve la liste de son « contenu » sous la forme **(nom du fichier, numéro d'inode)** ♥

Répertoires ou dossiers

- Si le système de fichier était une commode, les répertoires seraient des tiroirs contenant des fichiers ou des sous-tiroirs.
- Un *répertoire* est un fichier dans lequel se trouve la liste de son « contenu » sous la forme **(nom du fichier, numéro d'inode)** ♥
- Dans un répertoire on trouve deux fichiers particuliers notés « `.` » (répertoire courant) et « `..` » (répertoire parent). ♥

Répertoires ou dossiers

- Si le système de fichier était une commode, les répertoires seraient des tiroirs contenant des fichiers ou des sous-tiroirs.
- Un *répertoire* est un fichier dans lequel se trouve la liste de son « contenu » sous la forme **(nom du fichier, numéro d'inode)** ♡
- Dans un répertoire on trouve deux fichiers particuliers notés « `.` » (répertoire courant) et « `..` » (répertoire parent). ♡
- Le système de fichier est représenté par un arbre dont la racine est le répertoire *root* noté « `/` » ♡.

Répertoires ou dossiers

- Si le système de fichier était une commode, les répertoires seraient des tiroirs contenant des fichiers ou des sous-tiroirs.
- Un *répertoire* est un fichier dans lequel se trouve la liste de son « contenu » sous la forme **(nom du fichier, numéro d'inode)** ♥
- Dans un répertoire on trouve deux fichiers particuliers notés « `.` » (répertoire courant) et « `..` » (répertoire parent). ♥
- Le système de fichier est représenté par un arbre dont la racine est le répertoire *root* noté « `/` » ♥.
- Pratique personnelle de nommage pour les répertoires et fichiers persos : la première lettre des noms de répertoires en majuscules, celle des autres fichiers en minuscule : **UnNomDeRepertoire**, **unNomDeFichier**.

Exemple d'arborescence de fichiers

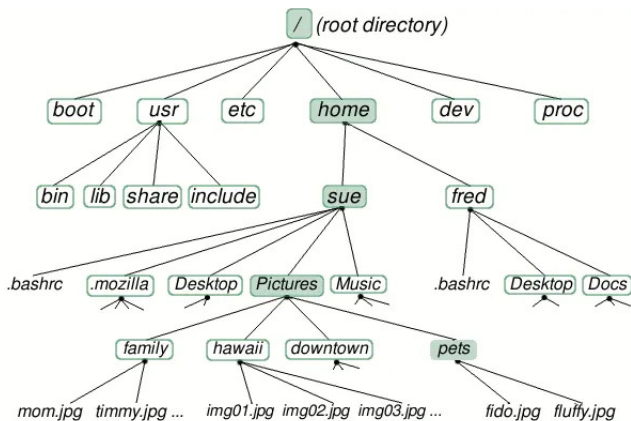
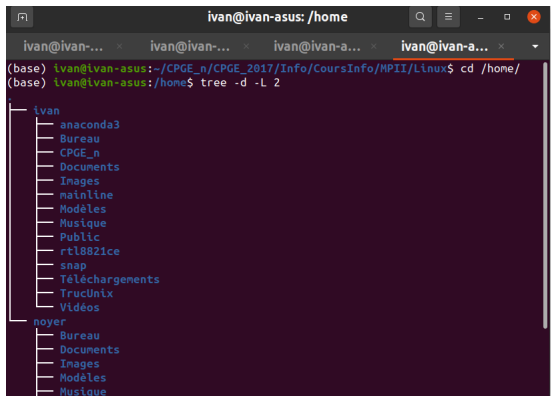


FIGURE – Le système de fichiers Linux (d'après [malekal](#))

Exemple personnel



```
ivan@ivan-asus: /home
ivan@ivan-... x ivan@ivan-... x ivan@ivan-a... x ivan@ivan-a... x
(base) ivan@ivan-asus:~/CPGE_n/CPGE_2017/Info/CoursInfo/MPII/Linux$ cd /home/
(base) ivan@ivan-asus:/home$ tree -d -L 2
ivan
├── anaconda3
├── Bureau
├── CPGE_n
├── Documents
├── Images
├── mainline
├── Modèles
├── Musique
├── Public
├── rtl8821ce
├── snap
├── Téléchargements
├── TrucUnix
├── Vidéos
└── noyer
    ├── Bureau
    ├── Documents
    ├── Images
    ├── Modèles
    └── Musique
```

FIGURE – Une image de mon répertoire **home** donnée par la commande **tree**

L'option **-d** pour ne lister que les répertoires, l'option **-L** pour indiquer la profondeur de récursion.

- 1 Système d'exploitation
- 2 Présentation de Linux
- 3 Système de fichiers**
 - Vue logique
 - Les principaux répertoires systèmes
- 4 Le shell bash

Principaux répertoires

/ C'est la racine de l'arborescence ♡

Principaux répertoires

- / C'est la racine de l'arborescence ♥
- /bin stocke les exécutables et binaires essentiels lors du démarrage. Commandes utilisables ensuite par les utilisateurs (exemple **ping**). ♥

Principaux répertoires

- / C'est la racine de l'arborescence ♡
- /bin stocke les exécutables et binaires essentiels lors du démarrage. Commandes utilisables ensuite par les utilisateurs (exemple **ping**). ♡
- /sbin stocke les exécutables et binaires essentiels lors du démarrage mais réservées au *superuser* (administrateur système) (exemple **reboot**). ♡

Principaux répertoires

- / C'est la racine de l'arborescence ♡
- /bin stocke les exécutable et binaires essentiels lors du démarrage. Commandes utilisables ensuite par les utilisateurs (exemple **ping**). ♡
- /sbin stocke les exécutable et binaires essentiels lors du démarrage mais réservées au *superuser* (administrateur système) (exemple **reboot**). ♡
- /boot stocke les fichiers de démarrage Linux ♡

Principaux répertoires

- / C'est la racine de l'arborescence ♡
- /bin stocke les exécutable et binaires essentiels lors du démarrage. Commandes utilisables ensuite par les utilisateurs (exemple **ping**). ♡
- /sbin stocke les exécutable et binaires essentiels lors du démarrage mais réservées au *superuser* (administrateur système) (exemple **reboot**). ♡
- /boot stocke les fichiers de démarrage Linux ♡
- /dev (pour *devices*) les fichiers liés aux périphériques.

Principaux répertoires

- / C'est la racine de l'arborescence ♡
- /bin stocke les exécutable et binaires essentiels lors du démarrage. Commandes utilisables ensuite par les utilisateurs (exemple **ping**). ♡
- /sbin stocke les exécutable et binaires essentiels lors du démarrage mais réservées au *superuser* (administrateur système) (exemple **reboot**). ♡
- /boot stocke les fichiers de démarrage Linux ♡
- /dev (pour *devices*) les fichiers liés aux périphériques.
- /etc Les fichiers de configuration de Linux et des applications. ♡

Principaux répertoires

- / C'est la racine de l'arborescence ♥
- /bin stocke les exécutable et binaires essentiels lors du démarrage. Commandes utilisables ensuite par les utilisateurs (exemple **ping**). ♥
- /sbin stocke les exécutable et binaires essentiels lors du démarrage mais réservées au *superuser* (administrateur système) (exemple **reboot**). ♥
- /boot stocke les fichiers de démarrage Linux ♥
- /dev (pour *devices*) les fichiers liés aux périphériques.
- /etc Les fichiers de configuration de Linux et des applications. ♥
- /home comptes des utilisateurs. ♥

Principaux répertoires

- / C'est la racine de l'arborescence ♥
- /bin stocke les exécutable et binaires essentiels lors du démarrage. Commandes utilisables ensuite par les utilisateurs (exemple **ping**). ♥
- /sbin stocke les exécutable et binaires essentiels lors du démarrage mais réservées au *superuser* (administrateur système) (exemple **reboot**). ♥
- /boot stocke les fichiers de démarrage Linux ♥
- /dev (pour *devices*) les fichiers liés aux périphériques.
- /etc Les fichiers de configuration de Linux et des applications. ♥
- /home comptes des utilisateurs. ♥
 - /lib Les librairies et bibliothèques partagés pour le fonctionnement de l'OS et des applications

Les principaux répertoires

`/usr` répertoire des applications partagées par différentes machines ou utilisateurs. Les programmes dans `/usr` ne sont pas nécessaires au démarrage. ♡

Les principaux répertoires

`/usr` répertoire des applications partagées par différentes machines ou utilisateurs. Les programmes dans `/usr` ne sont pas nécessaires au démarrage. ♡

`/media` points de montages des médias amovibles (clés USB...) ♡

Les principaux répertoires

- `/usr` répertoire des applications partagées par différentes machines ou utilisateurs. Les programmes dans `/usr` ne sont pas nécessaires au démarrage. ♡
- `/media` points de montages des médias amovibles (clés USB...) ♡
- `/proc` Répertoire virtuel avec les informations système (l'état du système, noyau Linux, etc) basé sur **proafs** (process file system)

Les principaux répertoires

- `/usr` répertoire des applications partagées par différentes machines ou utilisateurs. Les programmes dans `/usr` ne sont pas nécessaires au démarrage. ♡
- `/media` points de montages des médias amovibles (clés USB...) ♡
- `/proc` Répertoire virtuel avec les informations système (l'état du système, noyau Linux, etc) basé sur **proafs** (process file system)
- `/root` dossier personnel de l'utilisateur *root*

Les principaux répertoires

- `/usr` répertoire des applications partagées par différentes machines ou utilisateurs. Les programmes dans `/usr` ne sont pas nécessaires au démarrage. ♡
- `/media` points de montages des médias amovibles (clés USB...) ♡
- `/proc` Répertoire virtuel avec les informations système (l'état du système, noyau Linux, etc) basé sur **proafs** (process file system)
- `/root` dossier personnel de l'utilisateur *root*
- `/var` Contient des données mises à jour par différents programmes durant le fonctionnement du système (fichiers journaux (log), des fichiers de données (spool) ou des fichiers de blocage (lock)) ♡

Les principaux répertoires

- `/usr` répertoire des applications partagées par différentes machines ou utilisateurs. Les programmes dans `/usr` ne sont pas nécessaires au démarrage. ♡
- `/media` points de montages des médias amovibles (clés USB...) ♡
- `/proc` Répertoire virtuel avec les informations système (l'état du système, noyau Linux, etc) basé sur **proafs** (process file system)
- `/root` dossier personnel de l'utilisateur *root*
- `/var` Contient des données mises à jour par différents programmes durant le fonctionnement du système (fichiers journaux (log), des fichiers de données (spool) ou des fichiers de blocage (lock)) ♡
- `/tmp` les fichiers temporaires. ♡

le répertoire `/usr`

Données des applications des utilisateurs.

`/usr/bin` Commandes utilisables par tous les utilisateurs, et non nécessaires lors du démarrage du système.

le répertoire `/usr`

Données des applications des utilisateurs.

- `/usr/bin` Commandes utilisables par tous les utilisateurs, et non nécessaires lors du démarrage du système.
- `/usr/sbin` Commandes réservées au super-utilisateur , et non nécessaires lors du démarrage du système.

le répertoire `/usr`

Données des applications des utilisateurs.

- `/usr/bin` Commandes utilisables par tous les utilisateurs, et non nécessaires lors du démarrage du système.
- `/usr/sbin` Commandes réservées au super-utilisateur , et non nécessaires lors du démarrage du système.
- `/usr/lib` le dossier des librairies utilisées par les applications

le répertoire `/usr`

Données des applications des utilisateurs.

`/usr/bin` Commandes utilisables par tous les utilisateurs, et non nécessaires lors du démarrage du système.

`/usr/sbin` Commandes réservées au super-utilisateur , et non nécessaires lors du démarrage du système.

`/usr/lib` le dossier des bibliothèques utilisées par les applications

`/usr/local` applications installées localement par l'administrateur système.

le répertoire `/usr`

Données des applications des utilisateurs.

`/usr/bin` Commandes utilisables par tous les utilisateurs, et non nécessaires lors du démarrage du système.

`/usr/sbin` Commandes réservées au super-utilisateur , et non nécessaires lors du démarrage du système.

`/usr/lib` le dossier des bibliothèques utilisées par les applications

`/usr/local` applications installées localement par l'administrateur système.

`/usr/src` Les sources des applications que l'on peut compiler

le répertoire `/usr`

Données des applications des utilisateurs.

`/usr/bin` Commandes utilisables par tous les utilisateurs, et non nécessaires lors du démarrage du système.

`/usr/sbin` Commandes réservées au super-utilisateur , et non nécessaires lors du démarrage du système.

`/usr/lib` le dossier des bibliothèques utilisées par les applications

`/usr/local` applications installées localement par l'administrateur système.

`/usr/src` Les sources des applications que l'on peut compiler

`/usr/share` le dossier avec les fichiers qui peuvent être partagés avec toutes les architectures (i386 -INTEL-, amd64 -AMD-, etc).

le répertoire `/usr`

Données des applications des utilisateurs.

`/usr/bin` Commandes utilisables par tous les utilisateurs, et non nécessaires lors du démarrage du système.

`/usr/sbin` Commandes réservées au super-utilisateur, et non nécessaires lors du démarrage du système.

`/usr/lib` le dossier des bibliothèques utilisées par les applications

`/usr/local` applications installées localement par l'administrateur système.

`/usr/src` Les sources des applications que l'on peut compiler

`/usr/share` le dossier avec les fichiers qui peuvent être partagés avec toutes les architectures (i386 -INTEL-, amd64 -AMD-, etc).

`/usr/local/bin` c'est ici que peuvent être placés les programmes persos à partager avec d'autres utilisateurs (il faut quand même demander les droits au superuser).

le répertoire `/usr`

Données des applications des utilisateurs.

`/usr/bin` Commandes utilisables par tous les utilisateurs, et non nécessaires lors du démarrage du système.

`/usr/sbin` Commandes réservées au super-utilisateur, et non nécessaires lors du démarrage du système.

`/usr/lib` le dossier des bibliothèques utilisées par les applications

`/usr/local` applications installées localement par l'administrateur système.

`/usr/src` Les sources des applications que l'on peut compiler

`/usr/share` le dossier avec les fichiers qui peuvent être partagés avec toutes les architectures (i386 -INTEL-, amd64 -AMD-, etc).

`/usr/local/bin` c'est ici que peuvent être placés les programmes persos à partager avec d'autres utilisateurs (il faut quand même demander les droits au superuser).

le répertoire `/var`

`/var` Le répertoire `/usr/` étant en lecture seule, tous les programmes qui ont besoin d'écrire des fichiers journaux (log), des fichiers de données (spool) ou des fichiers de blocage (lock) devraient les écrire dans `/var`.

le répertoire `/var`

`/var` Le répertoire `/usr/` étant en lecture seule, tous les programmes qui ont besoin d'écrire des fichiers journaux (log), des fichiers de données (spool) ou des fichiers de blocage (lock) devraient les écrire dans `/var`.

`/var/lock` Fichiers de blocage, pour interdire par exemple deux utilisations simultanées d'un périphérique.

le répertoire `/var`

`/var` Le répertoire `/usr/` étant en lecture seule, tous les programmes qui ont besoin d'écrire des fichiers journaux (log), des fichiers de données (spool) ou des fichiers de blocage (lock) devraient les écrire dans `/var`.

`/var/lock` Fichiers de blocage, pour interdire par exemple deux utilisations simultanées d'un périphérique.

`/var/run` Des fichiers liés aux applications en cours de fonctionnement. Par exemple, on peut y trouver le PID de l'application.

le répertoire `/var`

`/var` Le répertoire `/usr/` étant en lecture seule, tous les programmes qui ont besoin d'écrire des fichiers journaux (log), des fichiers de données (spool) ou des fichiers de blocage (lock) devraient les écrire dans `/var`.

`/var/lock` Fichiers de blocage, pour interdire par exemple deux utilisations simultanées d'un périphérique.

`/var/run` Des fichiers liés aux applications en cours de fonctionnement. Par exemple, on peut y trouver le PID de l'application.

`/var/log` les journaux et logs du système et des applications

le répertoire `/var`

`/var` Le répertoire `/usr/` étant en lecture seule, tous les programmes qui ont besoin d'écrire des fichiers journaux (log), des fichiers de données (spool) ou des fichiers de blocage (lock) devraient les écrire dans `/var`.

`/var/lock` Fichiers de blocage, pour interdire par exemple deux utilisations simultanées d'un périphérique.

`/var/run` Des fichiers liés aux applications en cours de fonctionnement. Par exemple, on peut y trouver le PID de l'application.

`/var/log` les journaux et logs du système et des applications

`/var/cache` dossiers et fichiers de cache. Par exemple apt peut y stocker les packages pour installer ou mettre à jour le système et les applications.

le répertoire `/var`

`/var` Le répertoire `/usr/` étant en lecture seule, tous les programmes qui ont besoin d'écrire des fichiers journaux (log), des fichiers de données (spool) ou des fichiers de blocage (lock) devraient les écrire dans `/var`.

`/var/lock` Fichiers de blocage, pour interdire par exemple deux utilisations simultanées d'un périphérique.

`/var/run` Des fichiers liés aux applications en cours de fonctionnement. Par exemple, on peut y trouver le PID de l'application.

`/var/log` les journaux et logs du système et des applications

`/var/cache` dossiers et fichiers de cache. Par exemple apt peut y stocker les packages pour installer ou mettre à jour le système et les applications.

`/var/spool` Pour stocker les fichiers de données des programmes.

le répertoire `/etc`

stocke les fichiers de configurations du système ainsi que des applications.
Un sous-répertoire par application

`/etc/init.d` et `/etc/default` : les fichiers liés aux daemons Linux

le répertoire `/etc`

stocke les fichiers de configurations du système ainsi que des applications.

Un sous-répertoire par application

`/etc/init.d` et `/etc/default` : les fichiers liés aux daemons Linux

`/etc/password`, `/etc/group`, `/etc/shadow` : les fichiers de configuration des utilisateurs Linux.

le répertoire /etc

stocke les fichiers de configurations du système ainsi que des applications.

Un sous-répertoire par application

`/etc/init.d` et `/etc/default` : les fichiers liés aux daemons Linux

`/etc/password`, `/etc/group`, `/etc/shadow` : les fichiers de configuration des utilisateurs Linux.

`/etc/hosts` : le fichier HOSTS de Linux (liens entre adresses IP et littérales)

le répertoire /etc

stocke les fichiers de configurations du système ainsi que des applications.

Un sous-répertoire par application

`/etc/init.d` et `/etc/default` : les fichiers liés aux daemons Linux

`/etc/password`, `/etc/group`, `/etc/shadow` : les fichiers de configuration des utilisateurs Linux.

`/etc/hosts` : le fichier HOSTS de Linux (liens entre adresses IP et littérales)

`/etc/sudoers` et `/etc/sudoers.d` : la configuration de sudo.

le répertoire `/etc`

stocke les fichiers de configurations du système ainsi que des applications.
Un sous-répertoire par application

`/etc/init.d` et `/etc/default` : les fichiers liés aux daemons Linux

`/etc/password`, `/etc/group`, `/etc/shadow` : les fichiers de configuration des utilisateurs Linux.

`/etc/hosts` : le fichier HOSTS de Linux (liens entre adresses IP et littérales)

`/etc/sudoers` et `/etc/sudoers.d` : la configuration de sudo.

`/etc/sysctl.conf` et `/etc/sysctl.d` les fichiers de configuration de démarrage du noyau Linux.

le répertoire /etc

stocke les fichiers de configurations du système ainsi que des applications.
Un sous-répertoire par application

`/etc/init.d` et `/etc/default` : les fichiers liés aux daemons Linux

`/etc/password`, `/etc/group`, `/etc/shadow` : les fichiers de configuration des utilisateurs Linux.

`/etc/hosts` : le fichier HOSTS de Linux (liens entre adresses IP et littérales)

`/etc/sudoers` et `/etc/sudoers.d` : la configuration de sudo.

`/etc/sysctl.conf` et `/etc/sysctl.d` les fichiers de configuration de démarrage du noyau Linux.

...

Des fichiers sensibles

Quelques fichiers particulièrement importants, sur lesquels repose une grande partie de la stabilité du système, voire de son simple fonctionnement

`/etc/passwd` : fichier des utilisateurs et leurs mots de pass. Suppression de ce répertoire \implies impossible d'utiliser le système.

Des fichiers sensibles

Quelques fichiers particulièrement importants, sur lesquels repose une grande partie de la stabilité du système, voire de son simple fonctionnement

`/etc/passwd` : fichier des utilisateurs et leurs mots de pass. Suppression de ce répertoire \implies impossible d'utiliser le système.

`/etc/fstab` Liste des partitions utilisées par le système et selon quelle méthode il les utilise.

Des fichiers sensibles

Quelques fichiers particulièrement importants, sur lesquels repose une grande partie de la stabilité du système, voire de son simple fonctionnement

`/etc/passwd` : fichier des utilisateurs et leurs mots de pass. Suppression de ce répertoire \implies impossible d'utiliser le système.

`/etc/fstab` Liste des partitions utilisées par le système et selon quelle méthode il les utilise.

`/boot/vmlinuz` Se situe généralement, soit sous la racine (/), soit sous /boot. En fait, il s'agit du système lui-même ! Ultra sensible !

Des fichiers sensibles

Quelques fichiers particulièrement importants, sur lesquels repose une grande partie de la stabilité du système, voire de son simple fonctionnement

`/etc/passwd` : fichier des utilisateurs et leurs mots de pass. Suppression de ce répertoire \implies impossible d'utiliser le système.

`/etc/fstab` Liste des partitions utilisées par le système et selon quelle méthode il les utilise.

`/boot/vmlinuz` Se situe généralement, soit sous la racine (/), soit sous /boot. En fait, il s'agit du système lui-même ! Ultra sensible !

...

- 1 Système d'exploitation
- 2 Présentation de Linux
- 3 Système de fichiers
 - Vue logique
 - Les principaux répertoires systèmes
- 4 Le shell bash

Avertissement

On ne donne ici que quelques indications sur le shell **bash** :

- des principes généraux d'écritures des commandes

Avertissement

On ne donne ici que quelques indications sur le shell **bash** :

- des principes généraux d'écritures des commandes
- quelques commandes forcément incomplètes

Avertissement

On ne donne ici que quelques indications sur le shell **bash** :

- des principes généraux d'écritures des commandes
- quelques commandes forcément incomplètes
- pas d'instruction de programmation pour les scripts **bash** : on a assez à faire avec le langage C et OCAML.

Terminal et console

Terminal : environnement dans lequel on écrit et qui donne le retour des commandes.

Terminal et console

- Terminal** : environnement dans lequel on écrit et qui donne le retour des commandes.
- Il peut être fourni par les serveur graphique et disposer de fenêtres, menus, et autres boutons ;

Terminal et console

Terminal : environnement dans lequel on écrit et qui donne le retour des commandes.

- Il peut être fourni par le serveur graphique et disposer de fenêtres, menus, et autres boutons ;
- ou sans fenêtre, comme lorsque on fait Ctrl+Alt+F3 (entrer Ctrl+Alt+F7 ou Ctrl+Alt+F2 pour retourner au serveur graphique).

Terminal et console

Terminal : environnement dans lequel on écrit et qui donne le retour des commandes.

- Il peut être fourni par le serveur graphique et disposer de fenêtres, menus, et autres boutons ;
- ou sans fenêtre, comme lorsque on fait Ctrl+Alt+F3 (entrer Ctrl+Alt+F7 ou Ctrl+Alt+F2 pour retourner au serveur graphique).
- Peut très bien être constitué d'une imprimante avec un clavier comme un téléscripneur.

Terminal et console

Terminal : environnement dans lequel on écrit et qui donne le retour des commandes.

- Il peut être fourni par le serveur graphique et disposer de fenêtres, menus, et autres boutons ;
- ou sans fenêtre, comme lorsque on fait Ctrl+Alt+F3 (entrer Ctrl+Alt+F7 ou Ctrl+Alt+F2 pour retourner au serveur graphique).
- Peut très bien être constitué d'une imprimante avec un clavier comme un téléscripneur.

shell : interpréteur de commande ; programme lancé juste après la procédure de login et qui traite les commandes passées. Le shell le plus répandu sous **Linux** est le *bash* (Bourne again shell).

Terminal et console

Terminal : environnement dans lequel on écrit et qui donne le retour des commandes.

- Il peut être fourni par le serveur graphique et disposer de fenêtres, menus, et autres boutons ;
- ou sans fenêtre, comme lorsque on fait Ctrl+Alt+F3 (entrer Ctrl+Alt+F7 ou Ctrl+Alt+F2 pour retourner au serveur graphique).
- Peut très bien être constitué d'une imprimante avec un clavier comme un téléscripteur.

shell : interpréteur de commande ; programme lancé juste après la procédure de login et qui traite les commandes passées. Le shell le plus répandu sous **Linux** est le *bash* (Bourne again shell).

Console : combinaison d'un terminal et d'un shell.

Terminal et console

Terminal : environnement dans lequel on écrit et qui donne le retour des commandes.

- Il peut être fourni par le serveur graphique et disposer de fenêtres, menus, et autres boutons ;
- ou sans fenêtre, comme lorsque on fait Ctrl+Alt+F3 (entrer Ctrl+Alt+F7 ou Ctrl+Alt+F2 pour retourner au serveur graphique).
- Peut très bien être constitué d'une imprimante avec un clavier comme un téléscripneur.

shell : interpréteur de commande ; programme lancé juste après la procédure de login et qui traite les commandes passées. Le shell le plus répandu sous **Linux** est le *bash* (Bourne again shell).

Console : combinaison d'un terminal et d'un shell.

Ouverture d'un terminal sous Ubuntu : Ctr+Alt+T.

Un terminal

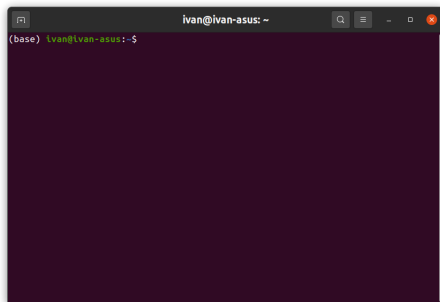


FIGURE – Un terminal dans lequel on n'a encore rien écrit

- Ouverture avec `Ctrl + Alt + T`
- Avant le prompt `$` : nom d'utilisateur @ nom de machine puis répertoire courant
- `~` désigne le répertoire racine de mon compte personnel.

Syntaxe générale des commandes

nom [-options] [argument1...]

- Explications :

Syntaxe générale des commandes

nom [-options] [argument1...]

- Explications :

`nom` nom de la commande

Syntaxe générale des commandes

nom [-options] [argument1...]

- Explications :

nom nom de la commande

options représente une ou plusieurs options

Syntaxe générale des commandes

nom [-options] [argument1...]

- Explications :

nom nom de la commande

options représente une ou plusieurs options

argument1 est le 1er argument

Syntaxe générale des commandes

nom [-options] [argument1...]

- Explications :

nom nom de la commande

options représente une ou plusieurs options

argument1 est le 1er argument

- les options sont écrites le plus souvent sous la forme d'un caractère accolé à un tiret (**ls -l**) mais pas toujours, par exemple **gcc -version**.

Les options courtes sont introduites par un tiret (**-a**), les longues par deux tirets (**-all**).

Syntaxe générale des commandes

nom [-options] [argument1...]

- Explications :

nom nom de la commande

options représente une ou plusieurs options

argument1 est le 1er argument

- les options sont écrites le plus souvent sous la forme d'un caractère accolé à un tiret (**ls -l**) .
- si paramètre demandé, il est séparé par un espace :
gcc essai.c -o sortie (2 paramètres : **essai.c** et **sortie**)

Syntaxe générale des commandes

nom [-options] [argument1...]

- Explications :

nom nom de la commande

options représente une ou plusieurs options

argument1 est le 1er argument

- les options sont écrites le plus souvent sous la forme d'un caractère accolé à un tiret (**ls -l**) .
- si paramètre demandé, il est séparé par un espace :
gcc essai.c -o sortie (2 paramètres : **essai.c** et **sortie**)
- les crochets indiquent un élément facultatif

Syntaxe générale des commandes

nom [-options] [argument1...]

- Explications :

nom nom de la commande

options représente une ou plusieurs options

argument1 est le 1er argument

- les options sont écrites le plus souvent sous la forme d'un caractère accolé à un tiret (**ls -l**) .
- si paramètre demandé, il est séparé par un espace :
gcc essai.c -o sortie (2 paramètres : **essai.c** et **sortie**)
- les crochets indiquent un élément facultatif
- les points de suspension indiquent la possibilité de répéter un argument, par exemple **ls /etc /usr/bin**

Syntaxe générale des commandes

nom [-options] [argument1...]

- Explications :

nom nom de la commande

options représente une ou plusieurs options

argument1 est le 1er argument

- les options sont écrites le plus souvent sous la forme d'un caractère accolé à un tiret (**ls -l**) .
- si paramètre demandé, il est séparé par un espace :
gcc essai.c -o sortie (2 paramètres : **essai.c** et **sortie**)
- les crochets indiquent un élément facultatif
- les points de suspension indiquent la possibilité de répéter un argument, par exemple **ls /etc /usr/bin**
- séparation par un espace ou une tabulation

Commandes internes vs externes

- Une commande *externe* est un fichier présent dans l'arborescence. Exemple : Quand un utilisateur exécute la commande **ls**, le shell demande au noyau Linux d'exécuter le fichier **/bin/ls**

```
$file /bin/ls
/bin/ls: ELF 64-bit LSB shared object, x86-64,
version 1 (SYSV), dynamically linked, [...]
```

Commandes internes vs externes

- Une commande *externe* est un fichier présent dans l'arborescence. Exemple : Quand un utilisateur exécute la commande **ls**, le shell demande au noyau Linux d'exécuter le fichier **/bin/ls**

```
$file /bin/ls
/bin/ls: ELF 64-bit LSB shared object, x86-64,
version 1 (SYSV), dynamically linked, [...]
```

- Une commande *interne* est intégrée au processus shell. Elle n'a aucune correspondance avec un fichier sur le disque. L'accès à une commande interne est plus rapide que pour une externe.

Commandes internes vs externes

- Une commande *externe* est un fichier présent dans l'arborescence. Exemple : Quand un utilisateur exécute la commande **ls**, le shell demande au noyau Linux d'exécuter le fichier **/bin/ls**

```
$file /bin/ls
/bin/ls: ELF 64-bit LSB shared object, x86-64,
version 1 (SYSV), dynamically linked, [...]
```

- Une commande *interne* est intégrée au processus shell. Elle n'a aucune correspondance avec un fichier sur le disque. L'accès à une commande interne est plus rapide que pour une externe.
- La commande **type** indique si une commande est interne ou externe.

```
$ type ls
ls est un alias vers << ls --color=auto >>
$ type cd
cd est une primitive du shell
```

Commandes internes vs externes

- Une commande *externe* est un fichier présent dans l'arborescence. Exemple : Quand un utilisateur exécute la commande **ls**, le shell demande au noyau Linux d'exécuter le fichier **/bin/ls**

```
$file /bin/ls
/bin/ls: ELF 64-bit LSB shared object, x86-64,
version 1 (SYSV), dynamically linked, [...]
```

- Une commande *interne* est intégrée au processus shell. Elle n'a aucune correspondance avec un fichier sur le disque. L'accès à une commande interne est plus rapide que pour une externe.
- La commande **type** indique si une commande est interne ou externe.

```
$ type ls
ls est un alias vers << ls --color=auto >>
$ type cd
cd est une primitive du shell
```

- certaines commande ont une version externe et une interne.

la commande **man**

- Pour connaître le mode d'emploi d'une commande, taper **man nomDeLaCommande** (ex : **man ls** pour connaître le manuel de **ls**). Entrer **q** pour quitter. ♡

la commande **man**

- Pour connaître le mode d'emploi d'une commande, taper **man nomDeLaCommande** (ex : **man ls** pour connaître le manuel de **ls**). Entrer **q** pour quitter. ♡
- Consulter la [documentation Ubuntu](#)

la commande **man**

- Pour connaître le mode d'emploi d'une commande, taper **man nomDeLaCommande** (ex : **man ls** pour connaître le manuel de **ls**). Entrer **q** pour quitter. ♡
- Consulter la [documentation Ubuntu](#)
- Le manuel est décomposé en plusieurs sections

la commande **man**

- Pour connaître le mode d'emploi d'une commande, taper **man nomDeLaCommande** (ex : **man ls** pour connaître le manuel de **ls**). Entrer **q** pour quitter. ♡
- Consulter la [documentation Ubuntu](#)
- Le manuel est décomposé en plusieurs sections
 - ① Programmes exécutables ou commandes de l'interpréteur de commandes (shell) ;

la commande **man**

- Pour connaître le mode d'emploi d'une commande, taper **man nomDeLaCommande** (ex : **man ls** pour connaître le manuel de **ls**). Entrer **q** pour quitter. ♡
- Consulter la [documentation Ubuntu](#)
- Le manuel est décomposé en plusieurs sections
 - 1 Programmes exécutables ou commandes de l'interpréteur de commandes (shell) ;
 - 2 Appels système (Fonctions fournies par le noyau) ;

la commande **man**

- Pour connaître le mode d'emploi d'une commande, taper **man nomDeLaCommande** (ex : **man ls** pour connaître le manuel de **ls**). Entrer **q** pour quitter. ♡
- Consulter la [documentation Ubuntu](#)
- Le manuel est décomposé en plusieurs sections
 - ① Programmes exécutables ou commandes de l'interpréteur de commandes (shell) ;
 - ② Appels système (Fonctions fournies par le noyau) ;
 - ③ Appels de bibliothèque (fonctions fournies par des bibliothèques) ;

la commande **man**

- Pour connaître le mode d'emploi d'une commande, taper **man nomDeLaCommande** (ex : **man ls** pour connaître le manuel de **ls**). Entrer **q** pour quitter. ♡
- Consulter la [documentation Ubuntu](#)
- Le manuel est décomposé en plusieurs sections
 - ① Programmes exécutables ou commandes de l'interpréteur de commandes (shell) ;
 - ② Appels système (Fonctions fournies par le noyau) ;
 - ③ Appels de bibliothèque (fonctions fournies par des bibliothèques) ;
 - ④ Fichiers spéciaux (situés généralement dans /dev) ;

la commande **man**

- Pour connaître le mode d'emploi d'une commande, taper **man nomDeLaCommande** (ex : **man ls** pour connaître le manuel de **ls**). Entrer **q** pour quitter. ♡
- Consulter la [documentation Ubuntu](#)
- Le manuel est décomposé en plusieurs sections
 - ① Programmes exécutables ou commandes de l'interpréteur de commandes (shell) ;
 - ② Appels système (Fonctions fournies par le noyau) ;
 - ③ Appels de bibliothèque (fonctions fournies par des bibliothèques) ;
 - ④ Fichiers spéciaux (situés généralement dans /dev) ;
 - ⑤ Formats des fichiers et conventions (Par exemple /etc/passwd) ;...

la commande **man**

- Parfois deux pages de manuel ont le même nom comme **printf** (en section 1 et 3). Entrer **man 1 printf** ou **man 3 printf** pour spécifier.

la commande **man**

- Parfois deux pages de manuel ont le même nom comme **printf** (en section 1 et 3). Entrer **man 1 printf** ou **man 3 printf** pour spécifier.
- Entrer **q** pour quitter le manuel.

Chemins absolus et relatifs

consulter l'exemple d'arborescence

- Un élément de l'arborescence est repéré par son nom (par exemple **nom.jpg**) précédé de :

/ : sépare les noms de fichiers

Chemins absolus et relatifs ♡

consulter l'exemple d'arborescence

- Un élément de l'arborescence est repéré par son nom (par exemple **nom.jpg**) précédé de :
 - son chemin absolu depuis la racine (ex :
/home/sue/Pictures/family/nom.jpg)

/ : sépare les noms de fichiers

Chemins absolus et relatifs ♥

consulter l'exemple d'arborescence

- Un élément de l'arborescence est repéré par son nom (par exemple **nom.jpg**) précédé de :
 - son chemin absolu depuis la racine (ex :
/home/sue/Pictures/family/nom.jpg)
 - son chemin relatif depuis le répertoire courant. Par exemple, si je suis dans **pets** : **../.. /family/nom.jpg**
- / : sépare les noms de fichiers

Chemins absolus et relatifs ♥

consulter l'exemple d'arborescence

- Un élément de l'arborescence est repéré par son nom (par exemple **nom.jpg**) précédé de :
 - son chemin absolu depuis la racine (ex : **/home/sue/Pictures/family/nom.jpg**)
 - son chemin relatif depuis le répertoire courant. Par exemple, si je suis dans **pets** : **../..family/nom.jpg**
- / : sépare les noms de fichiers
- ~ : répertoire racine du Home

Commentaire

Les commentaires ne sont pas interprétés :

- La commande **kalin** n'existe pas, elle soulève une erreur :

```
$ kalin
```

```
La commande <<kalin>> n'a pas été trouvée, ...
```

Commentaire

Les commentaires ne sont pas interprétés :

- La commande **kalin** n'existe pas, elle soulève une erreur :

```
$ kalin  
La commande <<kalin>> n'a pas été trouvée, ...
```

- Le caractère **#** n'est pas interprété. On peut écrire **kalin** sans erreur :

```
$ # kalin  
$
```

Contenu d'un répertoire

La commande **ls** donne le contenu d'un répertoire :

- sans argument : les entrées du répertoire courant **ls**

Contenu d'un répertoire ♥

La commande **ls** donne le contenu d'un répertoire :

- sans argument : les entrées du répertoire courant **ls**
- avec argument : les entrées repérées par le (ou les) argument(s) : **ls myFile, ls myRep, ls /etc /usr/bin**

Contenu d'un répertoire ♥

La commande **ls** donne le contenu d'un répertoire :

- sans argument : les entrées du répertoire courant **ls**
- avec argument : les entrées repérées par le (ou les) argument(s) : **ls myFile, ls myRep, ls /etc /usr/bin**
- pour afficher les fichiers cachés **ls -a** (indique notamment **.bashrc** si je suis en ~)

Contenu d'un répertoire ♥

La commande **ls** donne le contenu d'un répertoire :

- sans argument : les entrées du répertoire courant **ls**
- avec argument : les entrées repérées par le (ou les) argument(s) : **ls myFile, ls myRep, ls /etc /usr/bin**
- pour afficher les fichiers cachés **ls -a** (indique notamment **.bashrc** si je suis en ~)
- pour tous les attributs (type, droits, liens physiques, propriétaire, groupe, taille, date, nom) **ls -l**

Contenu d'un répertoire ♥

La commande **ls** donne le contenu d'un répertoire :

- sans argument : les entrées du répertoire courant **ls**
- avec argument : les entrées repérées par le (ou les) argument(s) : **ls myFile, ls myRep, ls /etc /usr/bin**
- pour afficher les fichiers cachés **ls -a** (indique notamment **.bashrc** si je suis en ~)
- pour tous les attributs (type, droits, liens physiques, propriétaire, groupe, taille, date, nom) **ls -l**
- **ls -al** au lieu de **ls -a -l**.

```
$ ls -lh
total 1,8M
-rw-rw-r-- 1 ivan ivan 67K août 25 2021 accesDirect.png
-rw-rw-r-- 1 ivan ivan 138K août 25 2021 accesSequentiel.png
```

Dans l'ordre : droits, nombres d'alias, username, groupname, taille (par défaut en octet), date de dernière modif., nom de fichier.

Contenu d'un répertoire ♡

La commande **ls** donne le contenu d'un répertoire :

- sans argument : les entrées du répertoire courant **ls**
- avec argument : les entrées repérées par le (ou les) argument(s) : **ls myFile, ls myRep, ls /etc /usr/bin**
- pour afficher les fichiers cachés **ls -a** (indique notamment **.bashrc** si je suis en ~)
- pour tous les attributs (type, droits, liens physiques, propriétaire, groupe, taille, date, nom) **ls -l**
- **ls -al** au lieu de **ls -a -l**.

```
$ ls -lh
total 1,8M
-rw-rw-r-- 1 ivan ivan 67K août 25 2021 accesDirect.png
-rw-rw-r-- 1 ivan ivan 138K août 25 2021 accesSequentiel.png
```

Dans l'ordre : droits, nombres d'alias, username, groupname, taille (par défaut en octet), date de dernière modif., nom de fichier.

- **ls -i** affiche le numéro d'inode.

Affichage d'une chaîne de caractère

La commande **echo** affiche une ligne de texte

- (Le caractère **#** indique le début d'un commentaire

```
$ echo 'coucou' # afficher coucou  
coucou
```

Affichage d'une chaîne de caractère

La commande **echo** affiche une ligne de texte

- (Le caractère **#** indique le début d'un commentaire

```
$ echo 'coucou' # afficher coucou  
coucou
```

- le choix des guillemets est important : « ' » (touche 4),
« " » (touche 3) et « ' » (ALT GR + 7) n'ont pas le même sens.

Affichage d'une chaîne de caractère

La commande **echo** affiche une ligne de texte

- (Le caractère **#** indique le début d'un commentaire

```
$ echo 'coucou' # afficher coucou
coucou
```

- le choix des guillemets est important : « ' » (touche 4), « " » (touche 3) et « ' » (ALT GR + 7) n'ont pas le même sens.
- Pour afficher le contenu d'une variable d'environnement :

```
$ echo $LANG
fr_FR.UTF-8
```

Les métacaractères

Les métacaractères du shell permettent :

- de construire des chaînes de caractères génériques

Les métacaractères

Les métacaractères du shell permettent :

- de construire des chaînes de caractères génériques
- de modifier l'interprétation d'une commande

Métacaractères de construction

Prioritaires : *, ?

- * désigne une chaîne de caractères quelconque ♡

Métacaractères de construction

Prioritaires : *, ?

- * désigne une chaîne de caractères quelconque ♡
- ? désigne un caractère quelconque ♡

Métacaractères de construction

Prioritaires : *, ?

- * désigne une chaîne de caractères quelconque ♥
- ? désigne un caractère quelconque ♥
- [...] désigne les caractères entre crochets, définis par énumération ou par un intervalle :

Métacaractères de construction

Prioritaires : *, ?

- * désigne une chaîne de caractères quelconque ♥
- ? désigne un caractère quelconque ♥
- [...] désigne les caractères entre crochets, définis par énumération ou par un intervalle :
 - [Aa] désigne les caractères A ou a,

Métacaractères de construction

Prioritaires : *, ?

- * désigne une chaîne de caractères quelconque ♡
- ? désigne un caractère quelconque ♡
- [...] désigne les caractères entre crochets, définis par énumération ou par un intervalle :
 - [Aa] désigne les caractères A ou a,
 - [0-9a-zA-Z] désigne un caractère alphanumérique quelconque.

Métacaractères de construction

Prioritaires : *, ?

- * désigne une chaîne de caractères quelconque ♡
- ? désigne un caractère quelconque ♡
- [...] désigne les caractères entre crochets, définis par énumération ou par un intervalle :
 - [Aa] désigne les caractères A ou a,
 - [0-9a-zA-Z] désigne un caractère alphanumérique quelconque.
 - [!0-9] désigne l'ensemble des caractères sauf les chiffres.

Métacaractères de construction

Voici le contenu du répertoire courant :

```
$ ls  
alain Ali tata titi toto tutu zut
```

- **ls t[ao]t[ao]** retourne **tata** et **toto**,

Métacaractères de construction

Voici le contenu du répertoire courant :

```
$ ls  
alain Ali tata titi toto tutu zut
```

- **ls t[ao]t[ao]** retourne **tata** et **toto**,
- **ls ???** retourne les noms de 3 lettres donc **zut** et **Ali**

Métacaractères de construction

Voici le contenu du répertoire courant :

```
$ ls  
alain Ali tata titi toto tutu zut
```

- **ls t[ao]t[ao]** retourne **tata** et **toto**,
- **ls ???** retourne les noms de 3 lettres donc **zut** et **Ali**
- **ls A*** retourne les noms qui commencent par A donc **Ali**

Métacaractères de construction

Voici le contenu du répertoire courant :

```
$ ls  
alain Ali tata titi toto tutu zut
```

- **ls t[ao]t[ao]** retourne **tata** et **toto**,
- **ls ???** retourne les noms de 3 lettres donc **zut** et **Ali**
- **ls A*** retourne les noms qui commencent par A donc **Ali**
- **ls t??o** retourne les noms de 4 lettres qui terminent par o et commencent par t donc **toto**

Métacaractères de construction

Voici le contenu du répertoire courant :

```
$ ls  
alain Ali tata titi toto tutu zut
```

- **ls t[ao]t[ao]** retourne **tata** et **toto**,
- **ls ???** retourne les noms de 3 lettres donc **zut** et **Ali**
- **ls A*** retourne les noms qui commencent par A donc **Ali**
- **ls t??o** retourne les noms de 4 lettres qui terminent par o et commencent par t donc **toto**
- **ls [!b-z]*** désigne les noms qui ne commencent pas par une lettre entre b et z donc **alain** et **Ali**.

Métacaractères de modification (PI)

- ; sépare deux commandes sur une même ligne

Métacaractères de modification (PI)

- ; sépare deux commandes sur une même ligne
- Les guillemets verticaux simples ' (touche 4) délimitent une chaîne de caractères contenant des espaces (à l'intérieur, tous les métacaractères perdent leur signification) ;

Métacaractères de modification (PI)

- ; sépare deux commandes sur une même ligne
- Les guillemets verticaux simples ' (touche 4) délimitent une chaîne de caractères contenant des espaces (à l'intérieur, tous les métacaractères perdent leur signification) ;
- Les guillemets verticaux doubles " (touche 3) délimitent une chaîne de caractères contenant des espaces (à l'intérieur, tous les métacaractères perdent leur signification, à l'exception des métacaractères ' et \$) ;

Métacaractères de modification (PI)

- ; sépare deux commandes sur une même ligne
- Les guillemets verticaux simples ' (touche 4) délimitent une chaîne de caractères contenant des espaces (à l'intérieur, tous les métacaractères perdent leur signification) ;
- Les guillemets verticaux doubles " (touche 3) délimitent une chaîne de caractères contenant des espaces (à l'intérieur, tous les métacaractères perdent leur signification, à l'exception des métacaractères ` et \$) ;
- Les guillemets obliques gauche-droite ` (ALT GR + 7) « capturent » la sortie standard pour former un nouvel argument ou une nouvelle commande ;

```
$ echo "$LANG"; echo '$LANG' # deux commandes successives
fr_FR.UTF-8
$LANG
$ echo "on est `date`" # observer les guillemets
on est jeu. 25 janv. 2024 19:17:45 CET
```

Métacaractères de modification (PI)

- \ annule la signification du métacaractère qui suit : c'est un caractère dit d'*échappement*.

```
$ echo "on est `date`"  
on est `date`
```

Métacaractères de modification (PI)

- `\` annule la signification du métacaractère qui suit : c'est un caractère dit d'*échappement*.

```
$ echo "on est `date`"
on est `date`
```

- le **&** à la fin d'une commande permet de lancer celle-ci en tâche de fond, donc sans bloquer le terminal.

```
$emacs toto # mon terminal va se bloquer
```

CTRL + C pour quitter brutalement. Ou mieux CTRL + z (passage à l'état zombi) suivi de **bg** (remise en tâche de fond).

```
$emacs toto & # mon terminal ne pas se bloquer
```


Métacaractères de modification (PI)

Parenthèses

- **(...)** : les parenthèses encadrent une suite de commandes qui sont exécutées par un shell secondaire. En particulier, les assignments n'ont pas d'effet en dehors des parenthèses.

Métacaractères de modification (PI)

Parenthèses

- **(...)** : les parenthèses encadrent une suite de commandes qui sont exécutées par un shell secondaire. En particulier, les assignments n'ont pas d'effet en dehors des parenthèses.
- **{...}** : les accolades encadrent une suite de commandes qui sont exécutées par le shell principal. En particulier, les assignments ont un effet en dehors des accolades.

Métacaractères de modification (PI)

Parenthèses

- **(...)** : les parenthèses encadrent une suite de commandes qui sont exécutées par un shell secondaire. En particulier, les assignments n'ont pas d'effet en dehors des parenthèses.
- **{...}** : les accolades encadrent une suite de commandes qui sont exécutées par le shell principal. En particulier, les assignments ont un effet en dehors des accolades.
- **[...]** : les crochets sont utilisés pour les instructions conditionnelles. Ils encadrent une expression à valeur booléenne.

```
$ [ -d presentationLinux.tex ] # est-ce un dossier ?  
$ echo $? # afficher la réponse du test précédent  
1
```

On obtient 0 si le fichier existe et est un répertoire, 1 sinon.

Métacaractères de modification (PI)

Parenthèses

- **(...)** : les parenthèses encadrent une suite de commandes qui sont exécutées par un shell secondaire. En particulier, les assignments n'ont pas d'effet en dehors des parenthèses.
- **{...}** : les accolades encadrent une suite de commandes qui sont exécutées par le shell principal. En particulier, les assignments ont un effet en dehors des accolades.
- **[...]** : les crochets sont utilisés pour les instructions conditionnelles. Ils encadrent une expression à valeur booléenne.

```
$ [ -d presentationLinux.tex ] # est-ce un dossier ?  
$ echo $? # afficher la réponse du test précédent  
1
```

On obtient 0 si le fichier existe et est un répertoire, 1 sinon.

- ...

Utiliser le résultat d'une commande comme argument d'une autre (PI)

Pour info. Les ` (ALT GR + 7) entourant une commande permettent d'utiliser le résultat de cette commande comme argument(s) dans la ligne de commande.

```
$ echo "Nous sommes le" `date +%d/%m/%y`  
Nous sommes le 27/08/21
```

Affiche la date du jour avec un format choisi.

Utiliser le résultat d'une commande comme argument d'une autre (PI)

Pour info. Les ``` (ALT GR + 7) entourant une commande permettent d'utiliser le résultat de cette commande comme argument(s) dans la ligne de commande.

```
$ echo "Nous sommes le" `date +%d/%m/%y`  
Nous sommes le 27/08/21
```

Affiche la date du jour avec un format choisi.

```
$ echo "2 + 2 =" `expr 2 + 2`  
2 + 2 = 4
```

Positionnement/recherche dans l'arborescence

`pwd` affiche le nom absolu du répertoire de travail

Positionnement/recherche dans l'arborescence

`pwd` affiche le nom absolu du répertoire de travail

`cd` change le répertoire de travail.

Positionnement/recherche dans l'arborescence

`pwd` affiche le nom absolu du répertoire de travail

`cd` change le répertoire de travail.

- Avec argument : se rend à la destination indiquée. **cd**
../Rep1 ;

Positionnement/recherche dans l'arborescence ♡

`pwd` affiche le nom absolu du répertoire de travail

`cd` change le répertoire de travail.

- Avec argument : se rend à la destination indiquée. **cd** `../Rep1` ;
- sans argument : retourne au répertoire de connexion du user. **cd**

Positionnement/recherche dans l'arborescence ♡

`pwd` affiche le nom absolu du répertoire de travail

`cd` change le répertoire de travail.

- Avec argument : se rend à la destination indiquée. **cd** `../Rep1` ;
- sans argument : retourne au répertoire de connexion du user. **cd**

`ls` liste les entrées d'un répertoire (déjà vu) **ls**.

Positionnement/recherche dans l'arborescence ♥

pwd affiche le nom absolu du répertoire de travail

cd change le répertoire de travail.

- Avec argument : se rend à la destination indiquée. **cd ../Rep1** ;
- sans argument : retourne au répertoire de connexion du user. **cd**

ls liste les entrées d'un répertoire (déjà vu) **ls**.

find pour chercher récursivement un ou plusieurs fichiers dans une arborescence. Beaucoup d'options (consulter le manuel).

Rechercher

- **find** cherche récursivement dans l'arborescence à partir du point indiqué. **find /usr -name "ls*"** cherche les fichiers dont le nom commence par **ls** dans le répertoire **/usr** et ses sous-répertoires. Il y en a beaucoup! ♡

Rechercher

- **find** cherche récursivement dans l'arborescence à partir du point indiqué. **find /usr -name "ls*"** cherche les fichiers dont le nom commence par **ls** dans le répertoire **/usr** et ses sous-répertoires. Il y en a beaucoup! ♡
- L'option **-type** permet de ne chercher que les fichiers (**f**) ou les répertoires (**d**). **find /var/log/ -type d -name "*sm*"** : chercher les répertoires dont le nom contient sm

Rechercher

- **find** cherche récursivement dans l'arborescence à partir du point indiqué. **find /usr -name "ls*"** cherche les fichiers dont le nom commence par **ls** dans le répertoire **/usr** et ses sous-répertoires. Il y en a beaucoup! ♡
- L'option **-type** permet de ne chercher que les fichiers (**f**) ou les répertoires (**d**). **find /var/log/ -type d -name "*sm*"** : chercher les répertoires dont le nom contient sm
- recherche par taille : **find /Téléchargements -size +20M -size -40M** cherche les fichiers dont la taille est comprise entre 20 Mo et 40Mo.

Rechercher

- **find** cherche récursivement dans l'arborescence à partir du point indiqué. **find /usr -name "ls*"** cherche les fichiers dont le nom commence par **ls** dans le répertoire **/usr** et ses sous-répertoires. Il y en a beaucoup! ♡
- L'option **-type** permet de ne chercher que les fichiers (**f**) ou les répertoires (**d**). **find /var/log/ -type d -name "*sm*"** : chercher les répertoires dont le nom contient sm
- recherche par taille : **find /Téléchargements -size +20M -size -40M** cherche les fichiers dont la taille est comprise entre 20 Mo et 40Mo.
- Recherche par utilisateur : **find /tmp -user adrien** cherche dans **/tmp** les fichiers dont le propriétaire est **adrien**.

Rechercher

- **find** cherche récursivement dans l'arborescence à partir du point indiqué. **find /usr -name "ls*"** cherche les fichiers dont le nom commence par **ls** dans le répertoire **/usr** et ses sous-répertoires. Il y en a beaucoup! ♡
- L'option **-type** permet de ne chercher que les fichiers (**f**) ou les répertoires (**d**). **find /var/log/ -type d -name "*"sm*"** : chercher les répertoires dont le nom contient sm
- recherche par taille : **find /Téléchargements -size +20M -size -40M** cherche les fichiers dont la taille est comprise entre 20 Mo et 40Mo.
- Recherche par utilisateur : **find /tmp -user adrien** cherche dans **/tmp** les fichiers dont le propriétaire est **adrien**.
- Beaucoup d'autres options : par date de création, date de dernière modification, par type de permissions, recherche de fichiers vides etc...

Consulter le contenu d'un fichier texte

Commandes de base :

- **cat monfichier**, **more monfichier** : affichage simple et page par page ;

Consulter le contenu d'un fichier texte

Commandes de base :

- **cat monfichier**, **more monfichier** : affichage simple et page par page ;
- **head monfichier**, **head -n monfichier** : affichage des n premières lignes ;

Consulter le contenu d'un fichier texte ♡

Commandes de base :

- **cat monfichier**, **more monfichier** : affichage simple et page par page ;
- **head monfichier**, **head -n monfichier** : affichage des n premières lignes ;
- **tail monfichier**, **tail -n monfichier** : affichage des n dernières lignes ;

Consulter le contenu d'un fichier texte ♥

Commandes de base :

- **cat monfichier**, **more monfichier** : affichage simple et page par page ;
- **head monfichier**, **head -n monfichier** : affichage des n premières lignes ;
- **tail monfichier**, **tail -n monfichier** : affichage des n dernières lignes ;
- **wc monfichier** : affichage du nombre de lignes, de mots, de caractères. Options **-l**, **-w** et **-c** pour les nombres de lignes, de mots et de caractères.

Consulter le contenu d'un fichier texte ♡

Commandes de base :

- **cat monfichier**, **more monfichier** : affichage simple et page par page ;
- **head monfichier**, **head -n monfichier** : affichage des n premières lignes ;
- **tail monfichier**, **tail -n monfichier** : affichage des n dernières lignes ;
- **wc monfichier** : affichage du nombre de lignes, de mots, de caractères. Options **-l**, **-w** et **-c** pour les nombres de lignes, de mots et de caractères.
- **cat toto titi** : affiche le contenu de **titi** à la suite de celui de **toto** (**cat** pour concatène).

Historique

- Le shell **bash** enregistre toutes les commandes tapées et permet de les rappeler pour les ré-exécuter soit telles quelles, soit modifiées.

Historique

- Le shell **bash** enregistre toutes les commandes tapées et permet de les rappeler pour les ré-exécuter soit telles quelles, soit modifiées.
- La commande **history** permet de lister le contenu de l'historique des commandes, de façon numérotée. Le caractère **!** permet de rappeler une commande.

Historique

- Le shell **bash** enregistre toutes les commandes tapées et permet de les rappeler pour les ré-exécuter soit telles quelles, soit modifiées.
- La commande **history** permet de lister le contenu de l'historique des commandes, de façon numérotée. Le caractère **!** permet de rappeler une commande.

!! rappelle la dernière commande

Historique

- Le shell **bash** enregistre toutes les commandes tapées et permet de les rappeler pour les ré-exécuter soit telles quelles, soit modifiées.
- La commande **history** permet de lister le contenu de l'historique des commandes, de façon numérotée. Le caractère **!** permet de rappeler une commande.

!! rappelle la dernière commande

!n rappelle la commande numéro *n*

Historique

- Le shell **bash** enregistre toutes les commandes tapées et permet de les rappeler pour les ré-exécuter soit telles quelles, soit modifiées.
- La commande **history** permet de lister le contenu de l'historique des commandes, de façon numérotée. Le caractère **!** permet de rappeler une commande.

!! rappelle la dernière commande

!n rappelle la commande numéro *n*

!chaine rappelle la dernière commande commençant par *chaine*

Historique ♥

- Le shell **bash** enregistre toutes les commandes tapées et permet de les rappeler pour les ré-exécuter soit telles quelles, soit modifiées.
- La commande **history** permet de lister le contenu de l'historique des commandes, de façon numérotée. Le caractère **!** permet de rappeler une commande.
 - !! rappelle la dernière commande
 - !*n* rappelle la commande numéro *n*
 - !*chaine* rappelle la dernière commande commençant par *chaine*
- On peut aussi utiliser les flèches haut et bas pour naviguer dans l'historique des commandes.

Taille du contenu d'un répertoire

- **du -h -d 1 monRepertoire** : taille des fichiers et sous-répertoires. (du pour Disk User))

Taille du contenu d'un répertoire

- **du -h -d 1 monRepertoire** : taille des fichiers et sous-répertoires. (**du** pour Disk User)
 - L'option **-h** force un affichage « human readable » (par exemple, 1k, 236M, 2G).

Taille du contenu d'un répertoire

- **du -h -d 1 monRépertoire** : taille des fichiers et sous-répertoires. (**du** pour Disk User)
 - L'option **-h** force un affichage « human readable » (par exemple, 1k, 236M, 2G).
 - L'option **-d** affiche la taille totale du répertoire exploré et pas seulement la taille de ses constituants. Et le paramètre 1 indique la profondeur de l'exploration (ici, on s'arrête aux fils, avec 2 comme paramètre , ce serait aux petits-fils)

Taille du contenu d'un répertoire

- **du -h -d 1 monRépertoire** : taille des fichiers et sous-répertoires. (**du** pour Disk User)
 - L'option **-h** force un affichage « human readable » (par exemple, 1k, 236M, 2G).
 - L'option **-d** affiche la taille totale du répertoire exploré et pas seulement la taille de ses constituants. Et le paramètre 1 indique la profondeur de l'exploration (ici, on s'arrête aux fils, avec 2 comme paramètre , ce serait aux petits-fils)
- La commande **ncdu nomDuRépertoire**, plus conviviale, permet de connaître la place prise par les fichiers et dossiers en navigant dans l'arborescence. Par exemple **ncdu /home** indique les tailles des différents répertoires utilisateurs.

Créer et supprimer

- Créer un répertoire vide : **mkdir Rep1**

Créer et supprimer

- Créer un répertoire vide : **mkdir Rep1**
- Supprimer un répertoire vide : **rmdir Rep1**

Créer et supprimer

- Créer un répertoire vide : **mkdir Rep1**
- Supprimer un répertoire vide : **rmdir Rep1**
- Créer un fichier vide : **touch file1**

Créer et supprimer

- Créer un répertoire vide : **mkdir Rep1**
- Supprimer un répertoire vide : **rmdir Rep1**
- Créer un fichier vide : **touch file1**
- Supprimer un fichier **rm file1**, supprimer tous les fichiers du répertoire **rm ***

Créer et supprimer

- Créer un répertoire vide : **mkdir Rep1**
- Supprimer un répertoire vide : **rmdir Rep1**
- Créer un fichier vide : **touch file1**
- Supprimer un fichier **rm file1**, supprimer tous les fichiers du répertoire **rm ***
- Créer un chemin complet **mkdir -p R1/R2** crée dans la foulée **R1** puis son sous-répertoire **R2**.

Copier (**cp**) et déplacer (**mv**) ♡

Situation : un répertoire parent **P** possède deux sous-répertoires **Rep1** et **Rep2**. Dans **Rep1** on trouve le fichier **file1**.

- Copier le fichier **file1** du répertoire **Rep1** dans le répertoire **Rep2** sans changer le nom : **cp Rep1/file1 Rep2/**

Copier (**cp**) et déplacer (**mv**) ♡

Situation : un répertoire parent **P** possède deux sous-répertoires **Rep1** et **Rep2**. Dans **Rep1** on trouve le fichier **file1**.

- Copier le fichier **file1** du répertoire **Rep1** dans le répertoire **Rep2** sans changer le nom : **cp Rep1/file1 Rep2/**
- Copier le fichier **file1** du répertoire **Rep1** dans le répertoire **Rep2** en changeant le nom : **cp Rep1/file1 Rep2/file2**

Copier (**cp**) et déplacer (**mv**) ♡

Situation : un répertoire parent **P** possède deux sous-répertoires **Rep1** et **Rep2**. Dans **Rep1** on trouve le fichier **file1**.

- Copier le fichier **file1** du répertoire **Rep1** dans le répertoire **Rep2** sans changer le nom : **cp Rep1/file1 Rep2/**
- Copier le fichier **file1** du répertoire **Rep1** dans le répertoire **Rep2** en changeant le nom : **cp Rep1/file1 Rep2/file2**
- Je suis dans **Rep1**. Changer le nom du fichier **file1** en restant dans le même répertoire : **mv file1 file2**.

Copier (**cp**) et déplacer (**mv**) ♡

Situation : un répertoire parent **P** possède deux sous-répertoires **Rep1** et **Rep2**. Dans **Rep1** on trouve le fichier **file1**.

- Copier le fichier **file1** du répertoire **Rep1** dans le répertoire **Rep2** sans changer le nom : **cp Rep1/file1 Rep2/**
- Copier le fichier **file1** du répertoire **Rep1** dans le répertoire **Rep2** en changeant le nom : **cp Rep1/file1 Rep2/file2**
- Je suis dans **Rep1**. Changer le nom du fichier **file1** en restant dans le même répertoire : **mv file1 file2**.
- Je suis dans **Rep1**. déplacer le fichier **file1** dans **rep2** sans changer son nom **mv file1 ../Rep2/**.

Copier (**cp**) et déplacer (**mv**) ♡

Situation : un répertoire parent **P** possède deux sous-répertoires **Rep1** et **Rep2**. Dans **Rep1** on trouve le fichier **file1**.

- Copier le fichier **file1** du répertoire **Rep1** dans le répertoire **Rep2** sans changer le nom : **cp Rep1/file1 Rep2/**
- Copier le fichier **file1** du répertoire **Rep1** dans le répertoire **Rep2** en changeant le nom : **cp Rep1/file1 Rep2/file2**
- Je suis dans **Rep1**. Changer le nom du fichier **file1** en restant dans le même répertoire : **mv file1 file2**.
- Je suis dans **Rep1**. déplacer le fichier **file1** dans **rep2** sans changer son nom **mv file1 ../Rep2/**.
- Je suis dans le répertoire parent de **P**. Je veux copier récursivement **P** et tout ce qu'il contient (donc aussi les sous-répertoires) dans un nouveau répertoire **P2** : **cp -r P P2**

Créer, remplir, vider, supprimer un répertoire

Exemple

```
$ mkdir Asup # Créer répertoire Asup
```

Créer, remplir, vider, supprimer un répertoire

Exemple

• `$ mkdir Asup # Créer répertoire Asup`

• `$ cd Asup # aller au répertoire Asup`
`$ ls # pas de contenu`

Créer, remplir, vider, supprimer un répertoire

Exemple

• `$ mkdir Asup # Créer répertoire Asup`

• `$ cd Asup # aller au répertoire Asup`

`$ ls # pas de contenu`

• `$ touch asup # créer le fichier vide asup`

`$ ls -al # afficher fichiers cachés`

total 8

drwxrwxr-x 2 ivan ivan 4096 août 19 15:28 .

drwxrwxr-x 3 ivan ivan 4096 août 19 15:28 ..

-rw-rw-r-- 1 ivan ivan 0 août 19 15:28 asup

Créer, remplir, vider, supprimer un répertoire

Exemple

```
$ mkdir Asup # Créer répertoire Asup
```

```
$ cd Asup # aller au répertoire Asup  
$ ls # pas de contenu
```

```
$ touch asup # créer le fichier vide asup  
$ ls -al # afficher fichiers cachés  
total 8  
drwxrwxr-x 2 ivan ivan 4096 août 19 15:28 .  
drwxrwxr-x 3 ivan ivan 4096 août 19 15:28 ..  
-rw-rw-r-- 1 ivan ivan 0 août 19 15:28 asup
```

```
$ rm asup # supprimer asup  
$ ls # plus de contenu
```

Créer, remplir, vider, supprimer un répertoire

Exemple

```
$ cd .. # revenir au père  
$ rmdir Asup # supprime rep Asup
```

Créer, remplir, vider, supprimer un répertoire

Exemple

- ```
$ cd .. # revenir au père
$ rmdir Asup # supprime rep Asup
```
- Les répertoires **Nouveau** et son fils **AutreNouveau** sont créés simultanément puis supprimés de même

```
$ mkdir -p Nouveau/AutreNouveau # creer un repertoire et son fils
$ rmdir -p Nouveau/AutreNouveau # supprimer un repertoire et son fils
```



## Explorer un fichier

**grep** affiche les lignes vérifiant un pattern.

Contenu d'un fichier `myfile` :

```
toto et gogo
tot et gogo
totoooo et gaga
atoto et titi
titi et tutu
```

- **grep "toto" myfile** cherche le mot « toto » dans **myfile**

## Explorer un fichier

**grep** affiche les lignes vérifiant un pattern.

Contenu d'un fichier `myfile` :

```
toto et gogo
```

```
tot et gogo
```

```
totoooo et gaga
```

```
atoto et titi
```

```
titi et tutu
```

- **grep "toto" myfile** cherche le mot « toto » dans **myfile**
- **grep -c "toto" myfile** cherche le nombre d'occurrences du mot « toto » dans **myfile**.

## Explorer un fichier

**grep** affiche les lignes vérifiant un pattern.

Contenu d'un fichier `myfile` :

```
toto et gogo
```

```
tot et gogo
```

```
totoooo et gaga
```

```
atoto et titi
```

```
titi et tutu
```

- **grep "toto" myfile** cherche le mot « toto » dans **myfile**
- **grep -c "toto" myfile** cherche le nombre d'occurrences du mot « toto » dans **myfile**.
- Option **grep -n "toto"** pour afficher les numéros de lignes.

## Explorer un fichier

**grep** affiche les lignes vérifiant un pattern.

Contenu d'un fichier `myfile` :

```
toto et gogo
```

```
tot et gogo
```

```
totoooo et gaga
```

```
atoto et titi
```

```
titi et tutu
```

- **grep "toto" myfile** cherche le mot « toto » dans **myfile**
- **grep -c "toto" myfile** cherche le nombre d'occurrences du mot « toto » dans **myfile**.
- Option **grep -n "toto"** pour afficher les numéros de lignes.
- Le joker `*` n'a pas ici la même signification que le métacaractère du shell ! **grep "toto\*" myfile** cherche les lignes contenant au moins un **tot** suivi par 0, 1 ou plusieurs lettres « o » (renvoie 4 lignes avec les radicaux `tot`, `toto`, `totoooo`).

## Explorer un fichier (suite)

**grep** affiche les lignes vérifiant un pattern.

Contenu d'un fichier `myfile` :

```
toto et gogo
```

```
tot et gogo
```

```
totoooo et gaga
```

```
atoto et titi
```

```
titi et tutu
```

- **grep "toto\$" myfile** : les lignes qui terminent par toto

## Explorer un fichier (suite)

**grep** affiche les lignes vérifiant un pattern.

Contenu d'un fichier `myfile` :

```
toto et gogo
```

```
tot et gogo
```

```
totoooo et gaga
```

```
atoto et titi
```

```
titi et tutu
```

- **grep "toto\$" myfile** : les lignes qui terminent par toto
- **grep "^toto"** : les lignes qui commencent par toto

## Explorer un fichier (suite)

**grep** affiche les lignes vérifiant un pattern.

Contenu d'un fichier `myfile` :

```
toto et gogo
```

```
tot et gogo
```

```
totoooo et gaga
```

```
atoto et titi
```

```
titi et tutu
```

- **grep "toto\$" myfile** : les lignes qui terminent par toto
- **grep "^toto"** : les lignes qui commencent par toto
- **grep -v "toto" myfile** : les lignes ne contenant pas toto

## Explorer un fichier (suite)

**grep** affiche les lignes vérifiant un pattern.

Contenu d'un fichier `myfile` :

```
toto et gogo
```

```
tot et gogo
```

```
totoooo et gaga
```

```
atoto et titi
```

```
titi et tutu
```

- **grep "toto\$" myfile** : les lignes qui terminent par toto
- **grep "^toto"** : les lignes qui commencent par toto
- **grep -v "toto" myfile** : les lignes ne contenant pas toto
- **grep -E "toto?"** : les lignes qui contiennent tot, toto mais pas totoo. L'option **E** permet de bénéficier des expressions régulières *étendues*.



# sed

**sed** (stream **e**ditor) permet de modifier ou de supprimer une partie d'une chaîne de caractères, par exemple pour remplacer un caractère par un autre dans un fichier, ou encore supprimer des chaînes de caractères inutiles.

- **sed 's/occurrence\_cherchée/occurrence\_substituée/g' fichier**  
Option **s** pour substituer, **g** pour appliquer récursivement la substitution.

```
$ sed "s/ //g" myfile # supprime tous les espaces
totoetgogo
totetgogo
...
$ sed "s/o/AAA/g" myfile #remplace o par AA
tAAAtAAA et gAAAgAAA
tAAAt et gAAAgAAA
```

# sed

**sed** (stream **e**ditor) permet de modifier ou de supprimer une partie d'une chaîne de caractères, par exemple pour remplacer un caractère par un autre dans un fichier, ou encore supprimer des chaînes de caractères inutiles.

- **sed 's/occurrence\_cherchée/occurrence\_substituée/g' fichier**  
Option **s** pour substituer, **g** pour appliquer récursivement la substitution.

```
$ sed "s/ //g" myfile # supprime tous les espaces
totoetgogo
totetgogo
...
$ sed "s/o/AAA/g" myfile #remplace o par AA
tAAAAtAAA et gAAAgAAA
tAAAAt et gAAAgAAA
```

- Normalisation des espaces (on les affiche pour comprendre) :

```
$ _export chaîne="un_ _et_ _ _ _ _deux" _#création_d'une_variable
$_echo $chaîne|_sed "s/_/_/g" _#normalisation_des_espaces
un_et_deux
```

# Montage et démontage

- L'arborescence de fichier de Linux inclut tous les périphériques externes.

# Montage et démontage

- L'arborescence de fichier de Linux inclut tous les périphériques externes.
- Le système d'exploitation d'un ordinateur basique est installé sur le disque dur. L'utilisateur peut décider d'insérer un périphérique externe (ex : une clé USB).

# Montage et démontage

- L'arborescence de fichier de Linux inclut tous les périphériques externes.
- Le système d'exploitation d'un ordinateur basique est installé sur le disque dur. L'utilisateur peut décider d'insérer un périphérique externe (ex : une clé USB).
- Les périphériques de stockage sont associés à un répertoire particulier. Par exemple le disque dur principal sur lequel l'OS est installé est associé au répertoire /

# Montage et démontage

- L'arborescence de fichier de Linux inclut tous les périphériques externes.
- Le système d'exploitation d'un ordinateur basique est installé sur le disque dur. L'utilisateur peut décider d'insérer un périphérique externe (ex : une clé USB).
- Les périphériques de stockage sont associés à un répertoire particulier. Par exemple le disque dur principal sur lequel l'OS est installé est associé au répertoire `/`
- L'utilisateur peut choisir de monter temporairement un système de fichier par ses propres moyens grâce à la commande **mount**. En général le montage se fait dans le répertoire `/mnt`.  
Les opérations de montage manuel nécessitent généralement les droits du superuser.

# Montage et démontage

- Les médias amovibles (comme une clé USB) sont, sous Ubuntu, *montés* automatiquement (*i.e.* appel automatique de **mount**) dans le répertoire `/media`.

## Montage et démontage

- Les médias amovibles (comme une clé USB) sont, sous Ubuntu, *montés* automatiquement (*i.e.* appel automatique de **mount**) dans le répertoire `/media`.
- Par exemple si l'utilisateur **toto** insère une clé USB, son arborescence de fichier est visible dans un répertoire comme `/media/toto/CLE`.



## Montage et démontage

- Les médias amovibles (comme une clé USB) sont, sous Ubuntu, *montés* automatiquement (*i.e.* appel automatique de **mount**) dans le répertoire `/media`.
- Par exemple si l'utilisateur **toto** insère une clé USB, son arborescence de fichier est visible dans un répertoire comme `/media/toto/CLE`.
- Tous les fichiers de la clés sont disponibles dans ce répertoire.

## Montage et démontage

- Les médias amovibles (comme une clé USB) sont, sous Ubuntu, *montés* automatiquement (*i.e.* appel automatique de **mount**) dans le répertoire `/media`.
- Par exemple si l'utilisateur **toto** insère une clé USB, son arborescence de fichier est visible dans un répertoire comme `/media/toto/CLE`.
- Tous les fichiers de la clés sont disponibles dans ce répertoire.
- Et les créations/modifications/suppressions de fichier dans ce répertoire sont répercutées sur la clé.

## Montage et démontage

- Les médias amovibles (comme une clé USB) sont, sous Ubuntu, *montés* automatiquement (*i.e.* appel automatique de **mount**) dans le répertoire `/media`.
- Par exemple si l'utilisateur **toto** insère une clé USB, son arborescence de fichier est visible dans un répertoire comme `/media/toto/CLE`.
- Tous les fichiers de la clés sont disponibles dans ce répertoire.
- Et les créations/modifications/suppressions de fichier dans ce répertoire sont répercutées sur la clé.
- Pour démonter un média amovible, il y a en général une interface graphique.

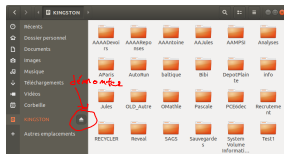


FIGURE – Un bouton pour « éjecter » la clé

# Montage et démontage

- Pour démonter manuellement un système de fichier, on utilise la commande **umount**.

# Montage et démontage

- Pour démonter manuellement un système de fichier, on utilise la commande **umount**.
- Cette commande requiert en général les droits du superuser.