

Systemes d'Exploitation 1

Génie Informatique – S3

Informatique Appliquée – S3

Pr. AZDAD Nabila

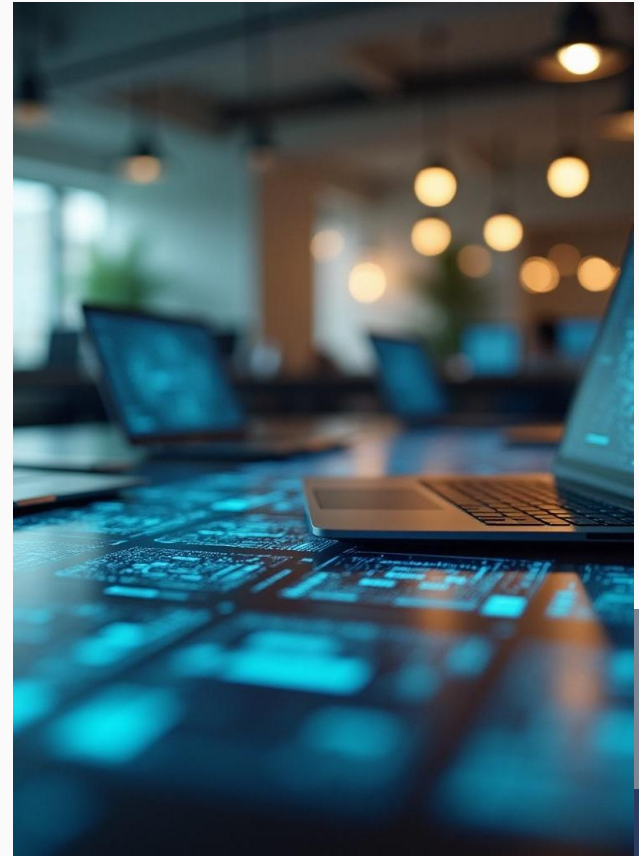
Département d'Informatique

Email: n.azdad@ump.ac.ma

Année Universitaire: 2025 - 2026

Objectifs du module

- Comprendre le rôle et les objectifs d'un système d'exploitation.
- Maîtriser l'utilisation des commandes Unix.
- Maîtriser la programmation shell



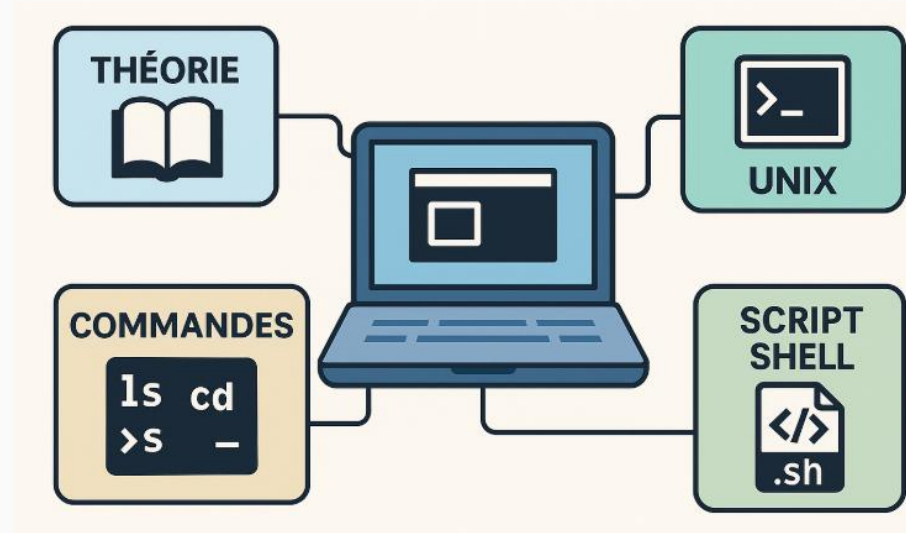
Plan

Partie 1: Introduction aux systèmes d'exploitation

Partie 2: Systèmes d'exploitation Unix

Partie 3: Les commandes de base du système Unix

Partie 4: La programmation Shell





01

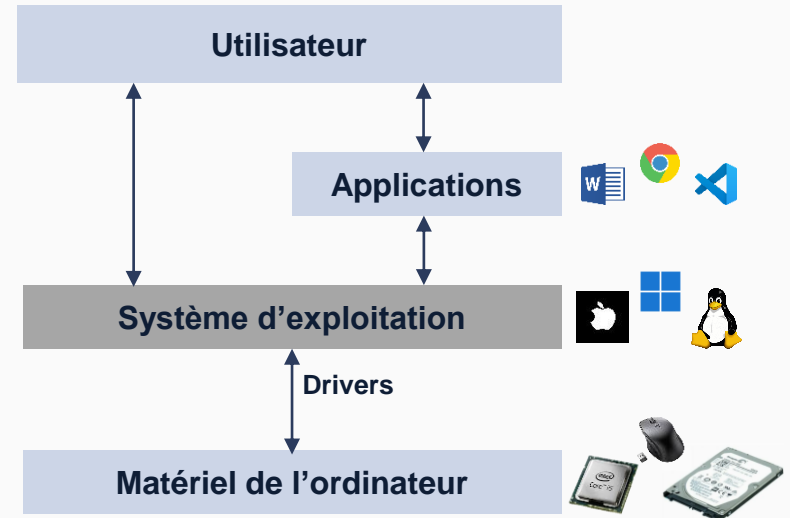
Introduction aux Systèmes d'Exploitation

Définition

- **Le système d'exploitation** (abrégé SE, en anglais Operating System ou OS) est **un ensemble de programmes** responsables de **la liaison** entre les ressources matérielles d'un ordinateur et les applications de l'utilisateur.
- Il assure le démarrage de l'ordinateur, et fournit aux programmes applicatifs des interfaces standardisées pour accéder aux périphériques.

Le **système d'exploitation** fait le lien entre
deux mondes :

- **Le matériel (hardware)** : le processeur, la mémoire, le disque dur, l'écran, le clavier, etc.
- **Les applications de l'utilisateur** : traitement de texte, jeux vidéo, navigateur web, etc.



Rôle principal du SE

- Le SE agit comme un **intermédiaire** ou un **traducteur** :
 - D'un côté, l'utilisateur lance un logiciel (ex. : Word, navigateur web...).
 - De l'autre, le matériel exécute des instructions très techniques.
 - Le SE coordonne tout cela.
 - ✓ Grâce au système d'exploitation, l'utilisateur peut utiliser ses programmes sans avoir à gérer lui-même la mémoire, le processeur ou les composants matériels.

Objectifs des systèmes d'exploitation

1. Gestion des ressources matérielles

Le SE contrôle et alloue les ressources matérielles comme le processeur, la mémoire, les périphériques d'entrée/sortie (clavier, souris, disque dur, carte graphique...), pour que plusieurs programmes puissent les utiliser sans conflit.

2. Interface utilisateur

Fournir une interface (graphique ou en ligne de commande) qui permet à l'utilisateur d'interagir facilement avec l'ordinateur.

Objectifs des systèmes d'exploitation

3. Gestion des fichiers

Organiser, stocker, récupérer, sécuriser et gérer les fichiers et dossiers sur les dispositifs de stockage.

4. Exécution des programmes

Charger, exécuter et superviser les programmes, en assurant que chaque programme a les ressources nécessaires et ne gêne pas les autres.

5. Sécurité et protection

Protéger les données et le système contre les accès non autorisés, gérer les droits d'accès des utilisateurs et des applications.

Objectifs des systèmes d'exploitation

6. Gestion des erreurs

Détecter, signaler et gérer les erreurs matérielles ou logicielles pour assurer la stabilité du système.

7. Multitâche et gestion des processus

Permettre l'exécution simultanée de plusieurs processus, gérer leur priorité, synchronisation et communication.

8. Communication entre matériels et logiciels

Servir d'intermédiaire entre les applications et le matériel, en traduisant les commandes logicielles en instructions matérielles.

Classes des SE selon les contraintes d'utilisation

Mono-utilisateur/mono-tâche (MS-DOS)

- Un seul utilisateur peut exécuter une seule tâche à la fois.

Mono-utilisateur/multi-tâches(Windows XP)

- Un seul utilisateur peut exécuter plusieurs tâches simultanément.

Multi-utilisateurs/multi-tâches(Unix)

- Plusieurs utilisateurs peuvent exécuter simultanément plusieurs tâches, en partageant les mêmes ressources matérielles du système (processeur, mémoire, périphériques, etc.).

Classes des SE selon les services

Systèmes temps réel:

- Utilisés dans des domaines spécifiques (robotique, centrales nucléaires...)
- Temps de réponse des tâches critiques court
- Fiables et tolérants aux pannes

Systèmes transactionnels:

- Gestion des bases de données énormes (systèmes de réservation, systèmes bancaires...)
- garantir des mises à jour sans confusion.

Classes des SE selon l'architecture matérielle

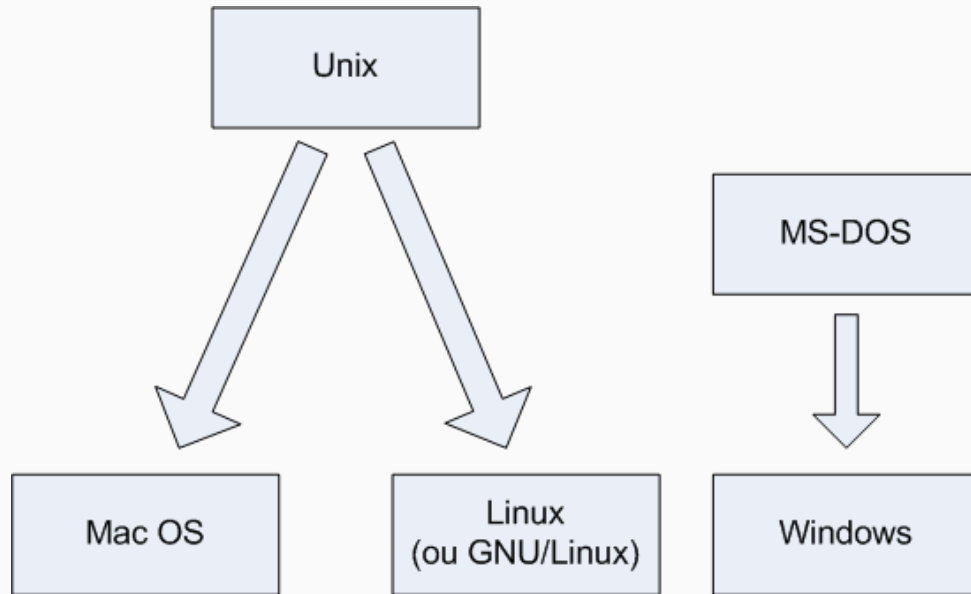
Systèmes mono-processeur:

- Le processeur ne peut exécuter qu'un programme à la fois, mais il change très rapidement d'un programme à l'autre pour donner l'impression que plusieurs programmes s'exécutent simultanément.
- Chaque programme reçoit une petite tranche de temps pour s'exécuter.

Systèmes parallèles : multiprocesseurs

- Traitement parallèle par plusieurs processeurs
- Grande capacité de traitement, temps de réponse court et fiabilité

Les principaux systèmes d'exploitation





02

Systemes d'exploitation Unix

Historique de Unix

1

Dans les années 1960 :

- Les ordinateurs étaient très coûteux, imposants et réservés aux grandes entreprises, universités ou gouvernements.
- Les systèmes d'exploitation étaient **propriétaires**, **non portables**, écrits spécifiquement pour chaque machine.

Historique de Unix

2

Création (1969)

- Unix a été créé en **1969** par **Ken Thompson** et **Dennis Ritchie**, dans les laboratoires **Bell Labs** (AT&T).
- Il était d'abord développé sur une petite machine appelée **PDP-7***.

* vieille machine qui n'était plus vraiment utilisée

3

Unix devient portable (1973)

- Réécriture complète du système Unix en **langage C**.
- Cela permet d'installer Unix sur différentes machines.
- Unix devient **portable**, ce qui était très rare à l'époque.

Historique de Unix

4

Diffusion dans les universités

- Dans les années 1970, Unix est utilisé dans plusieurs universités.
- Des chercheurs ajoutent de nouvelles fonctions.
- C'est là que naît la version **BSD** (à Berkeley).

5

Multiplication des versions (années 80)

- Plusieurs entreprises créent leur propre version d'Unix :
 - **System V, HP-UX, AIX, SunOS**, etc.
- Beaucoup de versions différentes.

Historique de Unix

6

L'arrivée de Linux (1991)

- En **1991**, **Linus Torvalds** crée **Linux**, un système inspiré d'Unix.
- Gratuit et libre, il devient très populaire.

7

Aujourd'hui

- Unix ou ses dérivés sont partout :
 - **macOS** est basé sur Unix,
 - **Android** est basé sur Linux,
 - Les serveurs Internet tournent souvent sous Unix/Linux.

Caractéristiques du système UNIX

- Unix est un système d'exploitation **multi-tâches** (*multithreaded* en anglais): plusieurs processus (process en anglais), également appelées « tâches », peuvent être exécutées simultanément.
 - ➔ A chaque instant, le processeur ne traite qu'un seul processus (programme lancé) , la gestion des processus est effectuée par le système.
- Unix est un système d'exploitation **multi-utilisateurs** (**multi-user**): plusieurs utilisateurs peuvent utiliser le système en même temps (les ressources sont réparties entre les différents utilisateurs).
 - ➔ Le système Unix se charge de contrôler et de gérer l'utilisation et l'attribution des ressources entre les différents utilisateurs.

Caractéristiques du système UNIX

- Sous Unix, du point de vue utilisateur, il n'y a pas de notion de disque physique (partition, disque externe, ...), en effet sous Unix **tout est fichier**. L'utilisateur ne voit qu'une seule arborescence de fichiers hiérarchiques .
 - Les périphériques sont aussi représentés par des fichiers, ce qui rend le système indépendant du matériel et par conséquent assure la portabilité ; l'accès aux périphériques est donc identique à l'accès aux fichiers ordinaires.
- La gestion de la mémoire virtuelle : Unix utilise un mécanisme d'échange entre la RAM et le disque dur qui permet de pallier au manque de RAM.

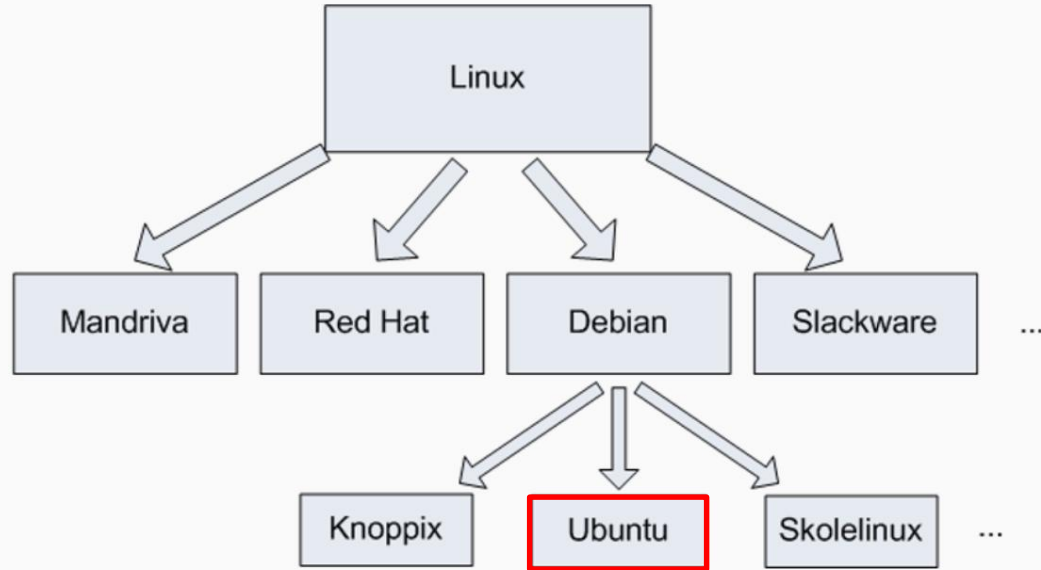
Linux

- **Linux** (parfois nommé **GNU/Linux**) est un système d'exploitation *open source* de type Unix.
- Linux a la particularité d'être **libre**, c'est-à-dire que **son code source est ouvert** : tout le monde peut le consulter, le modifier et le redistribuer.
- Par opposition, le code source qui a permis de concevoir Windows et Mac OS est **fermé**, on dit que ce sont des **systèmes d'exploitation propriétaires**.

Linux

- L'installation de Linux est simple et conviviale sur un ordinateur PC ou Mac, et il existe de nombreux tutoriels sur Internet.
- Il existe de nombreuses variantes de Linux, que l'on appelle **distributions**, à télécharger gratuitement sur Internet.

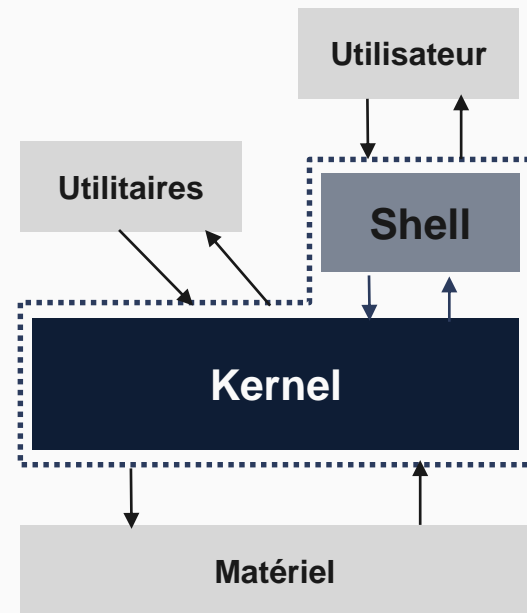
Linux



- **Ubuntu** est l'une des distributions les plus populaires à l'heure actuelle. C'est celle que nous utiliserons tout au long de ce cours. *Il s'agit d'un **dérivé** de la distribution Debian.*

Architecture d'Unix

- L'architecture d'Unix est organisée en **couches** qui séparent les responsabilités.
- Elle se compose principalement de :
 1. Matériel
 2. Noyau (Kernel)
 3. Shell
 4. Applications / Utilitaires



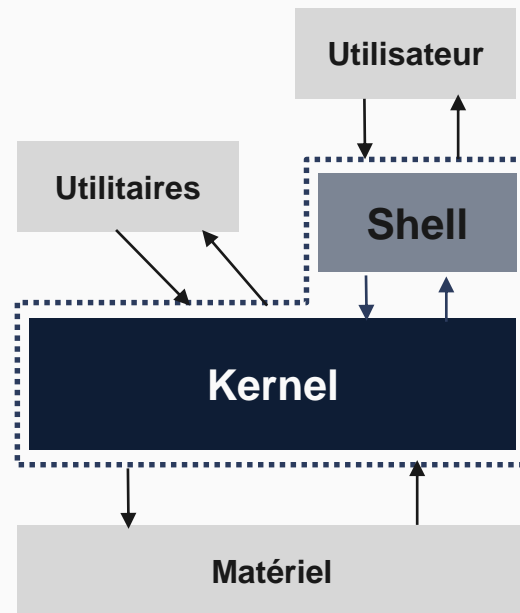
Architecture d'Unix

Le noyau (kernel)

• Un **noyau de système d'exploitation**, ou simplement **noyau**, ou **kernel** en anglais est le logiciel qui assure :

- l'interaction entre le matériel et les logiciels ;
- la gestion des divers logiciels (tâches) d'une machine (lancement des programmes, ordonnancement...) ;
- la gestion du matériel (mémoire, processeur, périphérique, stockage...).

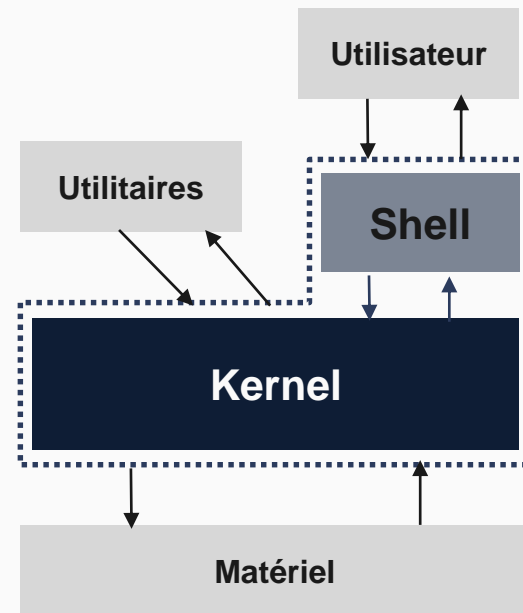
→ C'est le cœur du système.



Architecture d'Unix

Shell

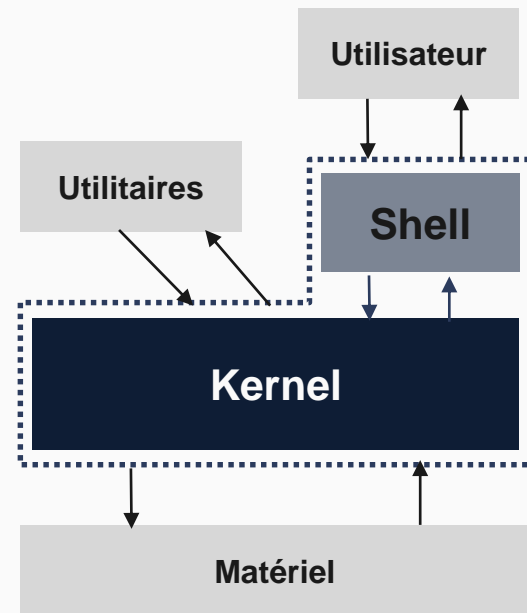
- Programme en **espace utilisateur** (*user space*).
- Sert d'**interface** entre l'utilisateur et le noyau.
- Fonction :
 - Interpréter les commandes saisies.
 - Exécuter les programmes demandés.
- Processus :
 - L'utilisateur tape une commande.
 - Le shell l'analyse.
 - Il envoie des appels système au noyau.
 - Le noyau exécute la commande.



Architecture d'Unix

Applications et utilitaires

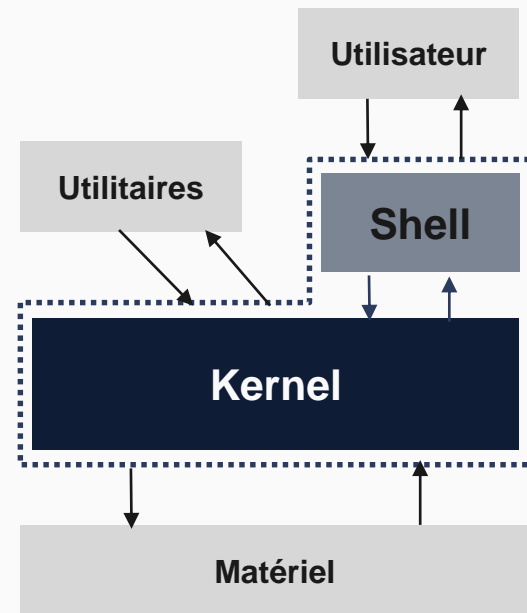
- Programmes supplémentaires fournis avec Unix :
 - Commandes (cp, mv, grep, cat...)
 - Éditeurs (vi, nano)
 - Compilateurs (gcc)
- Utilisent le shell ou une interface graphique pour interagir avec le noyau.



Architecture d'Unix

Matériel

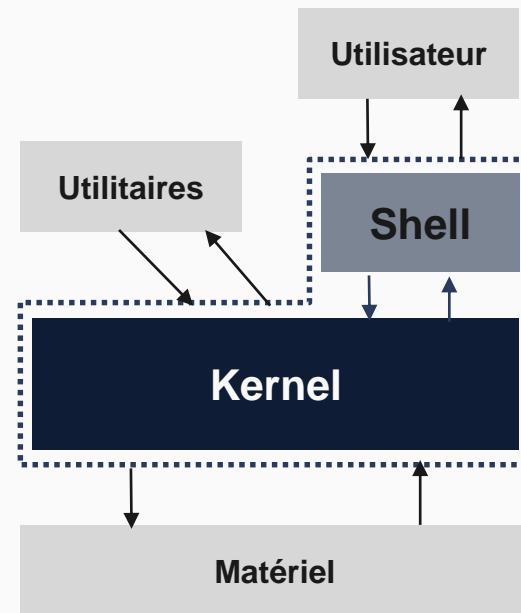
- Partie physique de l'ordinateur : processeur (CPU), mémoire (RAM), disques, périphériques (imprimante, clavier, carte réseau...).
- Le matériel **ne peut pas être utilisé directement par les programmes** → *il faut passer par le noyau.*



Architecture d'Unix

Points clés à retenir

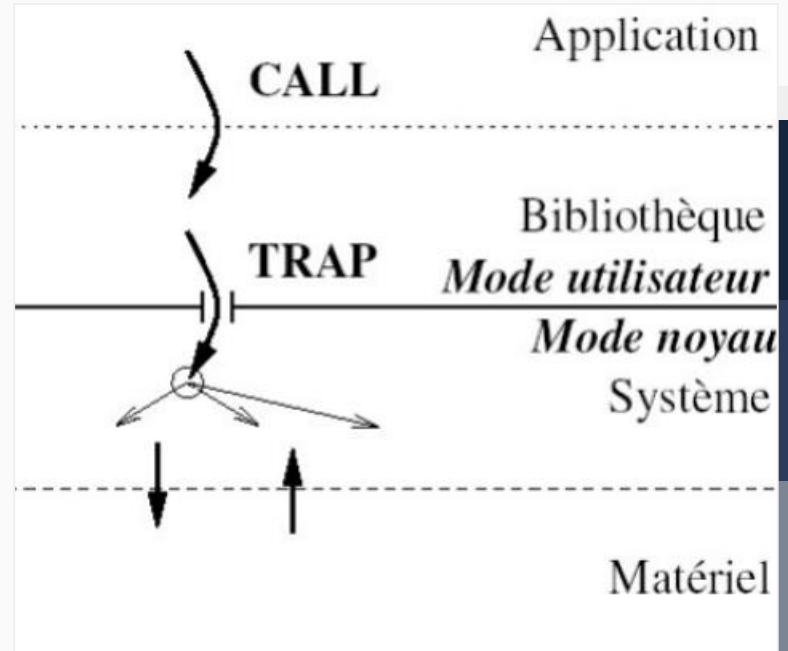
- Unix est multi-utilisateurs et multi-tâches.
- L'architecture sépare clairement :
 - **Espace noyau** : accès direct au matériel (kernel + pilotes).
 - **Espace utilisateur** : programmes, shell, applications.
- Le shell n'est pas dans le noyau, c'est un programme utilisateur.
- La communication entre espace utilisateur et espace noyau se fait via **les appels système**.



Modes d'exécution

1. Mode utilisateur (User Mode) :

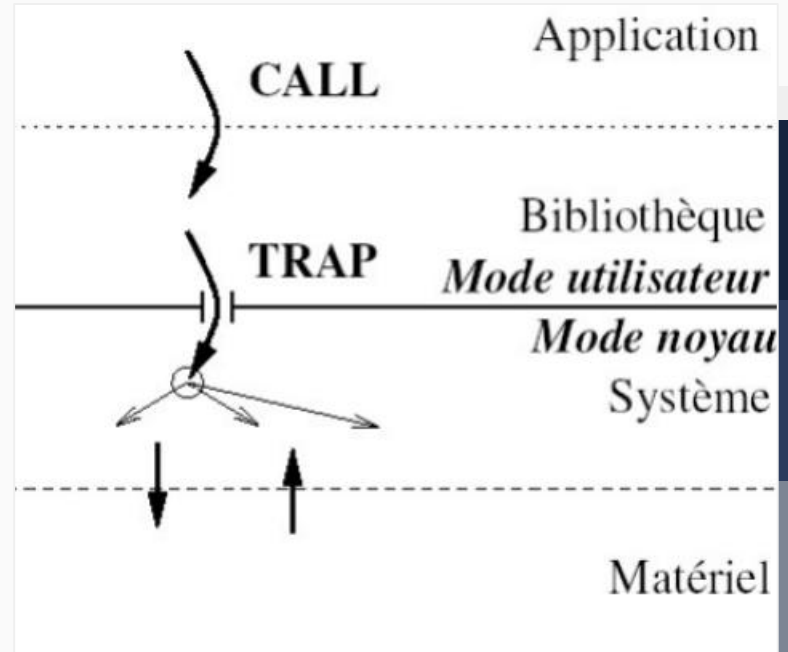
- **Non protégé** → le programme ne peut pas accéder directement aux ressources critiques.
- Exécution des programmes utilisateurs et manipulation des données propres au programme.



Modes d'exécution

2. Mode noyau (Kernel Mode) :

- Protégé et réservé au SE.
- Accès complet aux ressources : mémoire, ports d'E/S, périphériques, ...
- Permet de protéger le système et les autres processus contre les erreurs ou malveillances des programmes utilisateurs.
- Toute demande légitime de service du noyau doit être effectuée via **un appel système**.

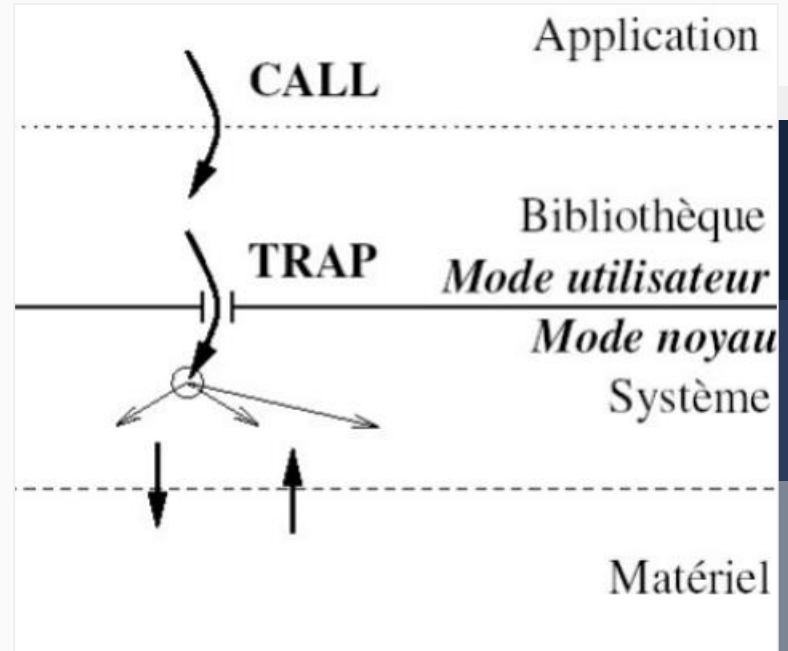


Modes d'exécution

Appels système (System Calls) :

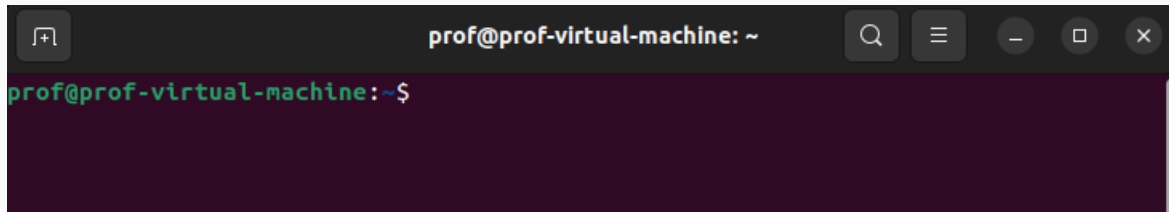
- Mécanisme qui permet aux programmes utilisateurs de **communiquer avec le noyau**.
- Lorsqu'un programme utilisateur demande un service du noyau, la fonction correspondante de la bibliothèque déclenche un trap* (instruction spéciale) que le CPU doit exécuter pour passer en mode noyau.

* **Le trap** est un mécanisme matériel/logiciel qui permet au processeur de passer du mode utilisateur au mode noyau afin d'exécuter le code du système d'exploitation associé à un appel système, puis de retourner le résultat au programme utilisateur.



Le fameux terminal

- C'est quoi un terminal ?
 - Un terminal est un programme qui émule une console dans une interface graphique, il permet de lancer des commandes.
- Pourquoi le terminal ?
 - Il est parfois plus simple de taper une commande que d'effectuer des manipulations demandant beaucoup de clics de souris dans une interface graphique.



Connexion et déconnexion

- Puisque Unix est un système multi-utilisateurs, alors il comporte les mécanismes d'identification et de protection permettant d'éviter toute interférence entre les différents utilisateurs.
- On distingue deux types d'utilisateurs: les administrateurs systèmes et les utilisateurs normaux:
 - **L'administrateur système** appelé aussi « root », utilisateur privilégié ou super utilisateur (super user). Il dispose de tous les droits sur la machine et le système Unix. Il s'occupe de l'administration du système, en particulier il crée les comptes des utilisateurs.
 - **L'utilisateur normal** dispose des droits réduits qui sont définis par l'administrateur système.

Connexion et déconnexion

- Unix associe à chaque utilisateur:
 - un nom d'utilisateur ou nom de connexion (appelé "login")
 - un mot de passe (password en anglais),
 - un Home Directory (répertoire de travail),
 - un langage de commandes (shell).

Connexion

- Pour se connecter à la machine et ouvrir une session de travail (pour pouvoir travailler sur la machine), il faut s'identifier. Pour cela, il faut:
 - Entrer le nom de connexion après le message «login»
Login : <on tape ici le nom d'utilisateur>
 - Entrer mot de passage après le message «password»
Password : <on tape ici le mot de passe>

Connexion

Remarque: Pour des raisons de sécurité, les caractères du mot de passe sont cachés, et la vérification se fait après avoir tapé le login et le mot de passe. Si le login ou le mot de passe est incorrecte, un message d'erreur est alors affiché: « **Invalid login name** »

- Une fois connecté, l'utilisateur se trouve alors dans son propre répertoire de connexion (home directory) correspondant à son login.

Attention: Unix fait la différence entre les minuscules et les MAJUSCULES.

Déconnexion


Pour terminer la session de travail, la méthode de déconnexion dépend de l'environnement de travail:

- Dans le cas d'un terminal, la commande de déconnexion est: **exit** ou **ctrl-D (^D)**
- Dans le cas d'environnement graphique, la méthode de déconnexion dépend de l'interface graphique.

Remarque importante: Si vous éteignez la machine avant d'utiliser les procédures de déconnexion, vous risquez d'endommager les fichiers sur les disques.



Introduction à la notion de système de fichiers

- Le fichier est la plus petite entité logique de stockage permanent sur un disque ou d'autres supports physiques. Il peut contenir du texte, des données, des images ou des programmes stockés sur un disque.
 - Les fichiers sont classés dans des répertoires (catalogues). Chaque répertoire peut contenir d'autres sous-répertoires, formant ainsi une organisation arborescente.
- 

Nommage des fichiers

- Le nom d'un fichier sous Unix est une suite de caractères, dont la taille peut aller jusqu'à 255 caractères.
- La plupart des caractères sont acceptés, y compris l'espace (très déconseillé). Cependant quelques caractères sont à éviter:
- * & ; () ~ **<espace>** \ | ` ? - (en début de nom)

Les différents types de fichiers

- Pour l'utilisateur sous Unix, il n'existe pas la notion de disques physiques (tout est fichier). L'utilisateur ne voit qu'une seule arborescence formée de répertoires et de fichiers.
- On distingue trois types principaux de fichiers :
 - **Les fichiers ordinaires**
 - **Les répertoires (Les fichiers répertoires)**
 - **Les fichiers spéciaux**

Les différents types de fichiers

Les fichiers ordinaires : Ce sont soit des fichiers contenant du texte, soit des exécutables (ou binaires), soit des fichiers de données.

Le système n'impose aucun format particulier aux fichiers et les traite comme des séquences d'octets.

Les répertoires (Les fichiers répertoires): c'est un ensemble de fichiers ou d'autres répertoires (sous-répertoires) de manière récursive. Ils permettent une organisation hiérarchique

Les différents types de fichiers

Les fichiers spéciaux : Ce sont des fichiers qui servent d'interface pour les divers périphériques (terminaux, disques durs, clavier, ...). Les opérations de lecture/écriture sur ces fichiers sont directement dirigées vers le périphérique associé. Les fichiers correspondant aux périphériques sont stockés dans le répertoire « **/dev** » (devices).

Système de fichiers d'Unix

- Un système de fichiers (File System en anglais), appelé aussi **système de gestion de fichiers**, est une structure de données qui définit l'organisation d'un disque (ou partition d'un disque). Il offre à l'utilisateur une vision homogène et structurée des données et des ressources : disques, mémoires, périphériques.
- Sous le système Unix, tout est traité comme un fichier, y compris les périphériques, les répertoires, les liens, etc.
 - les fichiers sont regroupés dans des répertoires.
 - les répertoires contiennent soit des fichiers, soit d'autres répertoires.

Système de fichiers d'Unix

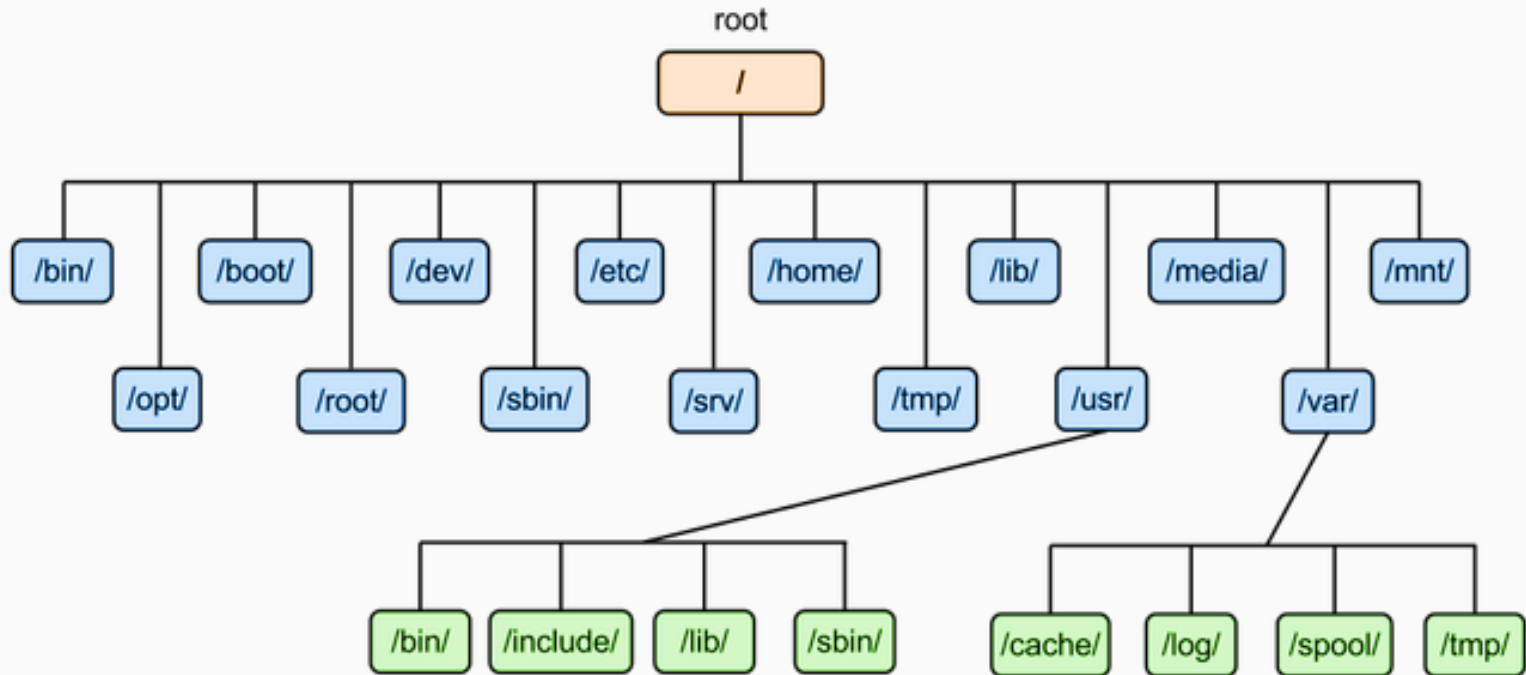
- Les fichiers sont organisés selon une structure hiérarchique en forme d'arbre :
 - Le point de départ est le **répertoire racine /**, c'est-à-dire le « **gros dossier de base qui contient tous les autres dossiers et fichiers** ».
 - Sous **/** se trouvent des fichiers et répertoires, qui peuvent eux-mêmes contenir d'autres fichiers/répertoires.
 - Les nœuds sont les répertoires non vides
 - Les feuilles sont les fichiers ou les répertoires "vides".

Système de fichiers d'Unix

Sous Unix, on a:

- Un seul arbre général
- Sa racine est désignée par « / »
- Chaque répertoire peut contenir des fichiers ou des sous-répertoires.

Structure des dossiers et des fichiers



Quelques dossiers important (1/3)

bin : contient des programmes (exécutables) susceptibles d'être utilisés par tous les utilisateurs de la machine.

boot : contient les fichiers permettant le démarrage de Linux.

dev : contient des fichiers spéciaux qui représentent les périphériques du système. On y retrouve ainsi par exemple le fichier qui représente le lecteur CD.

etc : contient les fichiers de configuration.

home : contient les répertoires personnels des utilisateurs.

Quelques dossiers important (2/3)

root : c'est le dossier personnel de l'utilisateur « root ». Normalement, les dossiers personnels sont placés dans home, mais celui de « root » fait exception. En effet, « root » est le super utilisateur, le « chef » de la machine en quelque sorte. Il a droit à un espace spécial.

lib : dossier contenant les bibliothèques partagées (généralement des fichiers.so) utilisées par les programmes. C'est en fait là qu'on trouve l'équivalent des « .dll » de Windows.

media : lorsqu'un périphérique amovible (comme une carte mémoire SD ou une clé USB) est inséré dans votre ordinateur, Linux vous permet d'y accéder à partir d'un sous-dossier de media. On parle de montage.

Quelques dossiers important (3/3)

mnt : c'est un peu pareil que media, mais pour un usage plus temporaire.

opt : répertoire utilisé pour les add-ons de programmes.

proc : contient des informations système.

sbin : contient des programmes système importants.

tmp : dossier temporaire utilisé par les programmes pour stocker des fichiers.

usr : c'est un des plus gros dossiers, dans lequel vont s'installer la plupart des programmes demandés par l'utilisateur.

var : ce dossier contient des données « variables », souvent des logs (traces écrites de ce qui s'est passé récemment sur l'ordinateur).

Les répertoires

- Un répertoire ou catalogue (directory) est un fichier qui contient une liste de noms de fichiers, parmi lesquels on peut trouver des sous-répertoires, et ainsi de suite (arborescence logique).
- Quand on crée un répertoire, le système génère automatiquement **deux** sous répertoires, du répertoire créé, qui sont:
 - le répertoire « **.** » : représente un lien vers le répertoire créé. Il désigne le répertoire courant.
 - le répertoire « **..** » : représente un lien vers le répertoire père.

Notion de chemin d'accès (pathname)

- Pour accéder à un fichier ou à un répertoire, on doit spécifier son emplacement dans l'arborescence du système de fichier. C'est-à-dire on doit spécifier **le chemin d'accès** (pathname) qui décrit l'emplacement où se trouve le fichier dans l'arborescence du système de fichier.
- **Le chemin d'accès** (pathname): est une **suite de noms de répertoires séparés** par « / » (slach), décrivant l'emplacement où se trouve un fichier ou un répertoire dans l'arborescence du système de fichiers.
- Pour accéder à un fichier, on utilise un chemin d'accès **absolu** ou **relatif**.

Chemins d'accès : Chemin absolu et relatif

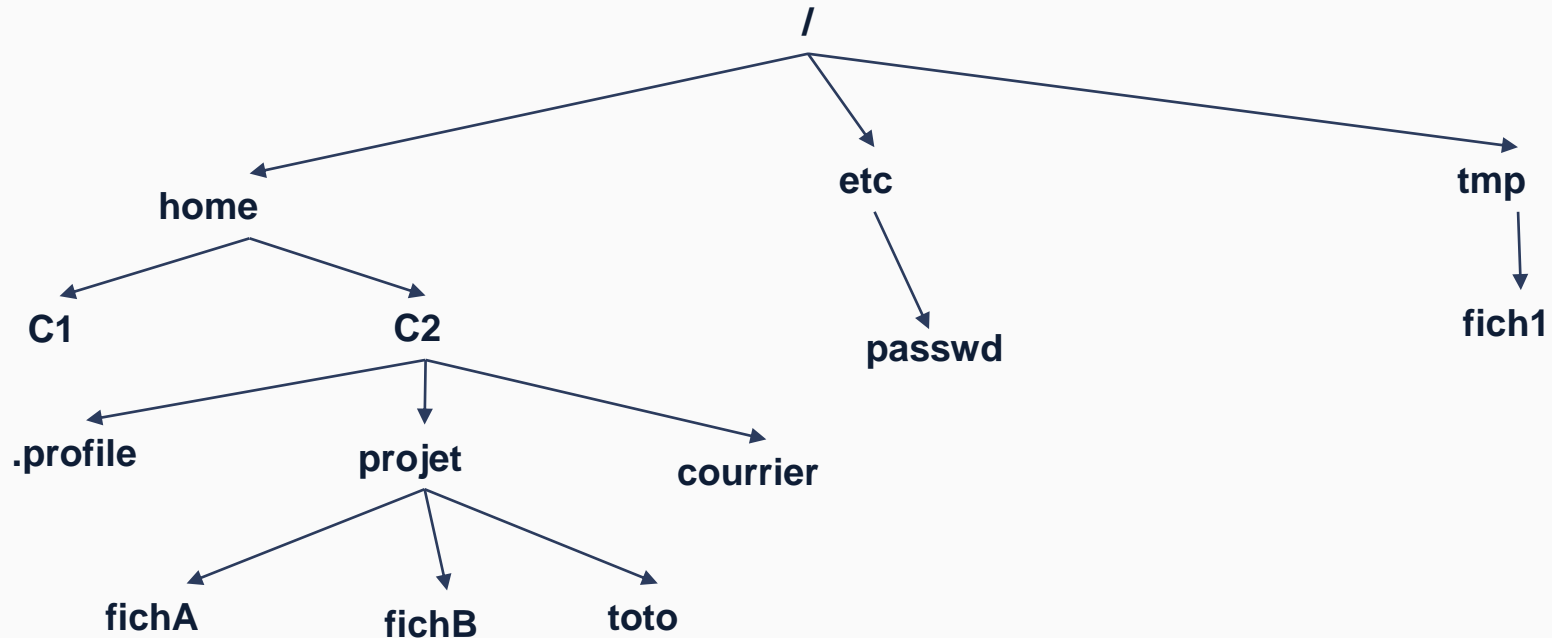
Chemin absolu

- C'est un chemin d'accès complet dans lequel on spécifie tous les répertoires à partir de la racine `/` jusqu'au fichier ou répertoire. Donc le chemin d'accès commence par la racine `«/»` (root directory).
- **Exemple** : `/home/azdad/Documents/rapport.txt`

Chemin relatif

- C'est un chemin d'accès dans lequel on spécifie tous les répertoires à partir du répertoire courant (où l'on se trouve). Par conséquent, un chemin relatif ne commence pas par `« / »`.
- **Exemple** : si on est dans `/home/azdad`, alors `./Documents/rapport.txt` est un chemin relatif.

Chemins d'accès : Exemple



Chemins d'accès : Exemple

- Un chemin absolu spécifie la suite des répertoires à traverser en partant de la racine, séparés par des caractères « / ».

Exemple:

Répertoire courant	Chemin absolu
/etc/passwd	/home/c2/projet/fichA

- Tout chemin qui ne commence pas par un caractère « / » (slash) est interprété comme un chemin relatif au répertoire courant.

Exemple:

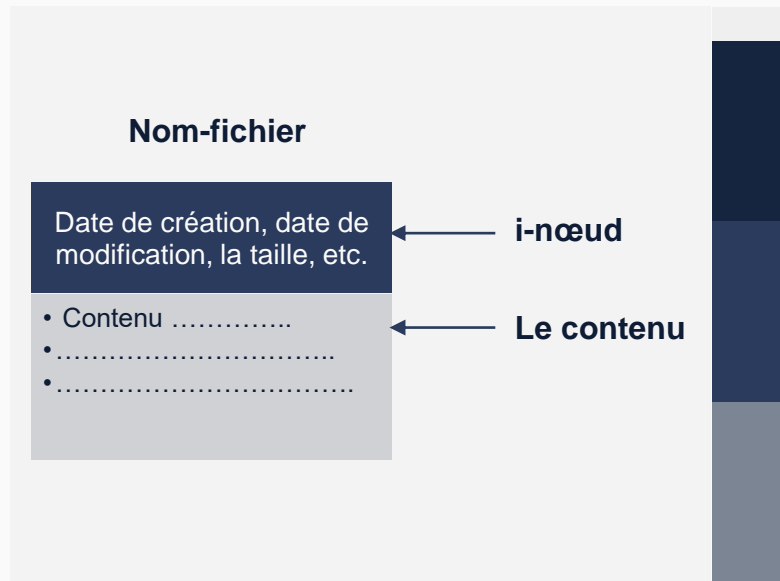
Répertoire courant	Chemin relatif
/home/c2	./profile
/home/c2/projet	./fichA
/tmp	../etc/passwd

Les fichiers Unix: Notion d'i-node

- Les fichiers Unix correspondent soit à des fichiers de données sur disque dur (fichiers ordinaires); soit à des répertoires, soit encore à des fichiers spéciaux permettant la gestion de certaines ressources du système.
- Un certain nombre de caractéristiques sont associées à un fichier : la date de sa création, celle de la dernière modification, le propriétaire, la taille, etc.
 - Ces informations sont regroupées dans une structure appelée **nœud d'index** (**i-nœud** ou **index node**).

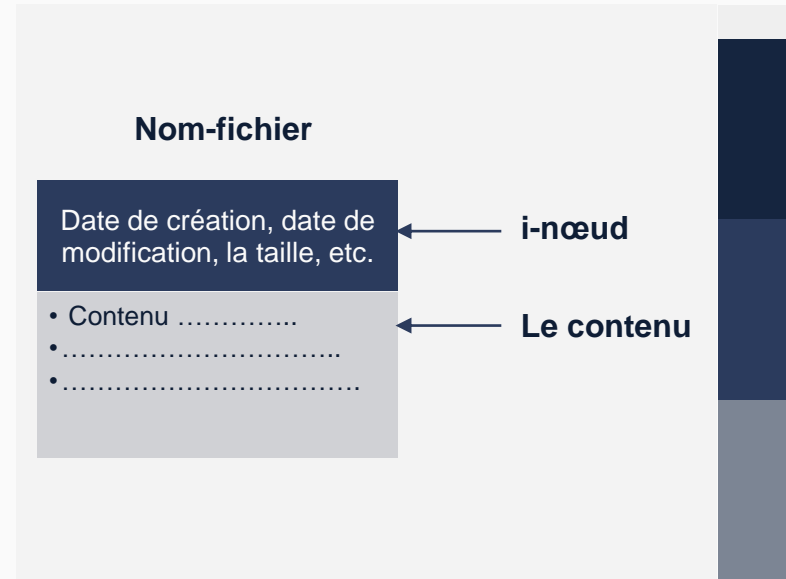
Notion d'i-node

- Dans un système de fichiers Unix, chaque fichier est composé de trois parties logiques :
 - **Nom du fichier**: c'est le nom visible dans un répertoire (par exemple : rapport.txt).
 - **i-nœud (i-node)** : Contient toutes les informations sur le fichier sauf son nom.
 - **Contenu (données)**: Les octets qui composent réellement le fichier sur le disque.



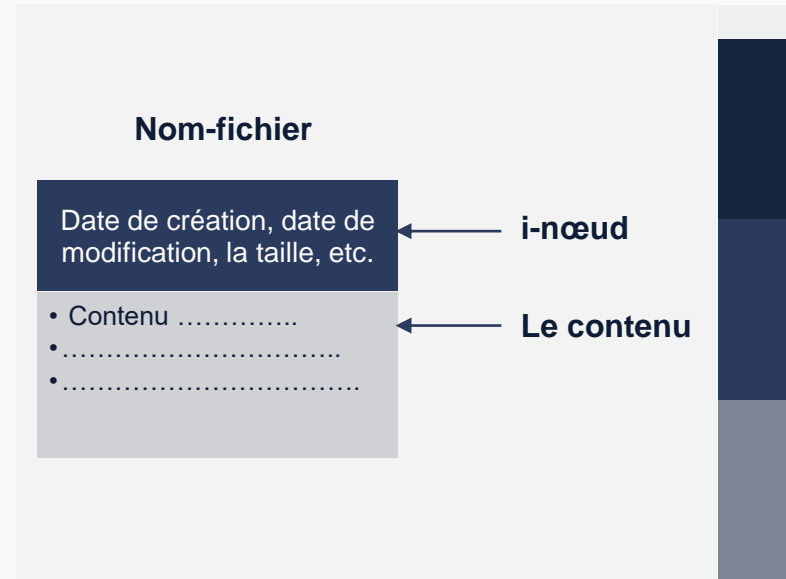
Notion d'i-node

- Métadonnées contenues dans l'i-node :
 - Numéro d'i-node (identifiant unique)
 - Permissions (lecture, écriture, exécution)
 - Propriétaire et groupe
 - Type et taille du fichier
 - Dates (création, modification, dernier accès)
 - Emplacement des blocs de données sur le disque (pointeurs vers les blocs)
 - Nombre de liens (combien de noms pointent vers cet i-node)



Notion d'i-node

- Le système de fichiers associe un nom de fichier à un i-node dans un répertoire.



Les données d'un fichier

- Les données sont stockées dans des blocs sur le disque dur.
- L'i-node pointe vers ces blocs pour retrouver le contenu.
- Le contenu peut être :
 - Texte,
 - Code binaire,
 - Données multimédia,
 - Etc.

Tables d'inodes & table catalogue

Chaque fichier est référencé dans deux tables :

- **Une table d'inœuds (ou table d'inodes)** : elle regroupe l'ensemble des i-nœuds d'un système de fichiers. Chaque i-nœud y est stocké de manière séquentielle et identifié par un numéro unique appelé numéro d'inœud (inumber). Ce numéro est unique pour chaque périphérique où le fichier est enregistré.
- **Une table catalogue** (une par répertoire) : cette table décrit les correspondances entre les noms de fichiers et les numéros d'i-nodes. La désignation d'un fichier se fait par l'intermédiaire du répertoire dans lequel il est stocké.

→ Le système identifie un fichier non pas par son nom, mais par son numéro d'inode.

Répertoire / Catalogue

- Répertoire = une table de liens (i-nombre, chaîne de caractères)
 - i-nombre = numéro d'index (i-noeud) de la structure décrivant le fichier.
 - Chaîne de caractère = un lien vers le fichier

Exemple :

fichier – répertoire « projet » contient:



Tables d'inodes & table catalogue : Cheminement

Clarification:

1. L'utilisateur demande un fichier par son nom (ex : `notes.txt`).
2. Le système de fichiers regarde dans la table de catalogue pour retrouver le numéro d'inode associé à `notes.txt`. (`notes.txt` → inode n°1234)
3. Une fois l'inode connu, le système accède à la table des inodes et lit l'inode n°1234.
 - Il récupère alors toutes les métadonnées (taille, droits, propriétaire, dates, etc.)
 - Et surtout les pointeurs vers les blocs de données.
4. Le système va chercher les blocs de données sur le disque (qui peuvent être dispersés).
5. Finalement, le contenu est lu/écrit et renvoyé au processus.

Répertoire de connexion

- Après le login, l'interpréteur de commande a pour répertoire courant le répertoire de connexion de l'utilisateur.
- **Le répertoire de connexion** contient aussi certains fichiers de configuration. Ces fichiers sont normalement invisibles car leur nom commence par un « `.` » point.
- On peut aussi spécifier un chemin à partir du répertoire de connexion d'un utilisateur en utilisant « `~` ».

Exemple : `~/projet/toto`

Le caractère tilde (« ~ »)

- Le caractère tilde (« ~ ») désigne le **répertoire de connexion d'un utilisateur** (le home directory):
- « ~ » seul désigne le répertoire de connexion de l'utilisateur courant.
- « ~nom » désigne le répertoire de connexion de l'utilisateur désigné par « nom ».

Exemples:

~etudiant/cours → désigne le sous-répertoire « cours » de l'utilisateur « etudiant ».

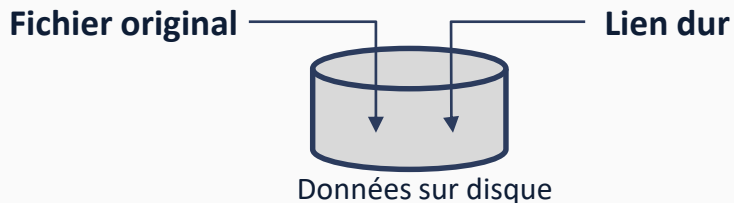
~/.profile → désigne le fichier « .profile » situé dans le répertoire de connexion de l'utilisateur courant.

Liens entre les fichiers

- Le système identifie un fichier (i-nœud + contenu) non pas par un nom, mais par son numéro de l'i-nœud.
 - ➔ il est possible de donner plusieurs noms à un même fichier grâce à la notion de lien: ceci permet d'accéder au même fichier à différents endroits de l'arborescence.
- On distingue deux types de liens :
 - les **liens symboliques** « symbolic links »
 - les **liens durs** (ou **liens physiques**) « hard links ».

Les liens physiques « hard »

- Un lien dur permet à plusieurs noms de fichiers de partager le même inode, c'est-à-dire le même contenu.
- Dans ce cas, le même fichier physique est pointé par différents noms de fichiers.
- Le fichier source et le fichier lien pointent directement sur les données résidant sur le disque (l'information ne réside qu'une seule fois sur le disque mais elle peut être accédée par deux noms de fichiers différents). Les droits du fichier source ne sont pas modifiés.



Les liens physiques « hard »

Avantage:

- Une seule copie sur le disque et plusieurs façon d'y accéder.
- Modification immédiatement sur tous les fichiers liés.
- Simplifier l'accès à des fichiers dont le nom est difficile à retenir.
- Si le fichier source est effacé, le contenu n'est pas perdu et il est toujours accessible par le fichier lien.

Limitation:

- Impossible de faire un lien sur un répertoire.
- Pas de lien dur sur les fichiers spéciaux

Les liens symboliques

- Un lien symbolique est un fichier texte spécial, rajouté dans la table des i-nodes, qui contient un lien vers un autre fichier ou répertoire.

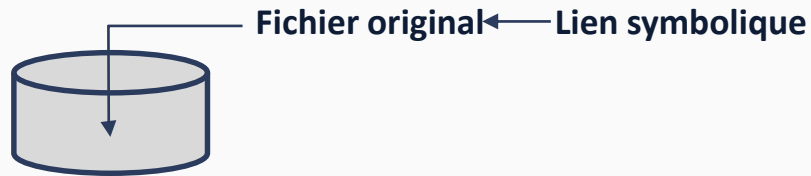
→ Un **lien symbolique** est un **fichier spécial** qui **contient le chemin (l'adresse)** d'un autre fichier ou répertoire.

→ Donc son numéro d'inode est différent du fichier source mais qui pointe sur le fichier source pour lui permettre d'accéder aux informations sur le disque.

- Toute opération sur ce fichier (lecture, écriture, ...) s'effectue sur le fichier référencé.

Les liens symboliques

- **Attention:** si le fichier source est effacé, le contenu est perdu et le fichier lien pointe sur quelque chose d'inexistant. L'information est donc perdue.



- Ces liens, permettent de lier plusieurs noms à un même fichier sans rattacher ces derniers au i-nœud correspondant.

Les liens symboliques

Remarques

- Un lien symbolique ne possède pas les limitations du lien dur ➔ Possibilité de faire des liens symboliques sur des répertoires.
- La suppression de tous les liens symboliques n'entraîne que la suppression de ces liens, mais pas la suppression du fichier pointé.
- La suppression du fichier pointé n'entraîne pas la suppression des liens symboliques associés. Dans ce cas le lien pointe dans le vide.

Les processus

- Le système Unix/Linux est multi-tâches car plusieurs programmes peuvent être en cours d'exécution en même temps sur une même machine. Mais à chaque instant, le processeur ne traite qu'un seul programme lancé (un seul processus).
- La gestion des processus est effectuée par le système.
- Un processus est une tâche en train de s'exécuter. Il est doté d'un espace d'adressage (ensemble d'adresses) dans lequel le processus peut lire et écrire

Les entrées/sorties

- Lors de l'exécution d'une commande, un processus est créé. Celui-ci va alors ouvrir trois canaux de communication:
 - l'entrée standard,
 - la sortie standard,
 - la sortie d'erreurs standard.
- A chacun des trois canaux est affecté un nom de fichier et un numéro :

Canal de communication	Fichier	descripteur
Entrée standard	stdin	0
Sortie standard	stdout	1
Sortie d'erreurs standard	stderr	2

Les entrées/sorties

Le fichier « stdin »: le processus lit les données en entrée à partir du fichier « stdin ». Il est ouvert avec le numéro logique 0. Par défaut, il est associé au clavier.

Le fichier « stdout »: le processus écrit les sorties qu'il produit dans le fichier « stdout ». Il est ouvert avec le numéro logique 1. Par défaut, il est associé à l'écran.

Le fichier « stderr » : le processus écrit les messages d'erreur dans le fichier « stderr ». Il est ouvert avec le numéro logique 2. Par défaut, il est associé à l'écran.