PLAN

- Chapitre 1: **Notions fondamentales des Systèmes d'exploitation**
- Chapitre 2: Gestion des fichiers sous Linux
- Chapitre 3: Les droits d'accès
- Chapitre 4: Les filtres
- Chapitre 5: Les scripts Shell

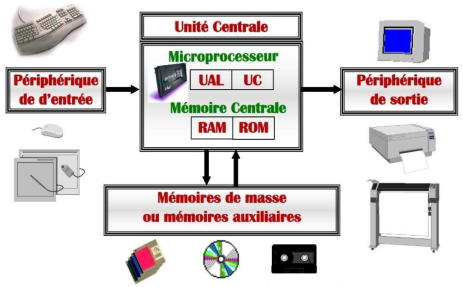
Chapitre 1

Notions fondamentales des systèmes d'exploitation

I. Introduction : Architecture de base d'un ordinateur

Comment fonctionne un ordinateur:

- Un microprocesseur : **UAL + registres + code opération → résultat.**
- Une **RAM** : permet le stockage de données temporairement.
- Un **BUS** (bus de données, bus de commandes , bus d'adresse) pour faire le lien.
- D'autres périphériques.



I. Introduction Architecture de base d'un ordinateur

- **Le processeur** (Central Processing Unit ou CPU) **:** est un circuit électronique qui assure les fonctions centrales de l'ordinateur. Il exécute les instructions constituant les différentes tâches demandées à l'ordinateur.

I. Introduction
Architecture de base d'un ordinateur

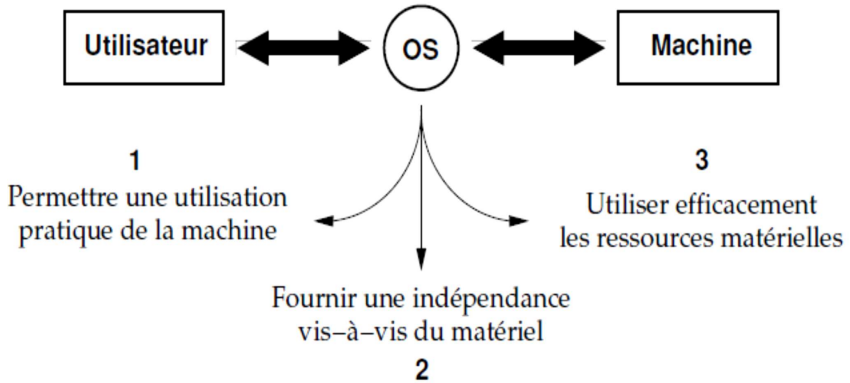
- **Mémoires :** sont des composants électroniques pouvant garder des informations temporairement ou à long terme.
  - Les **mémoires centrales** : sont utilisées pour stocker les informations nécessitant un accès rapide par le processeur. On distingue
    - les **mémoires vives** (*Random Access Memory* ou **RAM**)
    - et les **mémoires mortes** (*Read Only Memory* ou **ROM**).
  - Les **mémoires de masse** : ou mémoires auxiliaires sont utilisées pour stocker les informations à plus long terme comme les disques, les CDROM, flash disque.

II. Définition d'un système d'exploitation

Un **système d'exploitation** (noté **SE** et **OS** en anglais, abréviation du terme **Operating System**) est un ensemble de **programmes systèmes** qui permettent de gérer les différentes ressources matérielles (disques, mémoires, etc.) ou logicielles (compilateurs, traitements de textes, bases de données, etc.) de l'ordinateur.

II. Définition d'un système d'exploitation

Un SE est chargé d'assurer la liaison entre les ressources matérielles, l'utilisateur et les applications (traitement de texte, jeu vidéo, etc.).



III. Fonctions d'un système d'exploitation

- Le système d'exploitation possède deux principales fonctions :
  - **Machine virtuelle** (ou Virtualisation de la machine)
    - Vue uniforme des E/S
    - Gestion de la mémoire et des processus, réseau
    - Système de fichiers
  - **Gestion des ressources**
    - Fonctionnement des ressources (processeur, délais, ...)
    - Contrôle d'accès aux ressources (Allocation CPU, disque, mémoire, canal de communication réseau, ...)
    - Gestion des erreurs
    - Gestion des conflits

III. Fonctions d'un système d'exploitation
a) Virtualisation de la machine

- Un SE a pour tâche principale de présenter à l'utilisateur une machine virtuelle à la place de la machine physique permettant de gérer les ressources d'une manière **transparente**.
- Cette machine virtuelle est un ensemble de programmes qui masque la complexité du matériel aux utilisateurs et fournir une interface entre l'utilisateur et les composants électroniques de la machine réelle.
- Cette interface est plus simple et facile à utiliser que le matériel. Elle peut être **graphique** (cas de Windows) ou un **simple interpréteur de commande** (cas d'Unix).

III. Fonctions d'un système d'exploitation
b) Gestion des ressources

- Un SE contrôle l'utilisation efficace et optimale des ressources matérielles (Processeur, Mémoire, Imprimante, etc.) et logicielles (applications utilisateurs, Word, Excel, etc.) qui peuvent être sollicitées par les différents **processus** (programme en cours d'exécution) **s'exécutant** dans le système.
- Un SE doit donc contrôler l'allocation des ressources aux différents processus qui y font appel pour s'adapter au mieux aux demandes des utilisateurs.
- En pratique chaque programme encours d'exécution reçoit :
  - une tranche de la mémoire
  - une fraction de temps de traitement processeur.

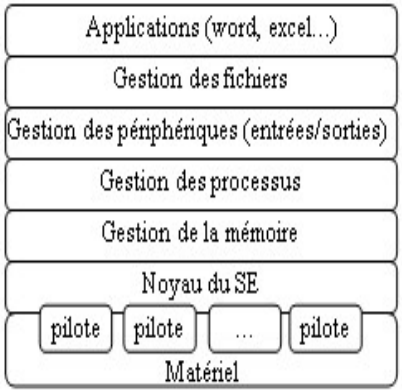
III. Fonctions d'un système d'exploitation
b) Gestion des ressources

- **Exemple :** *Gestion de l'imprimante*

Supposons que deux processus utilisateurs P1 et P2 veulent lancer simultanément l'impression de leurs résultats. Ainsi, ces deux processus vont utiliser la même ressource physique à savoir l'imprimante. Si le contrôle et la gestion de l'imprimante ne sont pas assurés par le système d'exploitation, nous risquons d'avoir les résultats du processus P1 mélangé avec ceux de P2.

IV. Rôles d'un système d'exploitation

- L'architecture d'un SE est composée d'un ensemble de couches fonctionnelles. Chaque couche repose sur les fonctionnalités offertes par la précédente mais ne voit pas les détails de la réalisation de ces fonctions.
- Les programmes utilisateurs peuvent accéder à ces différentes fonctionnalités à l'aide des **appels système**.



IV. Rôles d'un système d'exploitation
Gestion de la mémoire vive (1/2):

- Le système d'exploitation (SE) est chargé de gérer l'espace mémoire alloué à chaque application et à chaque utilisateur. En cas d'insuffisance de mémoire physique, le SE peut créer une zone mémoire sur le disque dur, appelée «**mémoire virtuelle**»

La mémoire virtuelle permet de faire fonctionner des applications nécessitant plus de mémoire qu'il n'y a de mémoire vive disponible sur le système. En contrepartie cette mémoire est beaucoup plus lente.

IV. Rôles d'un système d'exploitation
Gestion de la mémoire vive (2/2):

- La gestion de la mémoire comporte un ensemble de routines permettant de :
  - Connaître à tout moment l'état de la mémoire, c-à-d les blocs libres et occupés, quels processus utilisent quelles parties de la mémoire.
  - Allouer de la mémoire aux processus qui en ont besoin : savoir quels processus en demandent et combien, et d'allouer la mémoire nécessaire quand elle devient disponible.
  - Restituer ou libérer la mémoire chaque fois qu'un processus se termine.
  - Traiter le va et vient (lecture et écriture) entre le disque et la mémoire centrale lorsque le système met en œuvre la mémoire virtuelle.

IV. Rôles d'un système d'exploitation
Gestion du processeur :

- Le système d'exploitation est chargé de gérer l'allocation du processeur entre les différents programmes grâce à un algorithme d'ordonnancement. Le type d'ordonnanceur est totalement dépendant du système d'exploitation, en fonction de l'objectif visé.
- Un processus est un programme en cours d'exécution = un programme actif en mémoire centrale.
- Le SE est responsable de:
  - allocation de ressources aux processus
  - création, terminaison des processus
  - suspension, reprise des processus.
  - synchronisation, communication entre processus.

IV. Rôles d'un système d'exploitation
Gestion des périphériques (entrées/sorties) :

- Le SE permet d'unifier et de contrôler l'accès des programmes aux ressources matérielles par l'intermédiaire des **pilotes** (appelés également gestionnaires de périphériques).
- Cette gestion des périphériques fournit aux couches supérieures une abstraction des E/S physiques en associant à chaque type de périphérique connecté à la machine un programme particulier appelé souvent **pilote**.
- Le travail du pilote consiste à transformer l'entrée/sortie abstraite en une séquence de commandes destinée à être exécutée par le contrôleur du périphérique.

IV. Rôles d'un système d'exploitation
Gestion des fichiers :

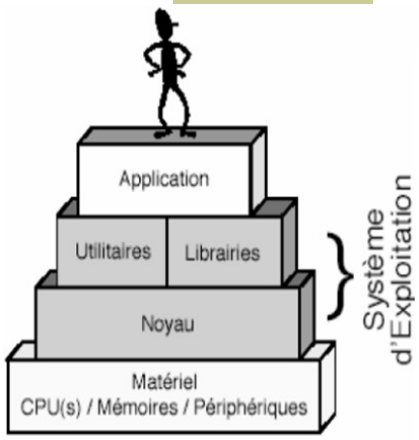
- Le système de fichiers gère la lecture et l'écriture des fichiers dans les mémoires auxiliaires (disque dur, clef USB, etc.) et les droits d'accès aux fichiers par les utilisateurs et les applications.
- Le système de fichiers est une collection de routines mise à la disposition de l'utilisateur, et qui permettent :
  - Identification des fichiers par des noms symboliques.
  - Création et la destruction de fichiers.
  - Accès efficaces et adaptées aux différents types d'applications (accès séquentiel, direct séquentiel indexé, etc.).
  - Partage des fichiers.
  - Distinction entre protection en lecture, écriture, …, etc.
  - Manipulation des fichiers : concaténation, reproduction , etc.

IV. Rôles d'un système d'exploitation

- **Gestion de l'exécution des applications :** le système d'exploitation est chargé de la bonne exécution des applications en leur affectant les ressources nécessaires à leur bon fonctionnement. Il permet à ce titre de «tuer» une application ne répondant plus correctement.
- **Gestion des droits :** le système d'exploitation est chargé de la sécurité liée à l'exécution des programmes en garantissant que les ressources ne sont utilisées que par les programmes et utilisateurs possédant les droits adéquats.

V. Les composantes d'un système d'exploitation

- Un système d'exploitation est typiquement composé de :
  - un noyau
  - bibliothèques dynamiques (*librairies*)
  - un ensemble d'outils système (utilitaires – *shell*)
  - programmes applicatifs de base



V. Les composants d'un système d'exploitation

- **Le noyau :** le noyau (en anglais **kernel**) représente les fonctions fondamentales du système d'exploitation telles que : la gestion de la mémoire, des processus, des fichiers, des entrées-sorties principales, et des fonctionnalités de communication.
- **L'interpréteur de commande :** l'interpréteur de commande (en anglais shell) est l'interface entre l'utilisateur et le système d'exploitation. Il permet la communication avec le SE par l'intermédiaire d'un **langage de commandes**.
- Le rôle de l'interpréteur de commande consiste à lire la commande, interpréter sa signification, exécuter la commande, puis retourner le résultat sur les sorties.

V. Les composants d'un système d'exploitation

- **Les bibliothèques dynamiques (Libraires) :** regroupent les opérations souvent utilisées, selon les fonctionnalités (E/S, fichier, …). Ces opérations sont disponibles pour être appelées et exécutées par d'autres programmes.
- **Le système de fichiers (en anglais «file system», noté FS) :** permet d'enregistrer les fichiers dans une arborescence.

VI. Histoire des systèmes d'exploitation

- Différents systèmes d'exploitation existent :

VI. Histoire des systèmes d'exploitation

- **Unix** : Créé en 1969, rapidement **multi-utilisateur**, écrit en langage C. Quelques versions ou distributions : AIX, OpenBSD, FreeBSD
- **Linux** : Clone gratuit d'UNIX pour les PC, *open source*. Quelques distributions parmi les plus connues : Ubuntu, Red-Hat, Mandrake / Mandriva, Debian
- **Mac OS** : Premier à proposer le concept des fenêtres, du glisser-déposer, la corbeille, le plug-and-play. Aujourd'hui, il possède le *noyau* Linux, avec une interface graphique élégante et ergonomique, et optimisation particulière des traitement multimédia.
- **MS-DOS** (Microsoft disque operating system) : SE des premiers PC, **mono-utilisateur, mono-tâche**, interface ligne de commande.
- **MS-Windows :** Inspiré par l'interface Macintosh; tout d'abord, une *coquille* graphique pour DOS. Seulement à partir de Windows 95 nous commençons à assister à un transfert de nombreuses fonctionnalités de DOS vers Windows. Quelques versions les plus connues :2.0, 3.1, 3.11, 95, NT4 et NT4 serveur, 98, 98se, Me, 2000 et 2000 serveur, XP, 2003 serveur, Vista,seven,8,10
- **Windows NT :** SE indépendant de DOS. Techniquement nettement supérieur à Windows.

VI. Histoire des systèmes d'exploitation

- L'histoire des SE peut être classée en cinq générations :
  - La première génération : le traitement par lots.
  - La deuxième génération : la multiprogrammation (multitâches).
  - La troisième génération : le temps partagé.
  - La quatrième génération : le temps réel.
  - La cinquième génération : les systèmes distribués

VII.Types des systèmes d'exploitation

1. Systèmes multitâches
2. Systèmes multiprocesseurs
3. Systèmes embarqués
4. Systèmes temps réel

VII.Types des systèmes d'exploitation

- Lorsqu'un programme qui a été traduit en instructions machines s'exécute, le processeur central lui fournit toutes ses ressources (registres internes, place en mémoire centrale, données, code,…), cet ensemble de ressources mises à disposition d'un programme est nommé son **contexte d'exécution**.

VII.Types des systèmes d'exploitation

- **Processus :** est l'image en mémoire centrale d'un programme s'exécutant avec son contexte d'exécution.
  - Un processus est un programme encours d'exécution et qui possède son propre espace mémoire, ses registres, ses piles, ses variables, etc.
  - Un processus peut être démarré par un utilisateur par l'intermédiaire d'un périphérique ou par un autre processus.
  - Les «applications» utilisateur, encours d'exécution, sont des ensembles de processus.
  - Un processus peut avoir différents états : **créé, détruit, encours d'exécution** (il a le contrôle du processeur central et exécute ses actions), **bloqué** (il est en attente d'une information), **passif** (il n'a plus le contrôle du processeur central).

VII.Types des systèmes d'exploitation

a) SE de 1ère génération : le traitement par lots

- **Un traitement par lots** : (*batch processing* en anglais) est un enchaînement automatique d'une suite de commandes (processus) sur un ordinateur sans intervention d'un opérateur.
- Une fois que ce processus est terminé (quel que soit le résultat), l'ordinateur traite le lot suivant. Le traitement des lots se termine une fois que tous les lots de la pile ont été exécutés.

VII. Types des systèmes d'exploitation

b) SE de 2ème génération : Systèmes multitâches

- Un système d'exploitation est dit «**multi-tâches**» (en anglais **multithreaded**) lorsque plusieurs «**tâches**» (également appelées processus) peuvent être exécutées simultanément.
- Les applications sont composées en séquence d'instructions que l'on appelle «processus légers» (en anglais «threads»). Ces threads seront tour à tour actifs, en attente, suspendus ou détruits, suivant la priorité qui leur est associée ou bien exécutés séquentiellement.
- Un système est dit **préemptif** lorsqu'il possède un **ordonnanceur** (aussi appelé planificateur), qui répartit, selon des critères de priorité, le temps machine entre les différents processus qui en font la demande.

VII. Types des systèmes d'exploitation

c) SE de 3ème génération : Systèmes à temps partagé

- Le système est dit à **temps partagé** lorsqu'un quota de temps est alloué à chaque processus par l'ordonnanceur. C'est le cas des systèmes **multi-utilisateurs** qui permettent à plusieurs utilisateurs d'utiliser simultanément sur une même machine des applications différentes ou bien similaires : le système est alors dit «**système transactionnel**».

VII. Types des systèmes d'exploitation

Systèmes multiprocesseurs

- **Systèmes multiprocesseurs :** le **multiprocessing** est une technique consistant à faire fonctionner plusieurs processeurs en parallèle afin d'obtenir une puissance de calcul plus importante que celle obtenue avec un processeur haut de gamme ou bien afin d'augmenter la disponibilité du système (en cas de panne d'un processeur).
- On appelle **SMP** (*Symmetric Multiprocessing* ou *Symmetric Multiprocessor*) une architecture dans laquelle tous les processeurs accèdent à un espace **mémoire partagé**.
- Un système multiprocesseurs doit donc être capable de gérer le partage de la mémoire entre plusieurs processeurs mais également de distribuer la charge de travail.

VII. Types des systèmes d'exploitation

Systèmes multicœurs

- **Systèmes multicœurs :** un processeur multicœur est un processeur doté de plusieurs cœurs qui peuvent lire et exécuter individuellement des instructions de programme, donnant l'impression que le système informatique a plusieurs processeurs.
- Les instructions peuvent être des calculs, des instructions de transfert de données, des instructions de branchement, etc.
- Le processeur peut exécuter des instructions sur des cœurs séparés en même temps. Cela augmente la vitesse globale d'exécution du programme dans le système. Ainsi, la chaleur générée par le processeur est réduite et augmente la vitesse globale d'exécution.
- Les systèmes multicœurs prennent en charge le multithreading et le calcul parallèle. Ils sont largement utilisés dans de nombreux domaines d'applications : réseau, traitement du signal numérique (DSP) et graphiques (GPU).

VII. Types des systèmes d'exploitation
Systèmes multicœurs

- **Avantages :**
  - Ces systèmes sont économes en énergie car ils permettent des performances plus élevées à moindre consommation d'énergie.
  - Il aura moins de trafic (cœurs intégrés dans une seule puce et nécessitera moins de temps).
- **Désavantages :**
  - Le processeur double cœur ne fonctionne pas à deux fois la vitesse d'un processeur unique. Ils n'obtiennent que 60 à 80 % de vitesse en plus.
  - Certains systèmes d'exploitation utilisent encore un processeur monocœur.
  - Le système d'exploitation compilé pour un processeur multicœur fonctionnera légèrement plus lentement sur un processeur monocœur.

VII. Types des systèmes d'exploitation
SE de 4ème génération : Systèmes temps réels

- Les **systèmes temps réel** (*real time systems*), essentiellement utilisés dans l'industrie, sont des systèmes dont l'objectif est de fonctionner de manière fiable selon des contraintes temporelles spécifiques, c-à-d qu'il doit être capable de délivrer un traitement correct des informations reçues à des intervalles de temps bien définis (réguliers ou non).
- **Exemples de SE temps réel :** OS-9 , RTLinux (RealTime Linux) , QNX , VxWorks.

VII. Types des systèmes d'exploitation
Systèmes embarqués

- Les **systèmes embarqués** sont des systèmes d'exploitation prévus pour fonctionner sur des machines de petite taille, telles que des PDA (*personal digital assistants*) ou des appareils électroniques autonomes (sondes spatiales, robot, ordinateur de bord de véhicule, etc.), possédant une autonomie réduite. Ainsi, une caractéristique essentielle des systèmes embarqués est leur gestion avancée de l'énergie et leur capacité à fonctionner avec des ressources limitées.
- **Exemples de SE embarqués :** PalmOS, Windows CE, Windows Mobile, Window Smartphone

VII. Types des systèmes d'exploitation

Système	Codage	Mono-utilisateur	Multi-utilisateur	Mono-tâche	Multitâche
DOS	16 bits	X		X	
Windows3.1	16/32 bits	X			non préemptif
Windows95/98/Me	32 bits	X			coopératif
WindowsNT/2000	32 bits		X		préemptif
WindowsXP	32/64 bits		X		préemptif
Unix / Linux	32/64 bits		X		préemptif
MAC/OS X	32 bits		X		préemptif
VMS	32 bits		X		préemptif

PLAN

- Chapitre 1: Notions fondamentales des Systèmes d'exploitation
- Chapitre 2: **Gestion des fichiers sous Linux**
- Chapitre 3: Les droits d'accès
- Chapitre 4: Les filtres
- Chapitre 5: Les scripts Shell

Chapitre 2

Gestion des fichiers sous Linux

Qu'est ce qu'un fichier ?

- **Nécessité des fichiers :** plusieurs applications ont besoin de stocker un grand nombre d'informations de façon **persistante** (**non volatile**) et de les rendre **accessibles** à tout moment.
- **Problème:** espace limité ou mémoire volatile (mémoire vive/registres) : un processus peut enregistrer une quantité limitée d'information dans son propre espace d'adressage (virtuel).

Qu'est ce qu'un fichier ?

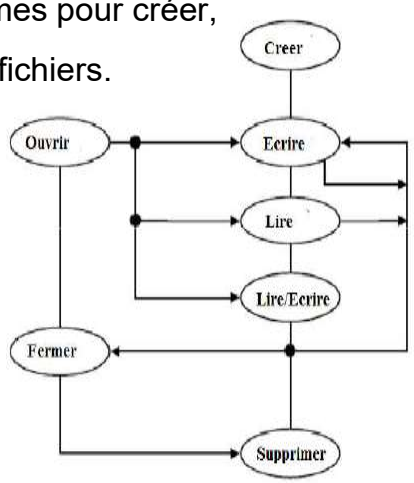
- **Définition:**

Un **fichier** est une suite d'octets (ou suite d'enregistrements , etc.) identifié par un nom auquel on associe un emplacement sur le disque ou la mémoire auxiliaire (une référence) et possède un ensemble de propriétés appelés aussi des attributs. Ainsi, **un fichier est une unité "logique" de stockage d'information**.

  - **Attributs des fichiers :** nom, taille, type, propriétaire, date, ..., etc.
  - **Opérations sur les fichiers :** création, écriture/lecture, suppression, concaténation.

Cycle de vie d'un fichier

- Le SE fournit des appels systèmes pour créer, écrire, lire, ouvrir et fermer des fichiers.
- Les fichiers ont un cycle de vie. Ils :
  - sont crées ou **ouverts**
  - peuvent être **modifiés** (écrire dedans)
  - peuvent être **lu**
  - et peuvent être **effacés**.



Deux visions d'un système de fichiers

- **Point de vue de l'utilisateur :**
  - nommage des fichiers
  - protection et droit d'accès
  - opération autorisées, etc.
- **Point de vue de l'implantation :**
  - organisation physique d'un fichier sur un disque
  - gestion des blocs et manipulation des blocs physiques attribués à un fichier
  - gestion de l'espace libre du disque.

Fonctionnalités d'un système de fichiers

Un système de fichiers est l'ensemble des fonctionnalités, dans un système d'exploitation, mises en œuvre pour la gestion des fichiers.

- **Gestion des fichiers** comporte :
  - correspondance entre fichiers et dispositifs physiques (disques, mémoires flash...)
  - la gestion de l'espace libre sur le disque dur.
  - l'organisation interne et externe des fichiers.
  - la gestion des requêtes pour l'accès aux fichiers.
  - la protection des fichiers.

Manipulation des fichiers avec Linux

- Une distribution Linux est un noyau auquel des logiciels ont été ajoutés. Les distributions peuvent être dédiées à un usage particulier.

Manipulation des fichiers avec Linux

- Le noyau Linux gère les tâches de base du système :
  - L'initialisation du système
  - La gestion des ressources
  - La gestion des processus
  - La gestion des fichiers
  - La gestion des Entrées/Sorties
- L'utilisateur communique avec le noyau par l'intermédiaire d'un SHELL.
- **Le shell :** la couche logicielle qui constitue l'interface utilisateur des systèmes d'exploitation UNIX/LINUX.
- Très souvent le mot « **shell** » fait référence à une interface en ligne de commande. **Les Shells sont aussi des langages de commandes et de programmation**.

Manipulation des fichiers avec le langage de commande shell

- **Exemples de shells :**
  - **sh (Bourne Shell) :** était le shell UNIX par défaut, beaucoup de shells modernes sont compatibles.
  - **bash (Bourne Again Shell) :** le shell par défaut sous la plupart de distributions Linux et MacOS.
  - **zsh (ZShell) :** un shell avec des fonctionnalités avancées
- **Remarque :** le **cmd** est la ligne de commande Windows. Offre des fonctionnalités pareils aux autres shells, mais est moins utilisé.

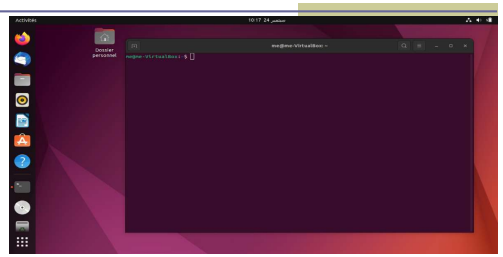
Manipulation des fichiers avec le langage de commande shell

- Lors de l'ouverture d'une session, le **shell** exécute des fichiers de configuration, qui peuvent contenir des commandes quelconques et sont généralement utilisées pour définir des variables d'environnement et des alias.
  - **sh** exécute le fichier **~/.profile**
  - **bash** exécute le fichier **~/.bash_profile** ou à défaut le fichier **~/.profile**
  - **csh** exécute le fichier **~/.cshrc**
  - **tcsh** exécute le fichier **~/.cshrc**

Manipulation des fichiers avec le langage de commande shell

- **Ouvrir un terminal :**

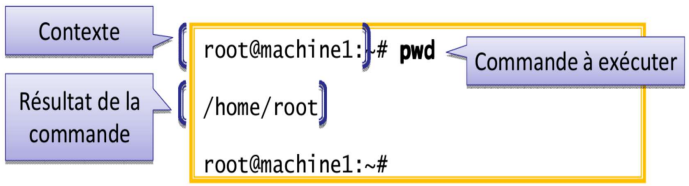
Ctrl + Alt + T



- **Syntaxe d'une commande Shell :**

commande [-options] [arguments]

- **Exemple :**

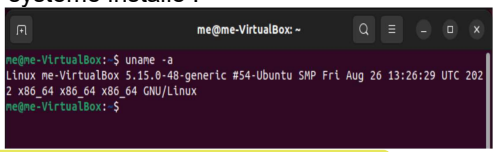


Manipulation des fichiers avec le langage de commande shell

Quelques commandes des informations système et d'aide (man)

- Commande **uname** pour afficher le système installé :

  uname –a → affiche



- Quelle est la distribution installée ?

  cat /etc/issue → affiche : « Ubuntu 22.04.1 LTS»

- Quels sont les utilisateurs en cours du système ? **who**
- Ouvrir un nouveau terminal avec un autre compte. **su nomUser**
- Fermer ce terminal. **exit**
- Obtenir de l'aide sur une commande : la commande **man**.
  - Exemple d'aide sur la commande man. **man man**

Types de fichiers

1. **Fichiers ordinaires :** contiennent les informations des utilisateurs
   - **ASCII : contiennent du texte ASCII pur** et peuvent être imprimés tels quels et édité avec n'importe quel éditeur.
   - **Binaire :** possède une structure interne propre aux programmes qui les exploitent.
     - **Exemple (UNIX) :** le SE exécutera le fichier seulement s'il possède un certain format composé de 5 parties : le *header*, le code, les données, les bits des translation, la table de symboles. Le début du *header* est un **nombre magique** qui identifie le fichier comme exécutable.
2. **Répertoires :** fichiers système qui conservent la structure du système de fichiers.
3. **Fichiers spéciaux caractère :** liés aux E/S série.
4. **Fichiers spéciaux blocs :** liés aux disques.

Attributs des fichiers

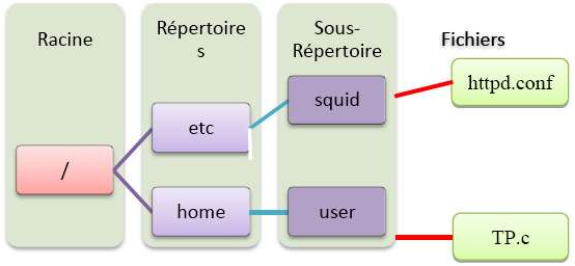
- Tous les SE associent (en plus du nom et des données) des informations complémentaires aux fichiers appelés attributs.
- Exemples d'attributs des fichiers :
  - type
  - emplacement
  - taille,
  - protection,
  - heure, date,
  - identification de l'utilisateur . . .

Arborescence

- Une arborescence est une organisation logique de fichiers sur un ou plusieurs systèmes de fichiers. Il s'agit d'une **structure de données hiérarchique** de type arbre.
- Sous Windows, il y a plusieurs racines.
  - « **C:** » est la racine du disque dur,
  - « **F:** » est la racine de votre lecteur CD (par exemple).

Arborescence
Arborescence typique d'un système Linux

- Sous Linux, **il n'y a qu'une et une seule racine** : « **/** ».
  - Il n'y a pas de lettre de lecteur car Linux ne donne pas de nom aux lecteurs comme le fait Windows.
  - Au lieu de séparer chaque disque dur, lecteur CD, lecteur de carte mémoire, …, etc, Linux place tout au même endroit sous une seule racine (/).



Arborescence
Chemin relatif et chemin absolu

- Sur les SE modernes les fichiers sont organisés en une structure hiérarchique.
- L'identification d'un fichier se fait par son nom précédé
  - d'un **chemin absolu :** c-à-d par rapport à la racine de l'arborescence.
  - ou **relatif :** c-à-d par rapport au répertoire courant.
- **Exemples des chemins absolus** (les séparateurs peuvent changer selon le SE)
  - **Windows: C:\Users\Mohamed\Desktop\Docs**
  - **Linux : /home/Mohamed/courrier**

Arborescence
Chemin relatif et chemin absolu

- Le chemin relatif fonctionne conjointement avec le concept de **répertoire de travail** ou **répertoire courant**.

Tous les chemins d'accès qui ne commence pas à la racine sont relatifs au répertoire courant

- **Exemple (sur Linux):** les commandes
  - **cp /home/ahmed/SE/TP1.sh /home/ahmed/SE/TP1.sh.bak**
  - **cp TP1.sh TP1.sh.bak**

font exactement la même chose si le répertoire de travail est **/home/ahmed/SE/**

Catégories des fichiers sous linux

Il existe 4 catégories de fichiers :

- **Les fichiers normaux (-) :** comme les fichiers textes, sources de programmes(c, java…), exécutables, programmes en code binaire.
- **Les fichiers répertoires** (en Anglais directories **(d)**) : ce sont des fichiers conteneurs qui contiennent des références à d'autres fichiers. Ils permettent d'organiser les fichiers par catégories.

Catégories des fichiers sous linux

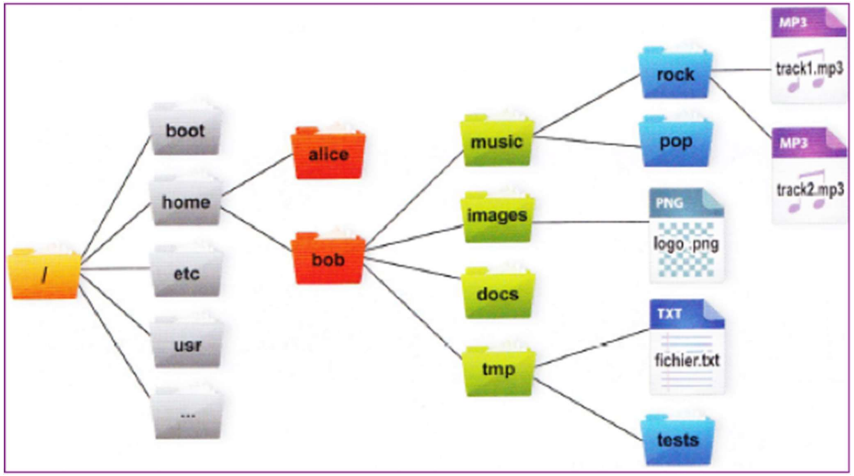
- **Les fichiers spéciaux :** situés sous **/dev**, ce sont les points d'accès préparés par le système aux périphériques. Le montage va réaliser une correspondance de ces fichiers spéciaux vers leur répertoire "point de montage". Par exemple, le fichier **/dev/hda** permet l'accès et le chargement du 1er disque.
- **Les fichiers liens symboliques (l) :** ce sont des fichiers qui ne contiennent qu'une référence (un pointeur) à un autre fichier. Cela permet d'utiliser un même fichier sous plusieurs noms sans avoir à le dupliquer sur le disque.

Les I-nodes des fichiers

- Une **i-node** est une structure de quelques dizaines d'octets qui contient la totalité des informations sur le fichier, sauf le nom :
  - le type du fichier (fichier ordinaire, spécial, répertoire,...).
  - les droits d'accès.
  - **UID :** le propriétaire du fichier.
  - **GID :** le group auquel appartient le propriétaire.
  - la date de dernière modification, la date de création.
  - la taille du fichier en octets.
  - le nombre de liens (un lien d'un fichier est un autre nom de ce fichier)
  - l'adresse physique d'implantation sur disque.
- **Remarque :** tout fichier possède son unique i-node et les i-nodes sont tous de même taille.

Arborescence typique d'un système Linux

L'arborescence des fichiers sous Linux possèdent la forme suivante :



Arborescence typique d'un système Linux

Répertoire	Signification
/boot	contient le noyau et les fichiers de démarrage du système.
/bin	contient les exécutables des programmes basiques, i.e les principales commandes disponibles pour les utilisateurs
/dev	contient les fichiers périphériques
/etc	contient les fichiers de configurations
/home	contient les fichiers utilisateurs (vos données)
/lib	contient les librairies partagées
/swap	est l'espace utilise pour décharger la mémoire
/proc	contient des informations sur les processus en exécution
/root	contient les fichiers du super-utilisateur
/sbin	contient les exécutables des fichiers d'administration
/tmp	est de l'espace réservé pour les données temporaires
/var	contient les données fréquemment modifiées (journaux, etc.)
/local	contient ce que les utilisateurs partagent et qui n'est pas standard au système.

Arborescence typique d'un système Linux

- **etc :** répertoire contenant les fichiers de configuration. Il contient des fichiers de données ainsi que des programmes réservés pour la maintenance du système tels que :
  - le fichier **passwd** : fichier texte contenant la liste des noms utilisateurs avec les mots de passe cryptés, leur identification (uid et gid), leurs répertoires initiaux et leurs Shells.
  - le fichier **group** : fichier texte contenant la liste des groupes d'utilisateurs, leurs identificateurs et les listes d'utilisateurs par groupe.

Arborescence typique d'un système Linux

- **home :** répertoire contenant les répertoires personnels des utilisateurs. Une fois connecté sous Linux, un utilisateur se retrouve dans son « **home directory »** qui a été assigné par l'administrateur système (Ex : /home/ahmed). Tous les fichiers ou répertoires qu'il va créer, le seront sous son home directory.
- **usr :** répertoire réservé pour l'utilisation du système. Il est essentiellement utilisé comme répertoire racine pour de nombreux sous répertoires, notamment pour représenter les répertoires personnels des utilisateurs.

Arborescence typique d'un système Linux

- **var :** répertoire contenant les journaux systèmes. Exemple : /var/spool/ est le répertoire contenant les files d'attente pour les sous-systèmes du courrier, de l'impression.
- **dev :** répertoire contient des fichiers spéciaux qui assurent le contrôle des accès aux différents dispositifs d'entrée/sortie. Ces fichiers ne doivent pas être détruits pour ne pas rendre impossible l'accès aux unités correspondantes.
- **proc :** pseudo-système contenant des informations sur les processus en exécution.

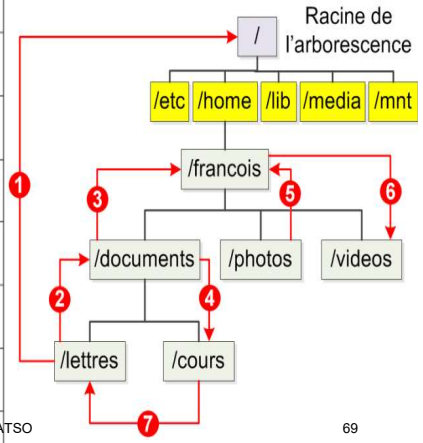
Arborescence typique d'un système Linux

- Sous le système Linux, plusieurs symboles sont utilisés pour désigner les répertoires :
  - « **.** » : désigne le répertoire courant.
    - La commande « **pwd** » affiche le nom du répertoire courant.
  - « **..** » : désigne le répertoire parent du courant
    - la commande « **cd ..** » se positionner au répertoire parent.
  - « **~** » : désigne le répertoire personnel de l'utilisateur.
    - La commande « **cd ~** » permet de se positionner au répertoire personnel (home).

Utilisation des chemins absolu et relatif

Utilisation de l'adressage absolu et relatif		
	Adressage absolu	**Adressage relatif**
1	cd /	cd ../../../..
2	cd /home/francois/documents	cd ..
3	cd /home/francois	cd ..
4	cd /home/francois/documents/cours	cd cours
5	cd /home/francois	cd ..
6	cd /home/francois/videos	cd videos
7	cd /home/francois/documents/lettres	cd ../lettres

La commande **cd** permet d'entrer dans un répertoire



Commandes shell de manipulation des fichiers

Commande	Signification
pwd	afficher le nom du répertoire courant
mkdir	créer un nouveau répertoire
rmdir	supprimer un répertoire
ls	Lister le contenu du répertoire
cd	changer de répertoire (entrer dans un répertoire)
mv	déplacer un fichier
cp	copier un fichier
rm	supprimer un fichier
find	rechercher un chier
grep	afficher les lignes des fichiers contenant une chaîne donnée de caractères
ps	afficher la liste des processus de l'utilisateur
chmod	changer les droits d'un fichier
passwd	créer ou changer de mot de passe

Commandes shell de manipulations de fichiers
Commandes de manipulation des répertoires

- **pwd :** afficher le répertoire courant
- **cd :** entrer dans un répertoire
- **mkdir :** créer un nouveau répertoire
- **rmdir :** supprimer un répertoire déjà existant
- **ls :** lister le contenu d'un répertoire

Commandes shell de manipulations de fichiers
La commande pwd

- **Définition : pwd (print working directory)**

  cette commande permet d'afficher le chemin d'accès au répertoire courant.
- **Syntaxe :** `pwd`
- **Exemple :**

pwd
→ **affiche : « /home/nomUser/» si l'utilisateur est dans son espace personnel**

Commandes shell de manipulations de fichiers
La commande cd

- **Définition :** la commande **cd** **(change directory)** permet de changer le répertoire courant.
- **Syntaxe :** **cd [répertoire]**
- **Exemples :**
  - « **cd ~** » et « **cd** » ramènent dans le répertoire de connexion (home).
  - « **cd .** » : ne change pas le répertoire courant.
  - « **cd ..** » : permet de se déplacer vers le répertoire parent.
  - « **cd /** » : permet de se déplacer vers la racine.
  - « **cd rep1**» : permet d'entrer dans le sous-répertoire rep1.

Commandes shell de manipulation des fichiers
La commande mkdir (make directory)

- **Définition :** cette commande permet de créer un répertoire
- **Syntaxe :** **mkdir [-options] répertoire**

  Le chemin peut être :
  - **relatif** (par exemple **mkdir ../issatso**)
  - **absolu** (par exemple **mkdir /home/Ahmed/issatso/cours**)
- **Options :**
  - **mkdir –p :** permet de créer une suite de répertoires.
  - **mkdir -v** : retourner des informations lors de la création d'un répertoire.
- **Exemple :** **mkdir –p cours/tp/SE**

  → crée le sous-répertoire **cours** ensuite le sous-répertoire **tp** (qui se trouve dans cours) ensuite le sous-répertoire **SE** (dans le répertoire tp).

Commandes shell de manipulation des fichiers
La commande rmdir (remove directory)

- **Définition :** cette commande permet de supprimer un répertoire vide.
- **Syntaxe :** **rmdir [-options] répertoire(s)**
- **Options :**
  - **-i :** cette option permet d'afficher un message demandant à l'utilisateur s'il souhaite vraiment supprimer le ou les répertoires en question.
- **Exemple :** **rmdir -i /home/Ahmed/TDs**

Commandes shell de manipulations de fichiers
La commande ls

- **Définition :** la commande **ls** **(list files)** permet de lister la description des fichiers ou répertoires précisés.
- **Syntaxe :** **ls [-options] chemin**
  - Le chemin est un nom de fichier ou de répertoire.
    - Si c'est un fichier, **ls** permet d'afficher sa description.
    - Si c'est un répertoire, elle affiche son contenu.
    - Sans arguments, le répertoire courant est traité.

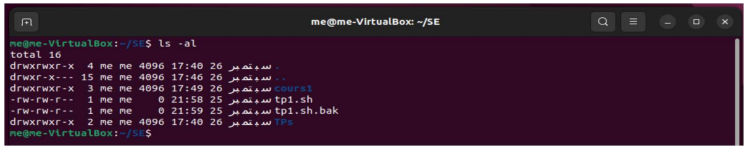
Commandes shell de manipulation des fichiers
La commande ls

- Les options de la commande **ls** sont :
  - **-l** : format long. Affiche les infos du types de fichier, droits, nombre de liens, propriétaire, groupe, taille en octets, date de la dernière modification, nom du fichier.
  - **-a** : liste tous les fichiers (y compris les fichiers qui commencent par le point : fichiers cachés).
  - **-F** : format court avec indication du type de fichier (ajoute * si exécutable, / si répertoire).
  - **-i** : affiche les numéros d'Inode des fichiers.
  - **-R** : récursif, génère la liste de tous les fichiers du sous-arbre tout entier.
  - **-r** : trie en ordre inverse.
  - **-d** : si l'argument est un répertoire, la commande affiche son nom.

Commandes shell de manipulation des fichiers
La commande ls

- **Exemples :** les options de la commande **ls**
  - **ls –a :**

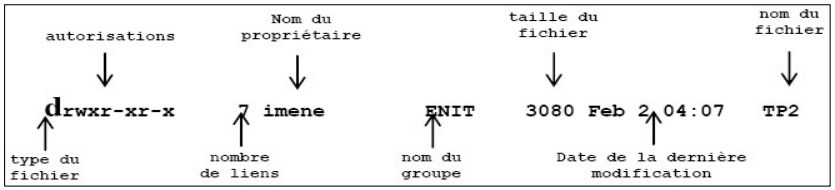
  - **ls -al :**



  - **ls -i :**

Commandes shell de manipulation des fichiers
La commande ls

- **Exemples :**
  - La commande **ls –l** donne un tel résultat :



  - "**ls -al**" affiche tous les fichiers, de façon détaillée, dans l'ordre chronologique, an ajoutant '/' après chaque nom de répertoire

```
[jice@taz jice]$ls -al
total 144
-rw-r--r--   1 jice  users     24 Aug  2 21:37 .bash_logout
-rw-r--r--   1 jice  users    230 Aug  2 21:37 .bash_profile
-rw-r--r--   1 jice  users    467 Aug  2 21:37 .bashrc
-rw-r--r--   1 jice  users   1452 Aug  2 21:37 .kderc
drwxr--r--  12 jice  users   1024 Aug  2 21:37 .kde/
drwxr--r--   2 jice  users   1024 Aug  2 21:37 Desktop/
-rw-r-----   1 jice  users   1728 Aug  2 21:37 adresses.txt
-rw-------   1 jice  users    144 Aug  2 21:37 motsdepasse.txt
lrwxrwxrwx   1 jice  users     14 Aug  2 21:37 linux -> /usr/src/linux
```

Commandes shell de manipulation des fichiers
Les méta caractères

- **Définition :** les **méta caractères**, sont des caractères interprétés spécialement par le Shell. Par exemple,
  - « * » désigne une chaîne de caractères quelconque (même une chaîne vide).
  - « ? » désigne un seul caractère quelconque.
  - « [ ] » désigne un des caractères mentionnés entre les crochets ou un intervalle.
- **Exemple :** si l'on entre **ls *.c**, le Shell remplace l'argument *.c par la liste des fichiers du répertoire courant dont le nom se termine par **.c**.

➔ **Les métacaractères permettent donc de spécifier facilement des ensembles de fichiers, sans avoir à rentrer tous leurs noms.**

Commandes shell de manipulation des fichiers
Les méta caractères

- **Exemples :**
  - **[a-z]** → pour désigner l'ensemble des lettres minuscules.
  - **ls *.c** → lister tous les fichiers dont le nom se termine par .c
  - **ls [aei]*** → lister tous les fichiers dont le nom commence par « **a** » ou bien « **e** » ou « **i** ».
  - **ls [!0-9]*** → lister tous les fichiers dont le nom ne commence pas par un chiffre.
  - **ls [0-9a-zA-Z]*** → lister tous les fichiers dont le nom commence par un caractère alphanumérique.

Commandes shell de manipulation des fichiers
Les méta caractères

- **Exercice 1 :** lister tous les fichiers :
  a) à partir de votre espace personnel, qui représentent des programmes java.
  b) dans le répertoire courant, dont les noms commencent par 'tp'.
  c) à partir de la racine, dont les noms commencent par 'tp' et de 4 lettres maximum.
  d) dont les noms commencent par 'annee' avec aucun chiffre numérique.
  d) commençant par 'b' ou 'B'.
  e) dont l'avant-dernier caractère est un '5' ou '1' dans le répertoire /tmp en une seule commande.

Commandes shell de manipulation des fichiers
Les méta caractères

- **Exercice 1 (Correction) :** lister tous les fichiers :
  a) à partir de votre espace personnel, qui représentent des programmes java → **ls –R ~/*.java**
  b) dans le répertoire courant, dont les noms commencent par 'tp' → **ls ./tp***
  c) à partir de la racine, dont les noms commencent par 'tp' et de 4 lettres maximum → **ls –R /tp??**
  d) dont les noms commencent par 'annee' suivi d'un caractère non numérique, → **ls annee[!0-9]**
  d) commençant par 'b' ou 'B' → **ls [aA]***
  e) dont l'avant-dernier caractère est un '5' ou '1' dans le répertoire /tmp en une seule commande. → **ls –R /tmp/*[51]?**

Commandes shell de manipulations de fichiers
Commandes de manipulation des fichiers et des répertoires

- **mv :** renommer un fichier ou répertoire
- **cp :** copier un fichier ou un répertoire
- **rm :** supprimer un fichier ou un répertoire
- **touch :** créer un fichier vide
- **cat:** afficher à l'écran le contenu d'un fichier
- **more :** afficher à l'écran le contenu d'un fichier page par page.
- **file :** afficher le type du ficher.
- **ln :** créer un lien vers un fichier ou répertoire.

Commandes shell de manipulation des fichiers
La commande mv (move)

- **Définition :** la commande **mv** permet de déplacer un fichier ou un répertoire.
- **Syntaxe :** **mv [–i] source destination**
  - Si « destination » est un répertoire, la commande **mv** déplace le fichier de nom « source » vers « destination ».
  - Si « destination » est un nom de fichier, la commande **mv** renomme le fichier de nom source avec le nom destination.
  - L'option **mv -i** permet de demander confirmation en cas d'écrasement de la destination.

Commandes shell de manipulation des fichiers
La commande mv (move)

- **Exemples :**
  - **mv Ex1 Ex2.c** → renomme le fichier **Ex1** en **Ex2.c**
  - **mv Ex1 TDs/** → déplace le fichier **Ex1** du répertoire courant au sous-répertoire **TDs**.
  - **mv Ex1 TDs/Ex2.c** → déplace le fichier **Ex1** du répertoire courant au sous-répertoire **TDs** et le renomme **Ex2.c**.

Commandes shell de manipulation des fichiers
La commande cp

- **Définition :** cette commande permet de copier un fichier ou un répertoire.
- **Syntaxe:** **cp [–options] source destination**
  - Si « destination » est un répertoire, la commande copie le ou les fichier(s) source vers « destination ».
  - Si « destination » est un nom de fichier, la commande cp renomme le fichier de nom « source » avec le nom « destination».
  - Si on effectue une copie d'un fichier sur un fichier qui existe déjà, celui-ci sera écrasé et remplacé par le nouveau fichier.

Commandes shell de manipulation des fichiers
La commande cp

- Les principales options de la commande **cp** sont :
  - **-i :** avertit l'utilisateur de l'existence d'un fichier du même nom et lui demande s'il peut ou non remplacer son contenu.
  - **-b :** permet de s'assurer que la copie n'écrase pas un fichier existant : le fichier écrasé est sauvegardé, seul le nom du fichier d'origine est modifié et **cp** ajoute une tilde (**~**) à la fin du nom du fichier.
  - **-p :** permet lors de la copie de préserver toutes les informations concernant le fichier comme le propriétaire, le groupe, la date de création.
  - **-r :** permet de copier un répertoire de manière récursive (le répertoire et ses sous-répertoires).

Commandes shell de manipulation des fichiers
La commande rm (remove file)

- **Définition :** la commande **rm** est utilisée pour supprimer un fichier ou un répertoire.
- **Syntaxe :** **rm [–options] fichiers**
- Ses principales options de la commande **rm** sont :
  - **rm -i :** permet de demander à l'utilisateur s'il souhaite vraiment supprimer le ou les fichiers en question.
  - **rm -r :** agit de façon récursive, c'est à dire détruit aussi les répertoires (pleins ou vide) et leurs sous-répertoires.
  - **rm -f :** permet d'ignorer les fichiers inexistant sans afficher de messages.

Commandes shell de manipulation des fichiers
La commande touch

- **Définition :** créer des fichiers vides.
- **Syntaxe :** **touch nom_du_fichier**
- **Exemples :**
  - **touch fich1.txt** → permet de créer le fichier vide **fich1.txt** dans le répertoire courant.
  - **touch Ex1.txt TD/Ex2.c** → permet de créer deux fichiers vides : **Ex1.txt** sous le répertoire courant et **Ex2.c** dans le sous-répertoire **TD**.

Commandes shell de manipulation des fichiers
Les commandes cat et more

- **La commande cat :** permet d'afficher les fichiers l'un après l'autre sur la sortie standard (écran). Si aucun argument n'est spécifié, lit sur l'entrée standard (clavier) jusqu'à rencontrer un caractère fin de fichier CTRL^D.
  - **Exemple :**
    - **cat fich.txt :** cette commande permet d'afficher sur l'écran le contenu du fichier fich.txt
    - **cat fich1 fich2 :** permet d'afficher successivement, sur l'écran, le contenu des deux fichiers fich1 et fich2.
- **La commande more :** similaire à la commande **cat**, mais affiche le contenu d'un fichier page à page.
  - **Syntaxe : more [fichier...]**

Commandes shell de manipulation des fichiers
La commande file

- **Définition :** cette commande permet de connaître le type de fichier. Tous les fichiers ont un entête permettant de déterminer leur type (répertoire, exécutable, texte ASCII, programme C, document...). La commande **file** permet de visualiser le type du fichier en question.
- **Syntaxe :** **file nom_du_fichier**
- **Exemple :**
  - **file toto.c** → affiche **toto.c: C source, ASCII text**

Commandes shell de manipulation des fichiers
La commande ln (link)

- **Définition :** cette commande permet de réaliser un lien avec un autre fichier ou répertoire. Ce lien peut être physique ou symbolique.
  - Les liens sont des fichiers spéciaux permettant d'associer plusieurs noms (liens) à un seul et même fichier ou répertoires. Ce dispositif permet d'avoir plusieurs instances d'un même fichier en plusieurs endroits de l'arborescence sans nécessiter de copie, ce qui permet d'économiser de l'espace disque.

Commandes shell de manipulation des fichiers
La commande ln (link)

- **Les liens symboliques :** représentent des pointeurs virtuels (raccourcis) vers des fichiers réels. En cas de suppression du lien symbolique le fichier pointé n'est pas supprimé.
  - **Syntaxe :** **ln -s nom-fichier-reel nom-lien-symbolique**
- **Les liens physiques :** (ou liens durs) représentent un nom alternatif pour un fichier
  - permet de donner plusieurs noms à un fichier
  - pas pour les répertoires
  - ne traverse pas les partitions
  - un fichier est détruit quand TOUS ses liens physiques sont supprimés.
  - **Syntaxe :** **ln nom-fichier-reel nom-lien-symbolique**

Commandes shell de manipulation des fichiers
La commande ln (link)

- **Exemples :**
  - **Liens symbolique :**
    - **ln –s ../coursSE.pdf coursLienSymb**
    - **ls –l coursLienSymb**

→ affiche : **lrwxrwxrwx 1 … coursLienSymb → ../coursSE.pdf**

  - Si on supprime le fichier coursLienSymb : **rm** coursLienSymb
  - Le fichier coursLienSymb est supprimé mais le fichier ../coursSE.pdf existe encore.
  - **Liens physique :**
    - **ln ../coursSE.pdf coursLienSymb**
    - **ls –l coursLienSymb**

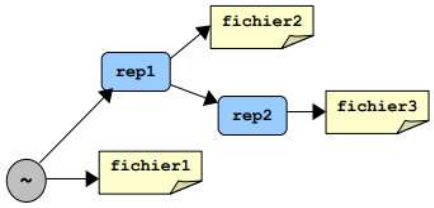
→ affiche : **-rwxrwxrwx 1 … coursLienSymb**

  - Si on supprime le fichier coursLienSymb : **rm coursLienSymb**
  - Le fichier **coursLienSymb** est supprimé et le fichier **../coursSE.pdf** peut aussi être supprimé.

Commandes shell de manipulation des fichiers
Application

- **Exercice 2 (1/2) :** lister tous les fichiers :

1. A partir de votre répertoire de travail, créer l'arborescence suivante



2. Supprimer le fichier fichier3. Afin de ne pas supprimer un fichier par erreur, afficher le message de confirmation avant la suppression.
3. Vérifier que la suppression a bien été effectuée.

Commandes shell de manipulation des fichiers
Application

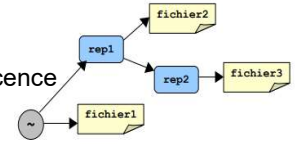
Exercice 2 (2/2) :

4. Copier le fichier fichier1 dans le répertoire rep2.
   a. Renommer le fichier fichier1 qui est dans rep2 à fichier3.
   b. Peut-on faire ces deux dernières actions en une seule commande.
5. Commet visualiser le contenu des deux fichiers fichier2 et fichier3 en une seule commande.
6. Afficher la liste des fichiers contenus dans rep1 et ses sous-répertoires.
7. Supprimer le répertoire rep1 et ses sous-répertoires.
8. Supprimer les fichiers et répertoires dont les noms commencent par la sous-chaîne « tp » ou « EX ».

Commandes shell de manipulation des fichiers
Application

- **Exercice 2 (Correction) :** lister tous les fichiers :

1. A partir de votre répertoire de travail, créer l'arborescence suivante




```
mkdir –p rep1/rep2
touch fichier1 rep1/fichier2 rep1/rep2/fichier3
```

2. Supprimer le fichier fichier3. Afin de ne pas supprimer un fichier par erreur, afficher le message de confirmation avant la suppression.

```
rm –i rep1/rep2/fichier3
```

3. Vérifier que la suppression a bien été effectuée.

```
ls -l rep1/rep2/    ou bien   file rep1/rep2/fichier3
```


Commandes shell de manipulation des fichiers
Application

4. Copier le fichier fichier1 dans le répertoire rep2. `cp fichier1 rep1/rep2/`
   a. Renommer le fichier fichier1 qui est dans rep2 à fichier3.
      `mv rep1/rep2/fichier1 rep1/rep2/fichier3`
   b. Peut-on faire ces deux dernières actions en une seule commande.
      `cp fichier1 rep1/rep2/fichier3`
5. Commet visualiser le contenu des deux fichiers fichier2 et fichier3 en une seule commande. `cat rep1/fichier1 rep1/rep2/fichier3`
6. Afficher la liste des fichiers contenus dans rep1 et ses sous-répertoires.
   `ls –R rep1`
7. Supprimer le répertoire rep1 et ses sous-répertoires. `rm –R rep1`
8. Supprimer les répertoires dont les noms commencent par la sous-chaîne « tp » ou « EX » `rm –r tp* EX*`