

# Les systèmes d'exploitation



Philippe ROOSE  
IUT de Bayonne

# Un système informatique

---

## □ Objectifs

- automatiser le traitement de l'information.
- constitué de deux entités :
  - matériel
  - logiciel.

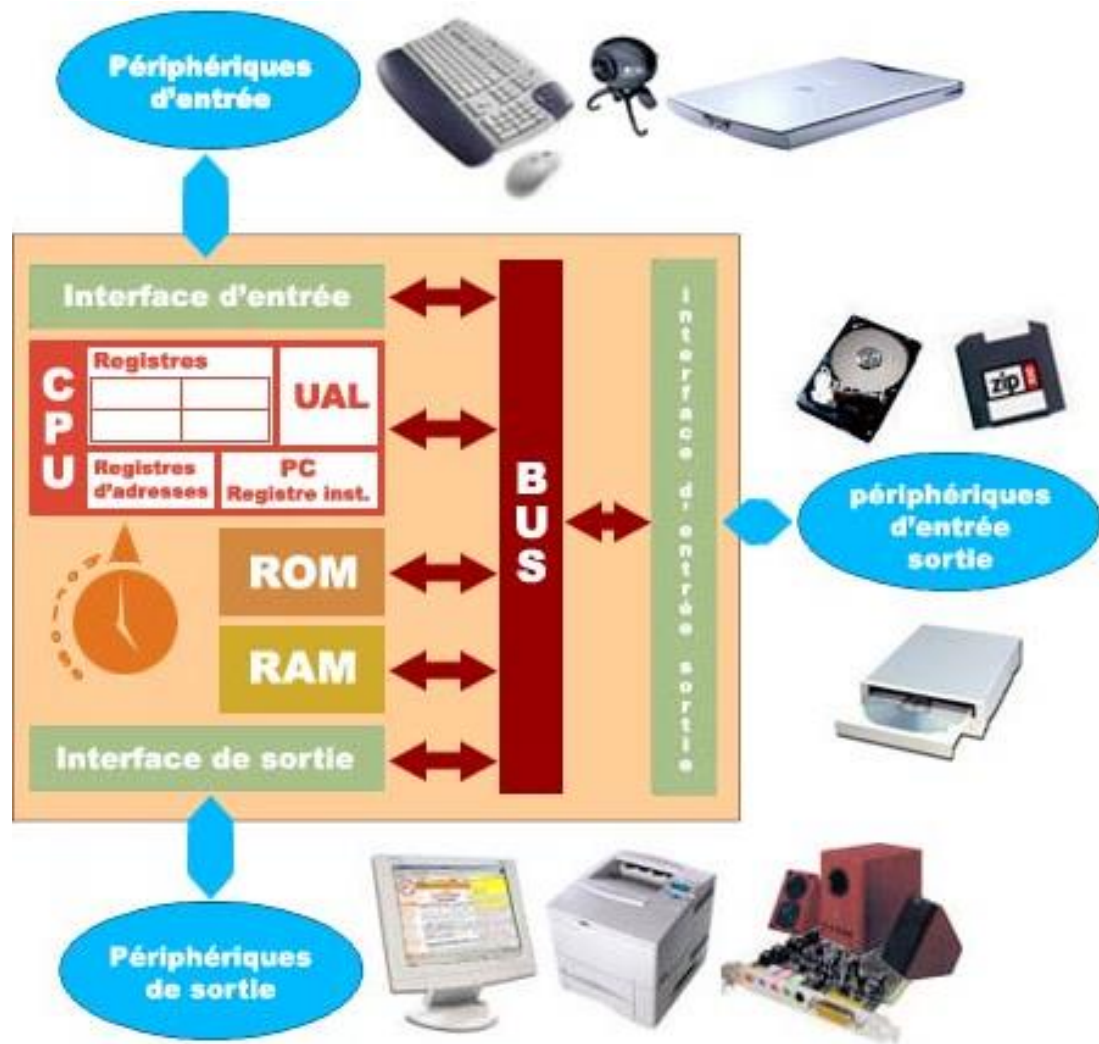
## □ Matériel, un ordinateur est composé de :

- L'Unité Centrale (UC) pour les traitements
- La Mémoire Centrale (MC) pour le stockage
- Les Périphériques : disque dur, clavier, souris, carte réseau... accessibles via des pilotes de périphériques
- Machine de type Von Neumann

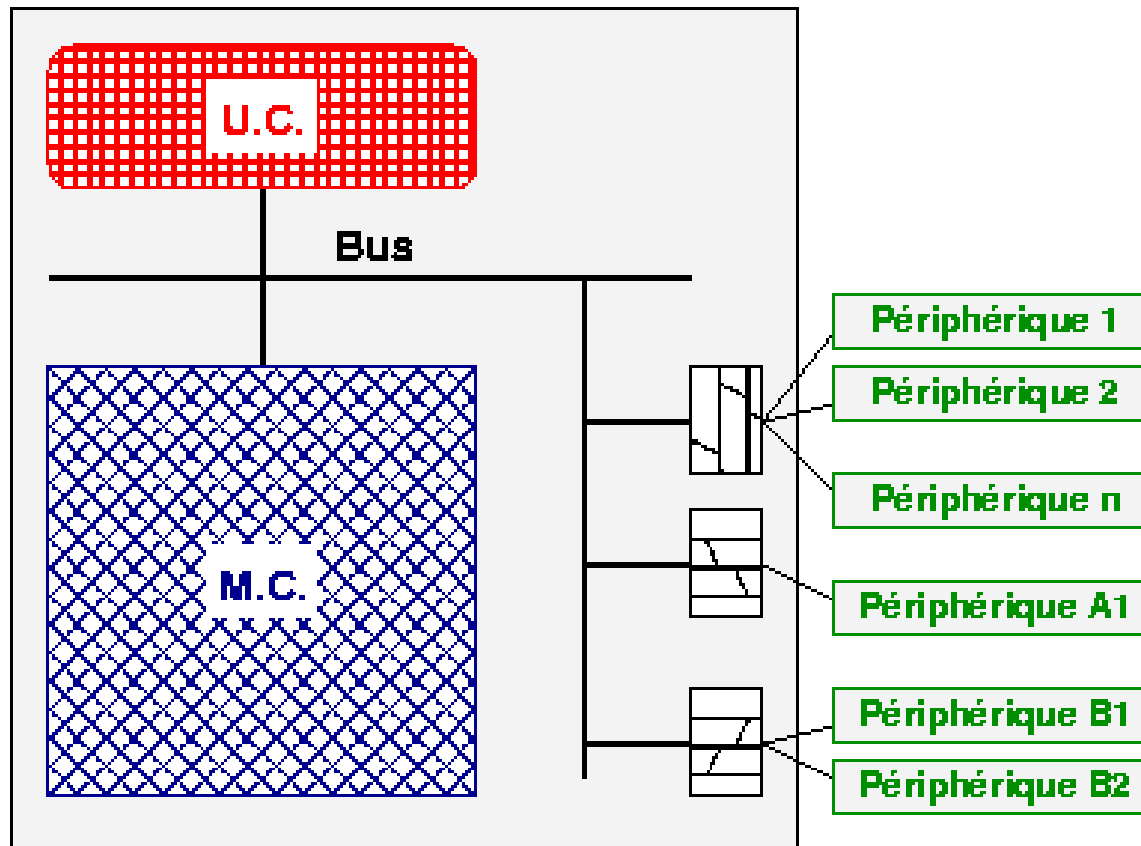
## □ Logiciel

- Un système d'exploitation est un logiciel dont le rôle est de gérer tous les périphériques et de fournir aux programmes utilisateur une interface simplifiée avec le matériel.

# Un système informatique (matériel)



# Un système informatique (matériel)



# Périphériques

---

- ❑ Dans un ordinateur, on trouve des périphériques d'entrée, des périphériques de sortie et des périphériques d'entrée/sortie d'information.
- ❑ Généralement, chaque périphérique peut faire entrer ou recevoir des informations sous une forme précise (texte, image, son, vidéo, ...). Mais toutes ces informations doivent être converties en format numérique pour qu'elles soient traitées par l'ordinateur.
  - Périphériques d'Entrée
  - Périphériques de Sortie

# Exemple : Sortie/Ecran

---

- ❑ Il permet l'affichage des informations qui seront manipulées ou exploitées par l'utilisateur.
- ❑ Il existe deux principales technologies d'écran: ~~les écrans à tube cathodique (CRT)~~ et les écrans plats (LCD, TFT).
- ❑ Les écrans (ou moniteurs) sont caractérisés par les données suivantes :
  - **La résolution** : C'est le nombre de points (pixel : Picture Element) horizontaux et verticaux que peut afficher un écran.
    - ❑ Exemple: 1152 x 864 ; 1920 x 1080 (full hd); 3840 x 2160 (UHD 4 K)
  - **La taille** : Elle est mesurée en pouces.
    - ❑ Exemple : 14" - 15" - 17" - 19" - 21"

# Les ports

---

- ❑ **Port parallèle** : chaque bit est transféré sur un fil. Un port parallèle est souvent utilisé par des périphériques comme les imprimantes et les scanners.
- ❑ **Port en série** : les bits sont transférés par le même fil l'un à la suite de l'autre. Un port série est souvent utilisé par des périphériques comme la souris, le modem, le clavier, etc.
- ❑ **Port USB** (Universal Serial Bus)
- ❑ **Port RJ45** (réseau)

# Les mémoires

---

- ❑ Ce sont des supports de stockage et d'enregistrement des informations de façon permanente, ils peuvent aussi être transportables d'un ordinateur à un autre.
- ❑ L'ordinateur ne manipule que des nombres binaires (0 et 1) appelé Bit.
- ❑ Chaque caractère (lettre, chiffre, symbole) est rangé dans une mémoire occupe 8 bits avec 8 bits = 1 octet
- ❑ Donc l'unité de mesure de la capacité de stockage d'une mémoire est l'octet.
- ❑ On trouve aussi :
  - Le Kilo octet Ko =  $2^{10}$  octets = 1024 octets
  - Le Méga octet Mo =  $2^{10}$  Ko
  - Le Giga octet Go =  $2^{10}$  Mo
  - Le To = ...
  - Le Po = ...



# Mémoire de masse

---

- ❑ Le disque dur : Un disque dur est caractérisé par :
  - Une capacité de stockage très importante (exprimé en Go/To)
  - Un temps d'accès aux données très court(millisecondes « ms »)
  - Une durée de vie assez longue
- ❑ Le CD (Compact Disk) et DVD (Digital Versatile Disk):
  - Un CD offre une capacité de stockage de 700 Mo équivalent à 450 disquettes.
  - Un DVD offre une capacité de stockage de l'ordre de quelques Go.
  - CD-R (Compact Disk - Read) : c'est un disque optique inscriptible une seule fois avec l'utilisation d'un graveur.
  - CD-RW (compact Disk – Read Write) : c'est un disque optique réinscriptible (accessible en lecture et en écriture).
- ❑ La Clé USB

# La carte mère

---

- ❑ C'est le plus grand circuit électronique de l'ordinateur. Elle est couverte de connecteur où sont branchés les autres composants de l'unité centrale.
- ❑ Cette carte contient le processeur, les mémoires internes, les autres cartes contrôleurs (carte vidéo, carte son, carte fax modem,...) ; ainsi que le système de communication interne qui assure la circulation des informations à travers tous composants de l'ordinateur (les bus).
  - CPU (Central Processing Unit) ou unité central de traitement. Le microprocesseur modern peut rassembler quelques millions de transistors (circuits électriques) sur une puce (plaque de silicium) d'environ cinq à six centimètre carrés. Il assure toutes les opérations de traitement (calcul, recherche, organisation,...) effectuées sur les informations proposés. Le microprocesseur est composé de 3 unités.
  - Unité arithmétique et logique (calcul) UAL.
  - Les registres sont des zones de stockages temporaires.
  - Unité de contrôle UC.
- ❑ **Remarque** : La vitesse de l'ordinateur est mesurée par la fréquence de fonctionnement de son processeur (nombre d'opération par seconde).
- ❑ 5 GHz
- ❑ Exemples : Core ix d'Intel à 1,60 GHz à

# Mémoires internes

---

- ❑ **ROM** (Read Only Memory = Mémoire à Lecture Seulement): C'est une mémoire morte dont le contenu est inscrit de manière définitive par le constructeur.
- ❑ **RAM** (Random Access Memory = Mémoire à Accès aléatoire): C'est une mémoire vive à accès en lecture et écriture. C'est un ensemble de cases auxquelles on peut accéder directement grâce à leur adresse. La RAM est une mémoire volatile, dont le contenu se perd dès que la machine n'est plus sous tension. ( $\neq$  ROM)
- ❑ **Mémoire cache** : C'est une mémoire à temps d'accès très rapide. Elle est intégrée dans le CPU pour améliorer la vitesse de transfert des informations entre la RAM et le processeur. Le rôle d'une mémoire cache est de stocker une petite quantité de données, mais qui doit être immédiatement accessible.
- ❑ **Mémoire CMOS** (Complementary Metal Oxide Semiconductor): C'est une mémoire volatile à taille réduite, alimentée par une pile. Elle permet l'enregistrement de la configuration du démarrage du système (tel que la date et l'heure du système, le mot de passe,...).

# Les Bus

---

- ❑ Les différents composants d'un ordinateur sont interconnectés par des systèmes de communication assurant la transmission des signaux nécessaires à son fonctionnement .les composants de la carte mère sont interconnectés par un ensemble de canaux, appelés bus.
- ❑ Le bus processeur : CPU <-> Puce de la carte mère (chipset).
- ❑ Le bus mémoire : CPU <-> RAM
- ❑ Le bus d'adresse et le bus de commande : Elles sont chargées de véhiculer des adresses et des commandes.
- ❑ Le bus de données(ou d'entrée/sortie) : le bus d'entrée/sortie permet non seulement aux composants figurant sur la carte mère de communiquer des mais également d'ajouter des périphériques supplémentaire à l'aide de carte d'extension.

# Les adaptateurs

---

- ❑ Ce sont des cartes d'extension qu'on appelle également des cartes contrôleurs ou simplement des interfaces. Parmi ces cartes on peut citer :
  - Carte graphique : Elle produit les images envoyées à l'écran.
  - Carte son : Transforme le PC en une machine de traitement sonore/musical.
  - Carte réseau : permet la connexion d'un ordinateur à un réseau local.

# Historique

---

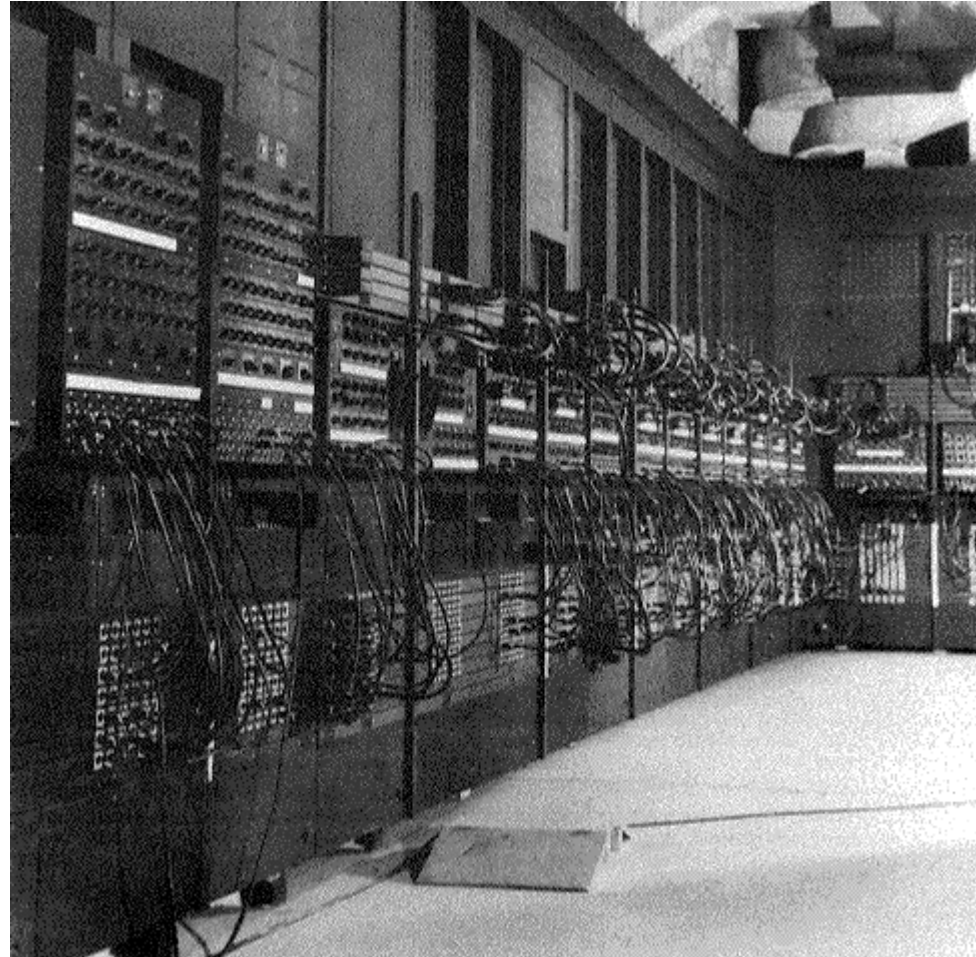
- ❑ 1945 - 55 : tubes et interrupteurs
  - Pas de système d'exploitation
- ❑ 1955 - 65 : transistors, cartes perforées
  - Traitement par lots
- ❑ 1965 - 80 : circuits intégrés, disques
  - Multiprogrammation, temps-partagé, entrées/sorties
  - Unix, version BSD, AT&T, interface POSIX
- ❑ 1980 -- : ordinateurs personnels (PC)
  - Interface graphique (concept créé vers 1960, Stanford)
  - Réseaux et systèmes distribués
  
- ❑ La complexité du matériel, l'hétérogénéité des périphériques, la sécurité, l'intégrité des données, la facilité d'utilisation
  - Système d'exploitation nécessaire

# Historique

---

## ▣ Premier ordinateur...

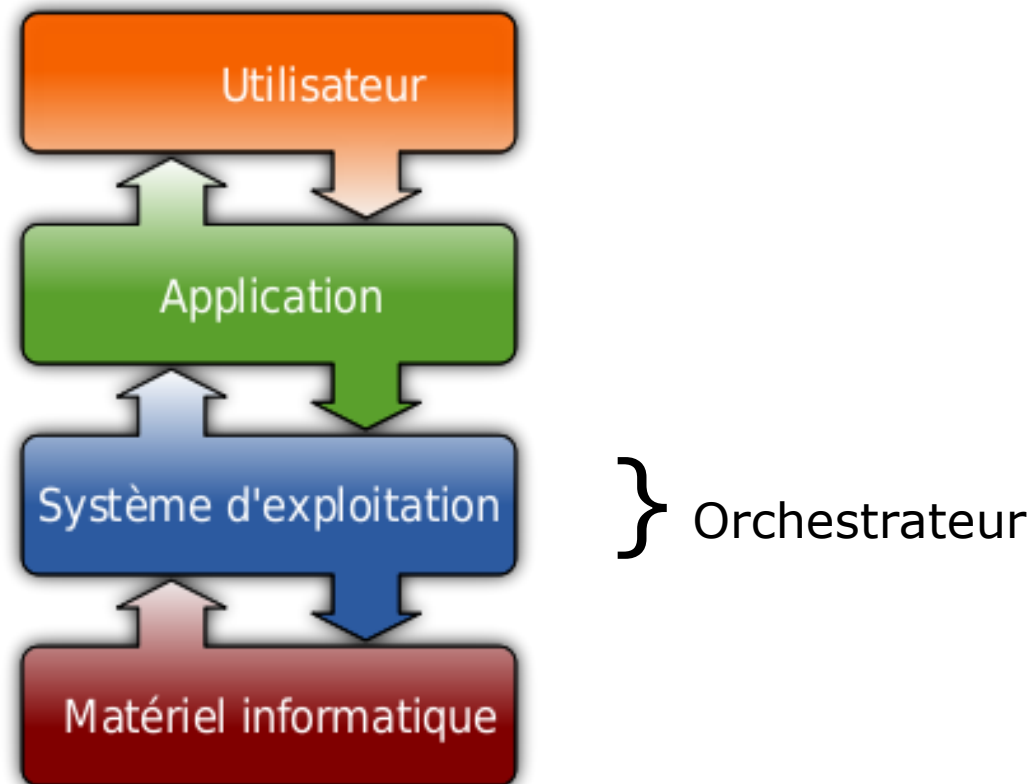
- ▣ 19.000 Tubes à vide
- ▣ 1500 relais
- ▣ 100.000 résistances
- ▣ 100 kHz
- ▣ 50 tonnes
  
- ▣ L'ENIAC



# Position du SE/OS

Système d'exploitation/Operating System

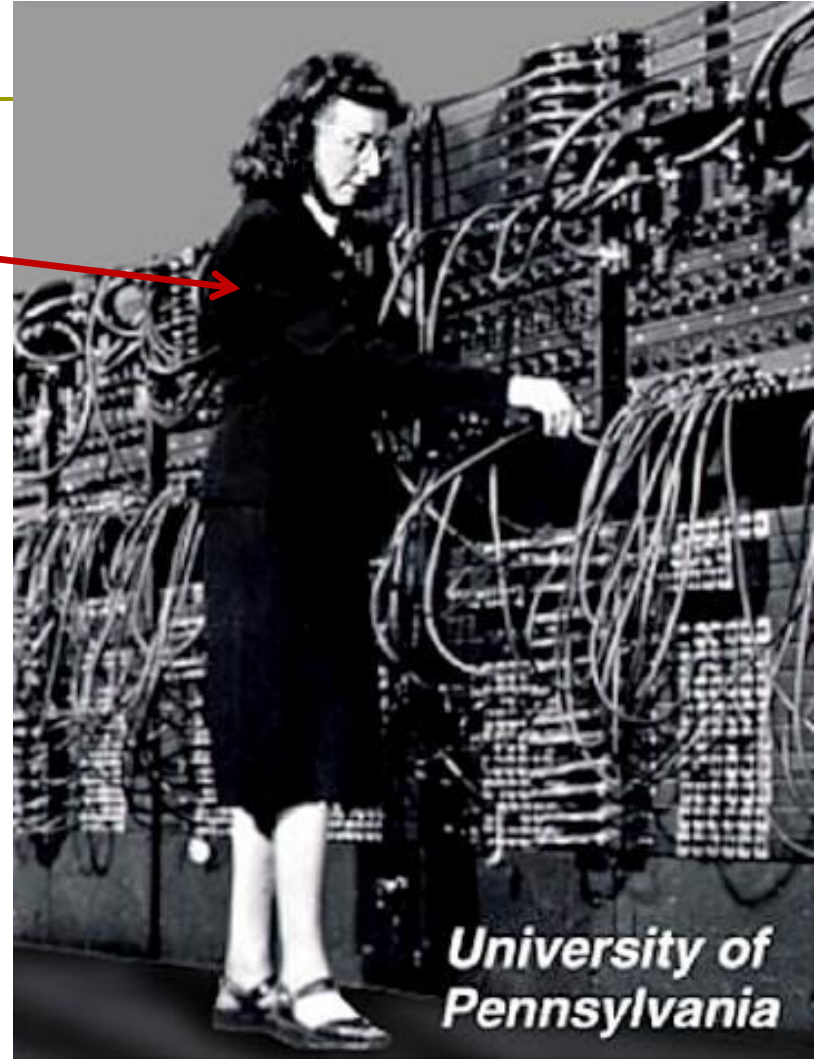
---





# Historique

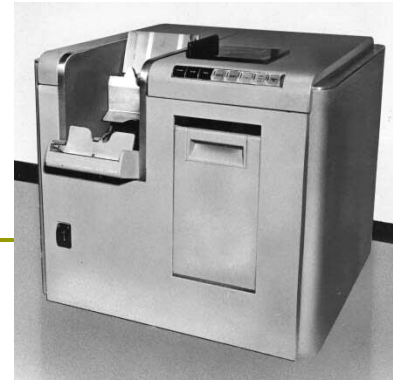
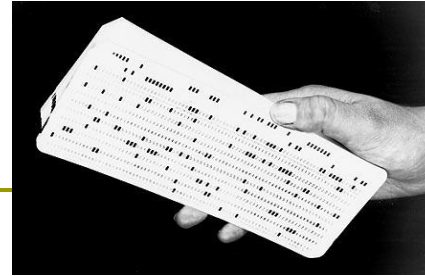
- Premier OS !
- En réalité
  - Rien!
  - Pas de distance  
Hardware-Software
  - 1 utilisateur
  - 1 programme à la fois



# Familles de systèmes d'exploitation

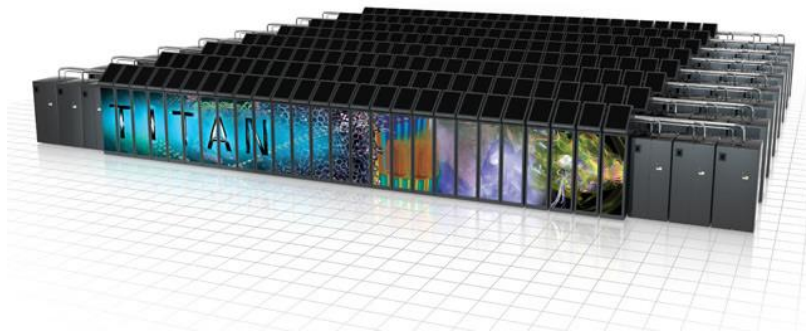
## ■ Les SE pour Mainframes

- Les mainframes sont des ordinateurs qui remplissent une pièce et qui sont généralement dédiés au centre de calcul de grandes sociétés. Ces ordinateurs peuvent être dotés d'un millier de disques pour plusieurs téraoctets de mémoire.
- Les SE pour ces ordinateurs offrent 3 services :
  - *Le batch* : qui exécute des travaux (jobs) sans interaction avec l'utilisateur,
  - *Le transactionnel* : qui gère en permanence un très grand nombre de petites requêtes concurrentes (ex : système de réservation de billets d'avion),
  - *Le temps partagé* : permet à plusieurs utilisateurs, pouvant être distants, d'exécuter des travaux en même temps sur la même machine.
- Exemple de SE pour mainframe : S/390 d'IBM.



# Titan (Cray)

---



27 000 millions de millions d'opérations/seconde  
Utilisé pour des études sismiques au niveau de la planète, en biologie, etc.

- 8 MW/an
- 300 000 cœurs

## Architecture hybride

- 18 688 processeurs AMD Opteron 6274, 16 cœurs à 2,2 GHz
- 18 688 accélérateurs GPU Nvidia Tesla K20X1,3.
- 27 PFLOPS en performance de pointe.
- RAM : 710 TB (598 TB CPU et 112 TB GPU).
- OS : Cray Linux Environment
- Puissance électrique : 8,2 MW.
- Coût : 97 millions de dollars.

Pour info : 100 MW - Tech : la puissance électrique du [barrage de Verbois](#), dans le [canton de Genève](#), en [Suisse](#).

# Familles de systèmes d'exploitation

---

- ❑ **Les SE « serveurs »**
- ❑ Servent en parallèle de nombreux utilisateurs à travers le réseau et permettent à ces derniers de partager des ressources communes : imprimantes, disques, logiciels...
- ❑ *Unix* et *Windows Server* sont les représentant les plus courant de cette famille.
- ❑ Serveurs d'imprimantes, serveurs Web, Billettique, etc.



# Familles de systèmes d'exploitation

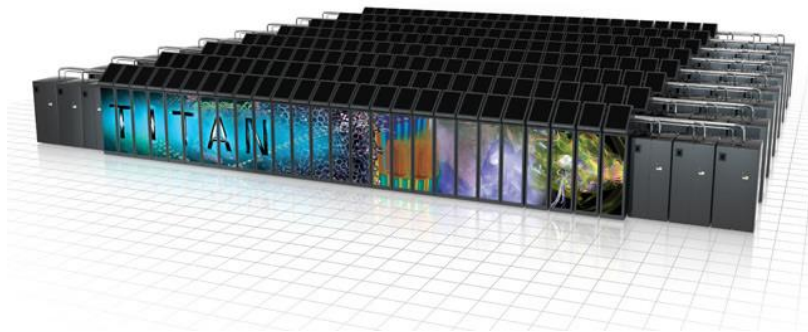
---

- ❑ **Les SE « Personnels »**
- ❑ Fournissent une interface plus ou moins conviviale avec l'utilisateur.
- ❑ Sont principalement utilisés pour les applications bureautiques, l'accès à internet.
- ❑ Exemple : Famille Windows, Mac OS et Linux.



# Titan (Cray)

---



27 000 millions de millions d'opérations/secondes  
Utilisé pour des études sismiques au niveau de la planète, en biologie, etc.

- 8 MW/an
- 300 000 cœurs

## Architecture hybride

- 18 688 processeurs AMD Opteron 6274, 16 cœurs à 2,2 GHz
- 18 688 accélérateurs GPU Nvidia Tesla K20X1,3.
- 27 PFLOPS en performance de pointe.
- RAM : 710 TB (598 TB CPU et 112 TB GPU).
- OS : Cray Linux Environment
- Puissance électrique : 8,2 MW.
- Coût : 97 millions de dollars.

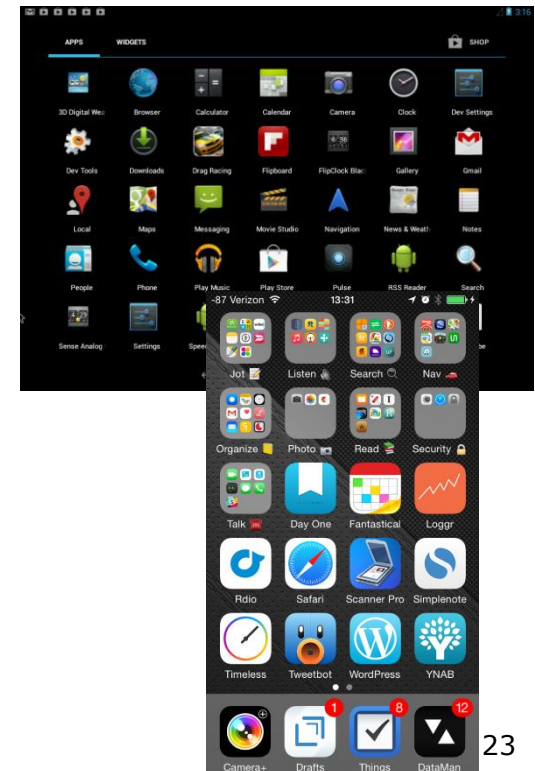
Pour info : 100 MW - Tech : la puissance électrique du [barrage de Verbois](#), dans le [canton de Genève](#), en [Suisse](#).



# Familles de systèmes d'exploitation

## Systèmes embarqués

- ❑ Notion d'informatique l'embarquée : Embarquée dans quoi ?
- ❑ Dans les matériels informatiques enfouis :
  - **Informatique Militaire** (*missiles, sous-marins, radars, capteurs de terrains, etc.*)
  - **Informatique Scientifique** (*satellites, scanners, microscopes électroniques, etc.*)
  - **Informatique grand public** (*téléphones, avions, voitures, hi fi, GSM, PDA, capteurs sans fils, etc.*)
- ❑ **SE Embarqués**
  - Ils sont dédiés pour des machines pilotant des périphériques qui ne sont pas forcément dépendant d'un ordinateur (ex : télévision, téléphones).
  - De part leur dimension, leur taille mémoire et leur restriction en terme d'alimentation ils forment une famille à part.
  - Exemple de SE
    - ❑ [TinyOS](#) (capteurs), Android, iPhoneOS
- ❑ Il y a parfois « fusion » entre Systèmes TR & Embarqués



# Deux grandes familles...

---



## ❑ **Unix**

- ❑ Non propriétaire : Gratuit le plus souvent
- ❑ Ouvert : sources disponibles
- ❑ Flexible : sources modifiables
- ❑ Puissant : Programmable
- ❑ Communauté active : entraide des utilisateurs
- ❑ Plus complexe : plutôt pour les informaticiens (interfaces de **programmation** optimisées)

## ❑ **Windows**

- ❑ Propriétaire : Payant
- ❑ Sources non disponibles
- ❑ Sources non modifiables
- ❑ Communauté active : nombreux utilisateurs
- ❑ Plus ergonomique : pour les utilisateurs (interfaces **d'utilisation** optimisées)



# Logiciels

---

- ❑ Un système informatique est composé de deux niveaux bien distincts :
  - le système d'exploitation **et**
  - les applications.
  
- ❑ Le principe du système d'exploitation est de
  - masquer les caractéristiques physiques du matériel.
  
- ❑ L'objectif du logiciel
  - d'offrir aux utilisateurs des fonctionnalités adaptées à leurs besoins.

# Un système d'exploitation

---

- Les deux objectifs majeurs d'un système d'exploitation sont :
  - **Transformer** le matériel en une machine utilisable, c.-à-d. fournir des outils adaptés aux besoins des utilisateurs indépendamment des caractéristiques physiques,
  - **Optimiser** l'utilisation des ressources (matérielles et logicielles).

# Un système d'exploitation

---

- ❑ Les systèmes d'exploitation (ou SE) regroupent au moins 4 fonctionnalités essentielles :
  - La gestion des processus
  - La gestion de la mémoire
  - Le système de fichiers
  - Les entrées/sorties

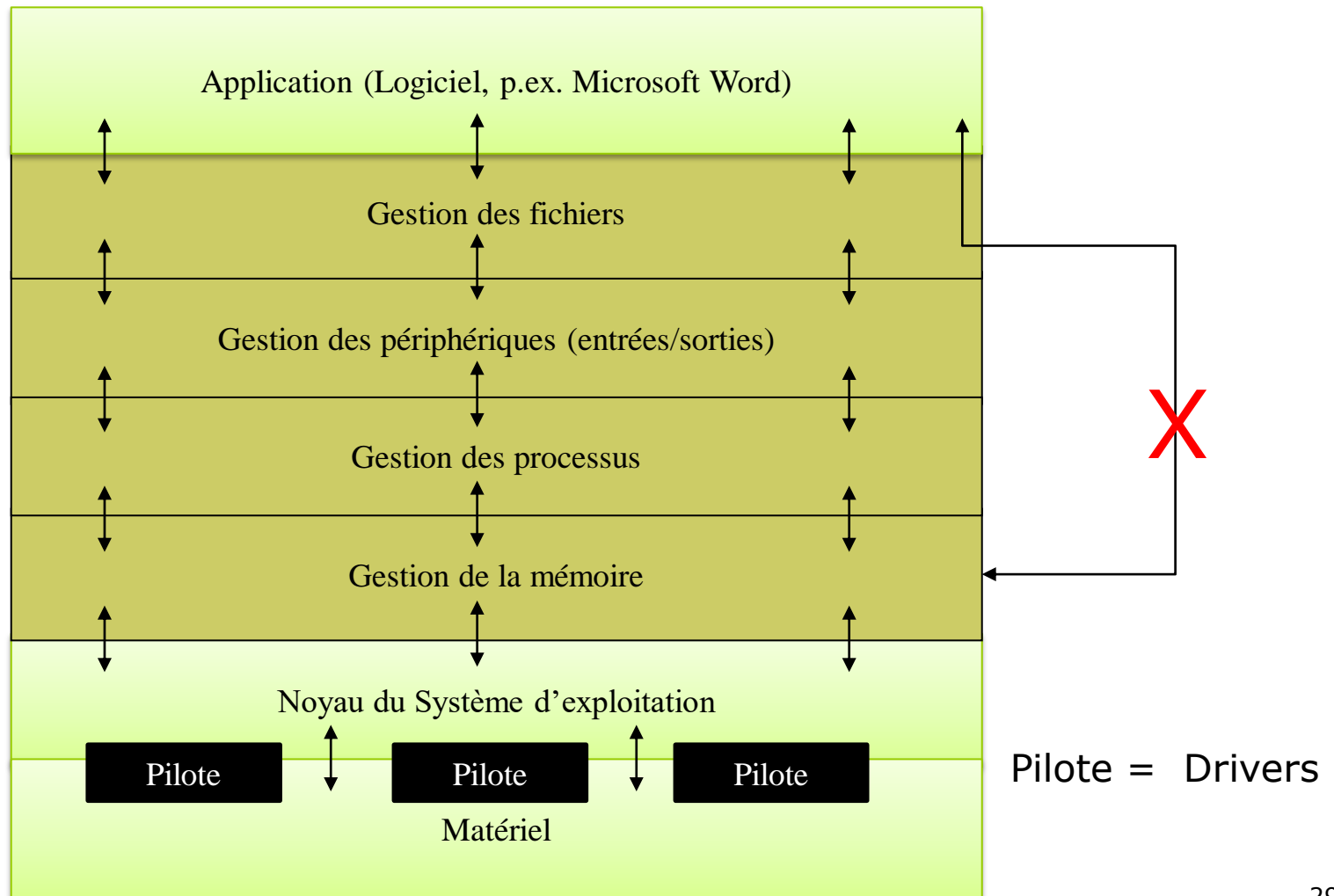
Nous allons voir maintenant chacune de ces caractéristiques

# Un système d'exploitation

---

- Ces deux objectifs doivent être atteints en garantissant un bon niveau en :
  - **Sécurité** : intégrité, contrôle des accès, confidentialité...
  - **Fiabilité** : satisfaction des utilisateurs même dans des conditions hostiles et imprévues
  - **Performance** du système informatique

# Plus « technique »



# La gestion des processus



- Un processus au sein d'un SE correspond à une instance d'un programme en cours d'exécution
  - Un seul processus en mémoire (*mono tâche*)
  - Exécuté en parallèle avec d'autres (*multitâches*)
  
- S'il est multitâches le SE doit permettre
  - de créer ces processus (= *instances de programme, éventuellement multiples*),
  - de les gérer,
  - de les synchroniser,
  - de leur permettre de communiquer ensemble.



# Point de vue historique

---

- ❑ Au début les SE étaient **mono processus** car on faisait surtout du calcul donc on utilisait les CPU au maximum.
- ❑ Plus tard, **programmes interactifs** et avec fichiers => beaucoup d'attente sur l'UE => le CPU est sous utilisé => changer de processus quand le processus courant est en attente = **multiprogrammation**.
- ❑ De plus en plus de machines multiprocesseurs => le partage du CPU se fait dans le temps **ET** dans l'espace.

# Le multitâche coopératif

---

- Lorsqu'un des processus fonctionne, les autres sont mis en attente. Ils ne reprendront la main (selon certaines priorité) que lorsque le processus en cours « donnera la main ».
  - Windows 3.1 -> 98
  - Pb : Si une appli plante, tout est planté



# Le multitâche préemptif

---

- ❑ Le cœur du système qui donne successivement la main aux différents processus pour un certain quantum de temps
  - Une fois ce quantum épuisé, le processus est mis ou remis dans la file d'attente et la main est donnée au processus suivant ayant la priorité la plus haute.
  - Mac OS, Unix/Windows > NT/XP/Vista/7/8/10
  - Si plantage, pas de monopolisation du processeur

# La gestion de la mémoire



- Lorsque le SE doit exécuter un programme, création d'une instance de ce programme (= processus)
  - Stockée pour exécution dans sa mémoire.
  - + les données nécessaires à son exécution en les puisant
- Le SE doit impérativement gérer l'accès à la mémoire afin d'éviter qu'une donnée soit écrasée par une autre.
- Il est nécessaire d'empêcher à tout prix qu'un programme puisse écrire dans l'espace de données de l'autre (exécution dans un espace protégé).

Windows Task Manager

File Options View Help

Processes Performance App History Startup Users Details Services

| Process                          | Status | 7% CPU | 28% Memory | 1% Disk | 0% Network |
|----------------------------------|--------|--------|------------|---------|------------|
| <b>Applications (8)</b>          |        |        |            |         |            |
| COM Surrogate                    |        | 0%     | 19.9 MB    | 0 MB/s  | 0 Mbps     |
| Internet Explorer                |        | 0%     | 61.3 MB    | 0 MB/s  | 0 Mbps     |
| Microsoft Expression Blend ...   |        | 0%     | 45.8 MB    | 0 MB/s  | 0 Mbps     |
| Microsoft Visual Studio 11 E...  |        | 0%     | 37.4 MB    | 0 MB/s  | 0 Mbps     |
| Paint                            |        | 0%     | 7.9 MB     | 0 MB/s  | 0 Mbps     |
| Tweet@rama                       |        | 1.6%   | 46.5 MB    | 0 MB/s  | 0 Mbps     |
| Windows Explorer                 |        | 1.0%   | 52.2 MB    | 0 MB/s  | 0 Mbps     |
| Windows Task Manager             |        | 1.9%   | 10.6 MB    | 0 MB/s  | 0 Mbps     |
| <b>Background processes (17)</b> |        |        |            |         |            |
| Antimalware Service Execut...    |        | 0%     | 55.1 MB    | 0 MB/s  | 0 Mbps     |
| Bluetooth Radio Managem...       |        | 0%     | 0.5 MB     | 0 MB/s  | 0 Mbps     |
| COM Surrogate                    |        | 0%     | 0.9 MB     | 0 MB/s  | 0 Mbps     |
| Fast User Switching Utility S... |        | 0%     | 0.5 MB     | 0 MB/s  | 0 Mbps     |
| IPoint.exe                       |        | 0%     | 3.4 MB     | 0 MB/s  | 0 Mbps     |
| Microsoft Windows Search ...     |        | 0%     | 16.2 MB    | 0 MB/s  | 0 Mbps     |
| Microsoft.VisualStudio.Perf...   |        | 0%     | 3.3 MB     | 0 MB/s  | 0 Mbps     |
| RemoteTestRunnerService...       |        | 0%     | 1.6 MB     | 0 MB/s  | 0 Mbps     |
| Runtime Broker                   |        | 0%     | 8.6 MB     | 0 MB/s  | 0 Mbps     |

^ Fewer details End Task

# Le système de fichiers

- ❑ Permet de stocker les fichiers sur un support de stockage.
- ❑ Propose à l'utilisateur une vision hiérarchisée des fichiers sous forme arborescente la plupart du temps (aide à la représentation mentale et donc à une meilleure recherche des informations).
- ❑ Il permet de créer, détruire, et protéger les fichiers.
- ❑ Il répartit ensuite ces fichiers sur les différents supports physiques de stockage (*disquettes – obsolète – , disques durs, Clé USB, Cartes SD, etc.*).
- ❑ C'est une sorte de grand catalogue dans lequel les fichiers sont stockés et rangés.



# Les entrées/sorties

---

- ❑ Permettent aux processus en mémoire de communiquer avec autre chose qu'eux même.
  - l'enregistrement de données sur un disque dur,
  - l'envoi d'information sur une imprimante, sur une carte vidéo,
  - la récupération des caractères tapés au clavier,
  - la lecture et l'écriture de toute information sur un périphérique les acceptant (c'est à dire, un périphérique d'entrée/sortie).

## Remarque :

Ces 4 points essentiels (gestion des processus, gestion de la mémoire, système de fichiers, entrées/sorties ) qui permettent de qualifier un programme de système d'exploitation sont de nos jours épaulés par d'autres caractéristiques permettant :

- ❑ de relier des machines entre elles (**réseaux**),
- ❑ de proposer une interface utilisateur graphique facilitant la communication homme/machine (**multifenêtrage - IHM**).

# La gestion des processus



- Un processus au sein d'un SE correspond à une instance d'un programme en cours d'exécution
  - Un seul processus en mémoire (mono tâche)
  - Exécuté en parallèle avec d'autres (multitâches)
- S'il est multitâches le SE doit permettre
  - de créer ces processus (= *instances de programme, éventuellement multiples*),
  - de les gérer,
  - de les synchroniser,
  - de leur permettre de communiquer ensemble.



# Les processus

---

## □ Définition

Un processus est un programme en cours d'exécution (*vue dynamique*)

## □ Il en existe de deux types :

- les **processus systèmes**, entièrement gérés par le SE,
- les **processus utilisateurs** qui sont lancés soit pas le biais d'une commande dans un interpréteur de commande ou soit par une interface graphique par « double-click ».

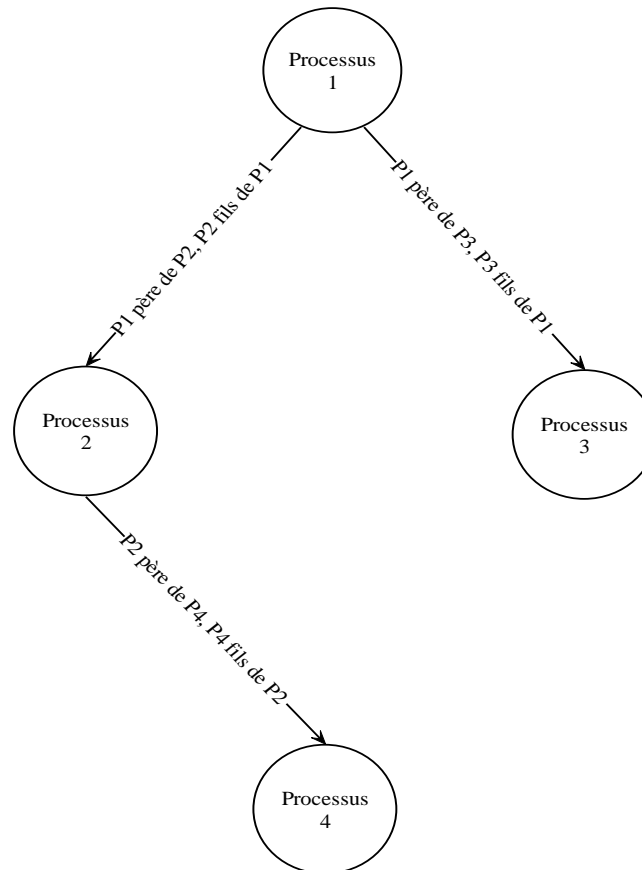
## □ Chaque processus a la possibilité de lancer d'autres processus.

- le processus d'origine, est appelé le processus père
- les processus créés sont appelés processus fils.
  - A leur tour, les processus fils peuvent également lancer d'autres processus fils, pour lesquels ce dernier sera le père.
- Au démarrage du SE, il n'existe qu'une seul processus père de tous les autres

## □ Notions liées aux processus :

- système multitâche, temps partagé.
- communication entre processus: signal, pipe, socket
- « qui est le père ? », « qui est le fils ? »
- suppression d'un processus (suicide, assassinat)

# Les processus



- Structure arborescente des processus

# Structure des processus

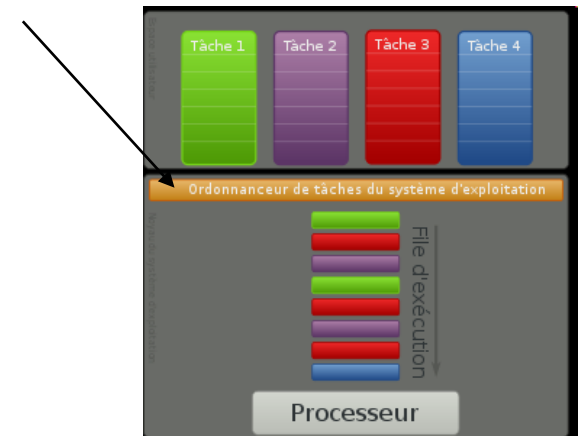
---

- ❑ Un processus n'est pas uniquement un programme, il est composé d'un espace de travail en mémoire comprenant 4 zones de :
  - **pile** : elle contient les appels de fonctions avec les paramètres et les variables locales. Cette zone n'a pas de taille fixe.
  - **données** : elle contient l'ensemble des données du programme (variables globales/statiques et les allocations dynamiques). Cette zone n'a pas de taille fixe.
  - **code** : elle correspond aux instructions en langage d'assemblage (assembleur) du programme à exécuter.
  - **contexte**, contrairement aux 3 précédentes, elle n'est pas directement liée au programme, mais contient l'ensemble des informations nécessaires à la gestion du processus. Ce contexte contient (entre autres) : le compteur ordinal pour savoir où on en est dans l'exécution et pour savoir où reprendre dans le cas d'une utilisation multitâche, la date de création, les adresses où trouver les données et la pile, son PID (identifiant unique dans le système), le PID de son père, ... Cette liste est loin d'être exhaustive.



# Structure des processus

- Dans un système multitâche préemptif, lorsqu'il est nécessaire de changer de processus en cours, le contexte est sauvegardé de manière à pouvoir être récupéré lorsque cette tâche sera de nouveau en cours d'exécution et ainsi continuer à l'endroit où elle a été interrompue.
- Lorsqu'il est nécessaire de changer de processus, un processus système appelé l'ordonnanceur produit une information appelée « **Interruption** ».



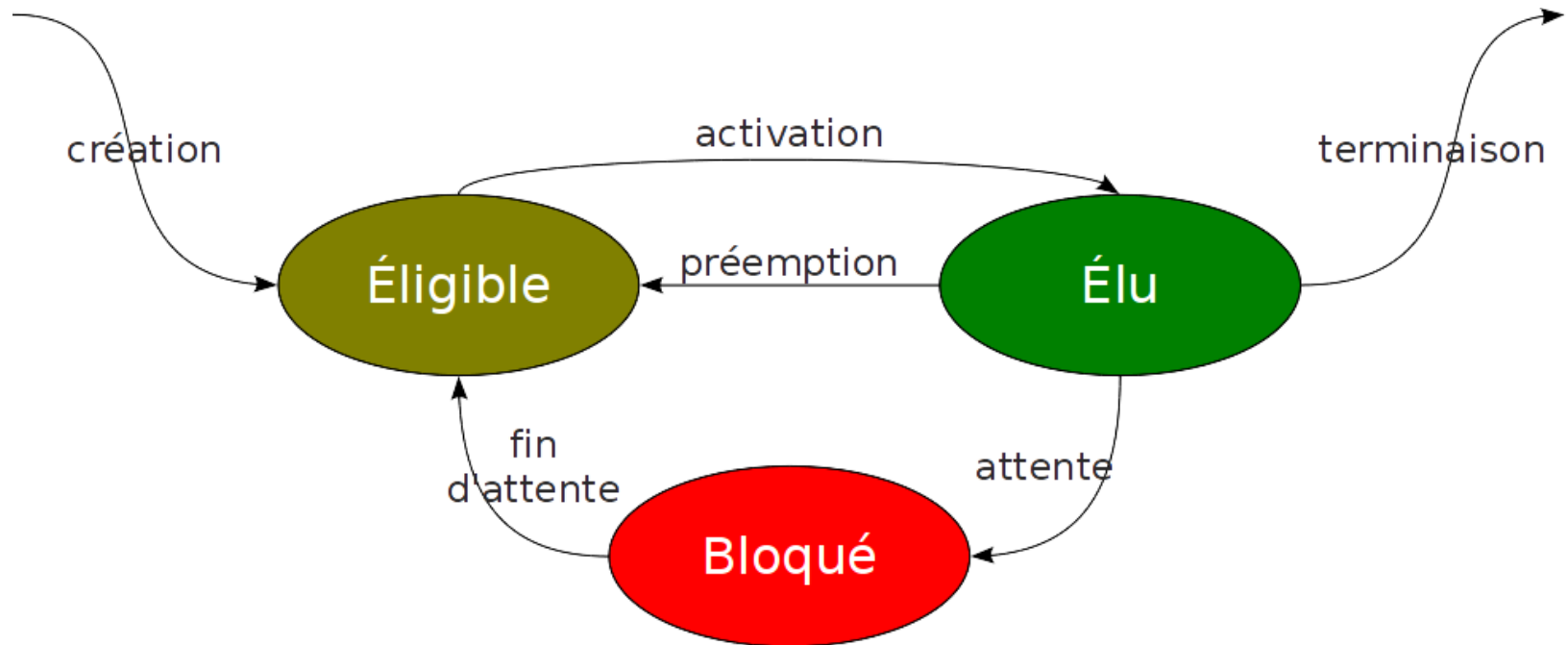
# États des processus

---

- Les processus peuvent donc avoir plusieurs états :
  - **Running** : actif en mémoire (*élu*)
  - **Ready** : suspendu en attente d'exécution (*prêt*)
  - **Waiting** : bloqué en attente d'une ressource (*bloqué*), par exemple en attendant la frappe d'un caractère au clavier
  - **Zombie** : état très particulier, le processus est mort seul reste son ID dans la table des processus en attente d'être lu ...

# États des processus

---



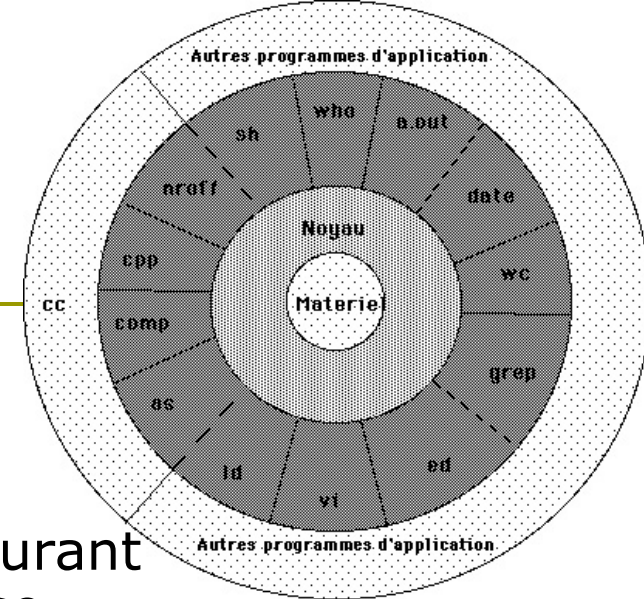
# Ordonnancement des processus

---

- ❑ L'ordonnancement des processus permet la réalisation de manière optimum du SE en :
  - Maximisant l'utilisation du processus,
  - Tentant d'être équitable avec les processus,
  - En étant rapide,
  - Assurant la gestion (éventuellement évolutive) des priorités.

# Algo du tourniquet

- ❑ Le plus utilisé, l'un des plus fiables.
- ❑ Lorsqu'un processus est élu, il s'exécute durant un quantum de temps ou jusqu'à ce qu'il se mette en attente d'une ressource (une E/S par exemple).
  - Lorsque le temps est épuisé ou s'il est bloqué, le suivant dans la liste des processus prêts est à son tour élu, le processus qui vient de passer la main est mis en dernière position dans la liste des processus.
- ❑ Lors de la mise en œuvre de l'algorithme du tourniquet, la seule chose qui est paramétrable est le quantum de temps situé en général entre 100 à 200ms.



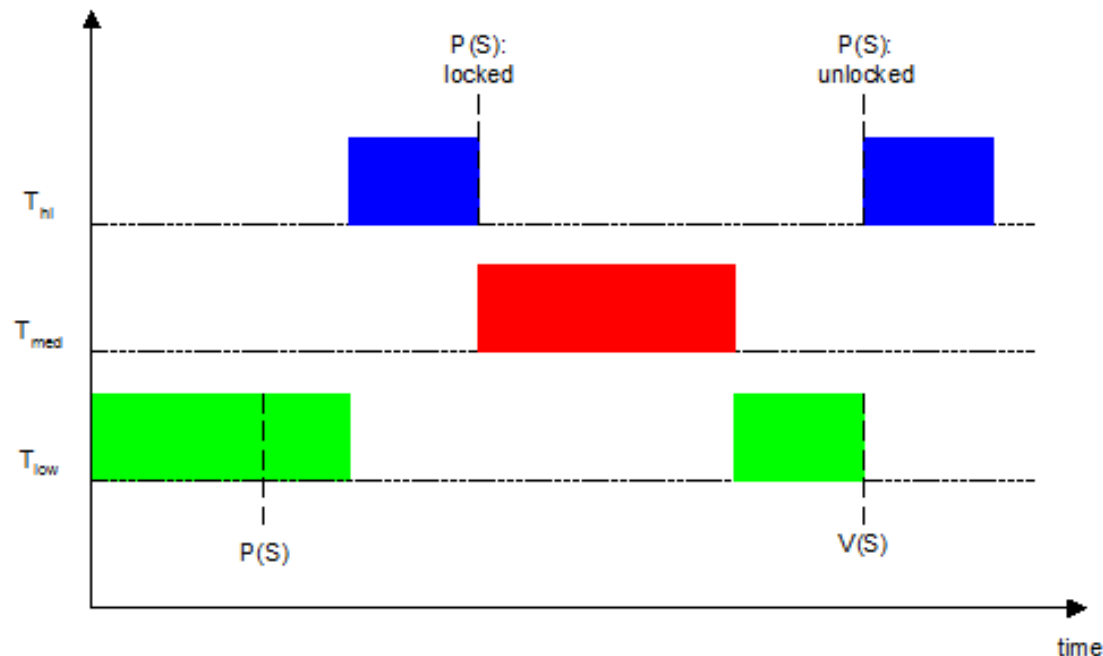
# Algo du tourniquet

---

- ❑ Il est parfois nécessaire de gérer des priorités dans les processus, certains devant être élus plus souvent.
  - le processus élu n'est pas forcément le suivant, mais le processus prêt suivant de plus haute priorité.
- ❑ Dans cette optique, on imagine aisément que certains processus de priorité basse ne seront jamais élus ! Tout est prévu.
  - Un processus venant d'être exécuté retourne dans la file avec une priorité abaissée.
  - Un processus n'ayant pas été élu depuis un certain temps verra sa priorité monter jusqu'à son exécution.
- ❑ A titre d'exemple, Windows Server gère jusqu'à 32 niveaux de priorité, Unix gère 9 niveaux.

# Exemple

- $T_{hi}$  de haute priorité ;  $T_{med}$  de priorité moyenne ;  $T_{low}$  de faible priorité.
- $T_{low}$  s'exécute
- $T_{hi}$  arrive parmi la liste des tâches éligibles et logiquement s'exécute. Cette tâche veut également accéder à la ressource, mais elle se retrouve bloquée.
- Maintenant, c'est au tour de  $T_{med}$  d'entrer dans la liste des tâches éligibles :  $T_{hi}$  est en attente de  $S$ ,  $T_{low}$  est moins prioritaire, c'est donc  $T_{med}$  qui s'exécute



# En résumé

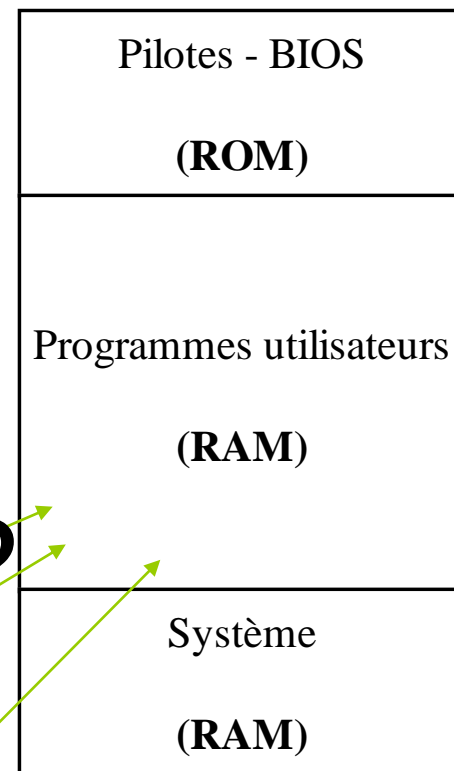
---

- ❑ Un processus est un programme en cours d'exécution.
- ❑ Un processus a besoin de ressources matérielles :
  - l'unité centrale,
  - la mémoire centrale
  - l'accès à des périphériques d'entrées/sorties.
- ❑ Ses caractéristiques statiques sont :
  - Un numéro unique : PID (Process IDentifier),
  - Un propriétaire déterminant les droits d'accès du processus aux ressources : ouverture de fichiers...
  - Un processus parent dont il hérite la plupart des caractéristiques,
  - Un terminal d'attache pour les entrées/sorties.
- ❑ Ses caractéristiques dynamiques sont :
  - Priorité, environnement d'exécution...
  - Quantité de ressources consommées (temps unité centrale utilisé...).



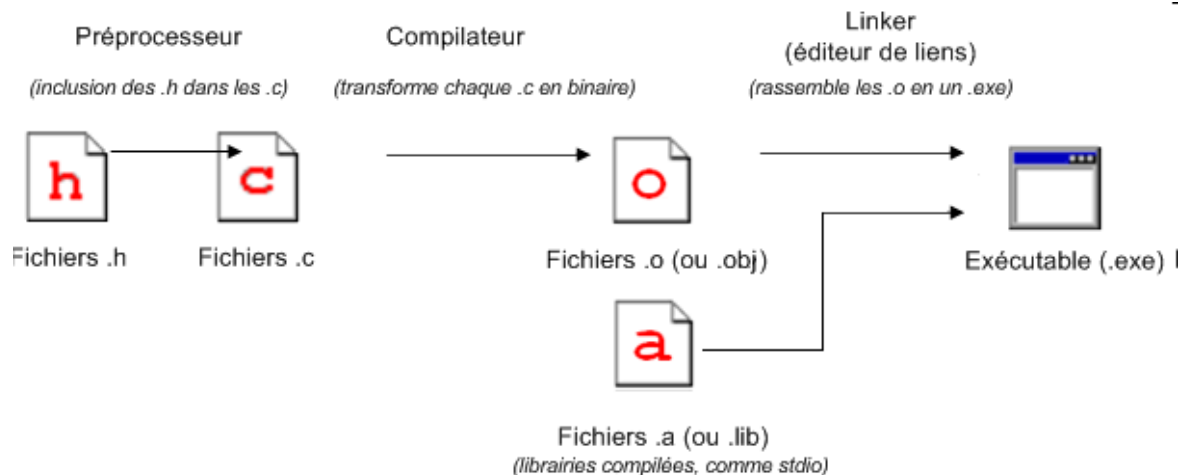
# La mémoire

- Lorsqu'un programme est exécuté, il (avec ses données) nécessite d'être transféré de son lieu de stockage (*disque dur, disquette, etc.*) vers la mémoire centrale de l'ordinateur. Le fait d'être en mémoire permet au processeur de l'exécuter.



# Écriture d'un programme : étapes

- Lorsque nous utilisons des programmes (traitement de texte, système d'exploitation, jeux, etc.), c'est derniers sont le résultat d'un ensemble d'étapes bien définies :
  - Le programmeur crée un fichier, appelé fichier source et écrit son programme dans un langage de programmation (*C, C++, Java, etc.*)
  - Le code du programme écrit dans un langage appelé langage de haut niveau (*C, C++, Java, etc.*) est transformé/traduit en module objet contenant la traduction du langage en langage machine, seul langage compréhensible par le processeur et donc par extension, par l'ordinateur (*chaque processeur possède son propre langage*).
  - Le code objet est ensuite passé à la moulinette de l'éditeur de liens qui permet d'ajouter les appels à des librairies (programmes ou bouts de programmes annexes). Ces appels de librairies permettent d'utiliser et d'intégrer dans un programme des fonctionnalités déjà écrites.



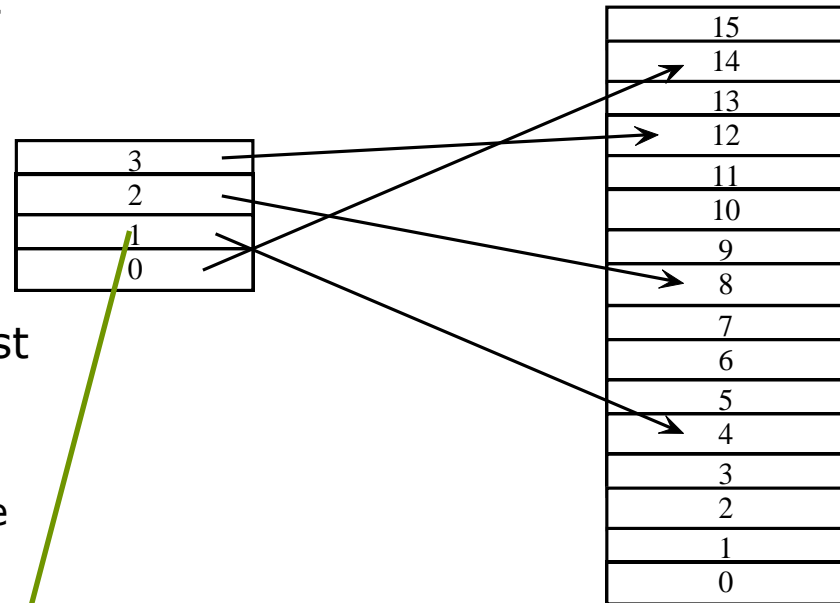
# Écriture d'un programme : étapes

---

- ❑ Les programmes compilés ne contiennent pas d'adresses absolues en RAM.
  - Un tel fonctionnement impliquerait que ce dernier soit toujours situé dans le même emplacement mémoire ce qui dans le cas d'un SE multitâche est une hypothèse quasi-irréalisable.
  - Chaque programme possède donc des adresses relatives (le décalage) à l'adresse 0.
  - Les adresses sont ensuite translatés par rapport à l'adresse 0.
- ❑ Un entête est également ajouté permettant de connaître le SE cible (*magic number*), mais surtout la taille des zones de données, pile, ...) et enfin le point d'entrée du programme, c'est à dire à qu'elle instruction il doit démarrer.

# Exécution d'un programme

- Lorsqu'une demande d'exécution de programme est réalisée
  - le système scrute la RAM afin d'y trouver un espace libre.
  - lorsqu'un espace d'un seul bloc n'est pas disponible, il peut en trouver plusieurs dont la taille globale est égale à la taille nécessaire (pagination).



- Le problème est que la mémoire RAM est en général toujours insuffisante,
  - donc au bout d'un certain temps, il n'y a plus de mémoire disponible.
  - des mécanisme de gestion de la mémoire sont mis en œuvre.

Avant sa mise en mémoire,  
un programme est découpé  
en « blocs »

# Gestion de la mémoire

---

- ❑ Le SE garde une image représentant l'occupation de la mémoire.
  - Il la découpe en unités de tailles fixes.
  - Plus l'unité est petite, meilleure la gestion de la mémoire est faite, mais plus l'image de la mémoire est grande (donc occupe plus de place et demande plus de temps)
  - Réciproquement, plus la taille de l'unité est grande, moins on fractionne la mémoire et donc sa gestion demande une image moins grande. L'inconvénient est la perte de mémoire lorsque ce dernier n'est pas de taille multiple de l'unité.
- ❑ Lorsqu'un programme n'a plus de raison d'être stocké en mémoire, on libère son espace en mémoire qui redevient disponible.

# Gestion de la mémoire

---

- Au fur et à mesure de l'exécution du SE et de programmes, de nombreuses allocations/libérations de mémoires sont réalisées ce qui a pour conséquence de **fragmenter** la mémoire et par extension, les programmes à charger impliquant plus de manipulations.
  - L'exemple le plus flagrant sont les programmes gérant le multi-fenêtrage dans les SE.

# Fragmentation

## Fragmentation

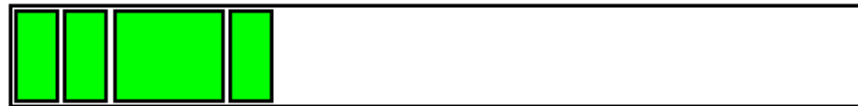


Figure 1.



Figure 2.



Figure 3.



Figure 4.

- => La mémoire est régulièrement « compactée », ce qui consiste à regrouper en début ou en fin de mémoire les processus en cours de manière à libérer de grandes zones de mémoire continues.

# Le va-et-vient (swap)

---

- Lorsque l'ensemble des processus ne peut tenir en mémoire, on doit en déplacer certains temporairement (jusqu'à leur prochaine utilisation) sur une mémoire secondaire (*swap*) du disque.
- L'avantage de cette solution (implémentée dans Windows et Unix) est sa simplicité. L'inconvénient est la lenteur engendrée lorsque le swap devient trop important.
  - Les temps d'accès calculés en *ns* pour une mémoire centrale (RAM)
  - Ils se calculent en *ms* pour une mémoire secondaire de type disque dur, c'est à dire 1000 fois moins vite.



# Gestion de la mémoire

---

## □ La mémoire virtuelle

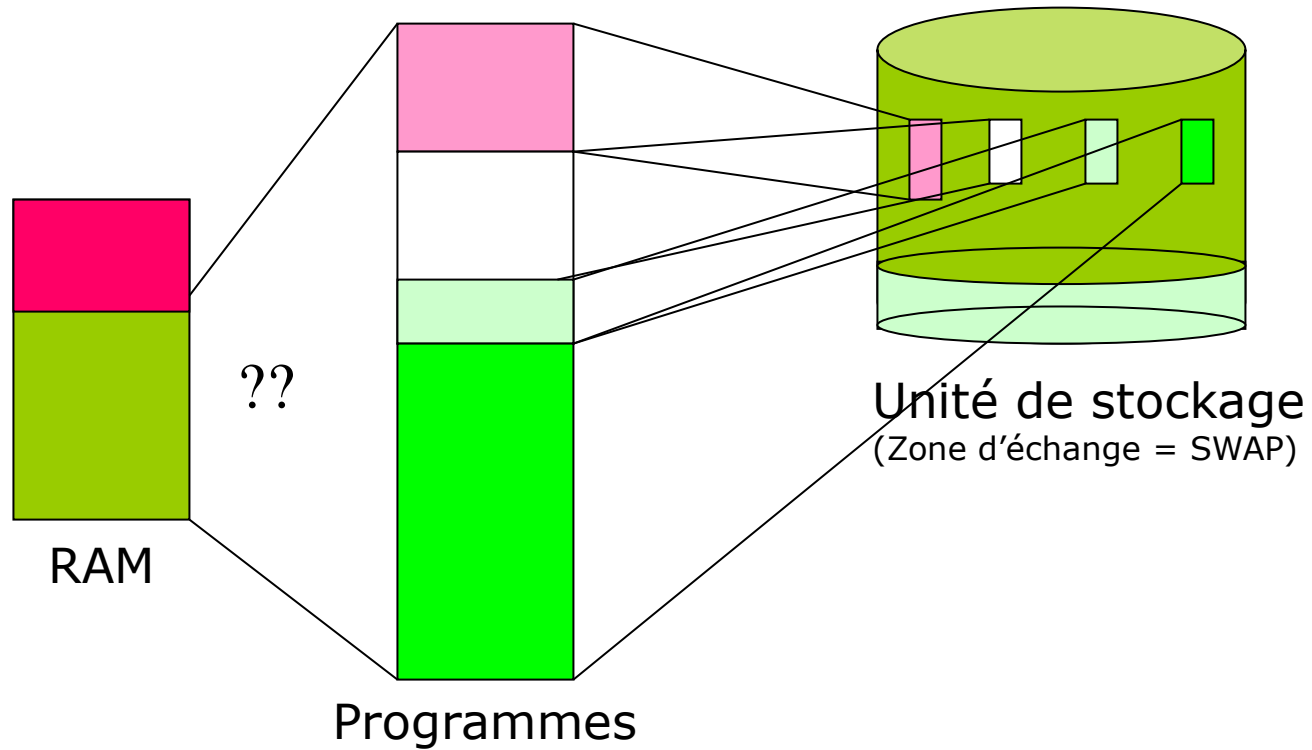
- Idée : Supprimer la contrainte de **dimension** de la mémoire.
- Mise en œuvre : On dote l'ordinateur de deux niveaux de mémoire : la mémoire centrale (performante et chère) et la mémoire de masse (le disque) moins performante mais de dimension pratiquement illimitée.
- Performances liées aux : *seek time* ; *latency time* ; *transmission time*. L'optimisation est gérée par le système d'exploitation qui met en place des stratégies.
- Enfin, est mis en place une stratégie (algorithme) de choix de la page dont dépend beaucoup la rapidité du système.
- **Avantages** : Beaucoup de place !
- **Inconvénients** : SWAP ou accès disque à optimiser + lenteur induite

Remarque 1 : Vient en combinaison d'autres techniques de gestion de la mémoire.

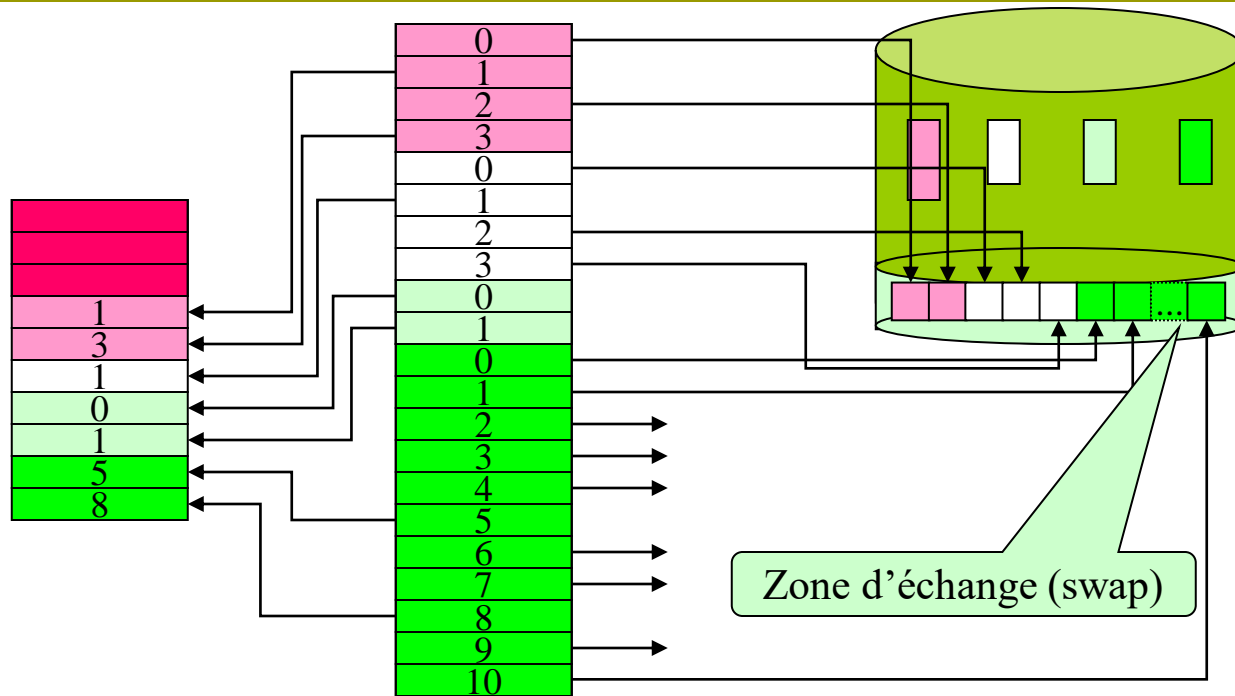
Remarque 2 : D'autres stratégies ou combinaisons de stratégie existent ... !

# Mémoire virtuelle

---





# Mémoire virtuelle



- ❑ L'espace mémoire est découpé en pages. Seules les pages nécessaires sont en mémoire physique.
- ❑ Les autres pages sont copiées dans une zone du disque. Elles seront échangées (*swapped*) au besoin.

# Le système de fichiers

---

- Au sein d'un SE, deux types de fichiers cohabitent :
  - les fichiers de type programme, 
  - les fichiers contenant les objets manipulés par les programmes. 
  - De manière générale, on parle de fichiers programme et
- Le type dépend du contenu du document :
  - nature de l'information (texte, image, son...)
  - norme de codification interne du document (mp3, doc, html,...)
- Exemple : le fichier **word.exe** est un fichier programme (non lisible), alors que **cv.docx** est un fichier de données.

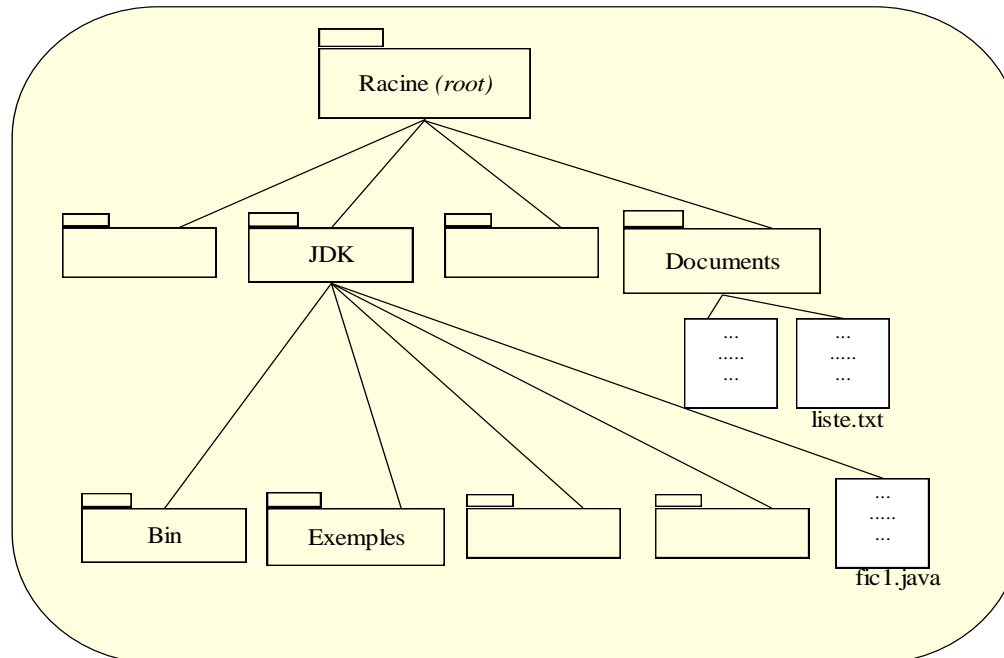
# Organisation des fichiers

---

- ❑ Les fichiers sont organisés dans un système de fichiers.
- ❑ Ce système de fichiers permet l'organisation la plus souvent hiérarchique (sous forme d'arbres)
- ❑ Il permet au SE de les organiser sur le disque, de pouvoir y accéder, les retrouver, ...
- ❑ Du côté utilisateur, le système de fichiers permet de proposer une interface permettant à ces utilisateurs « de s'y retrouver ».
- ❑ Tout fichier appartient à un groupe de fichiers, tout groupe de fichiers appartient à un autre groupe de niveau supérieur jusqu'à remonter au groupe d'origine appelé **racine**.

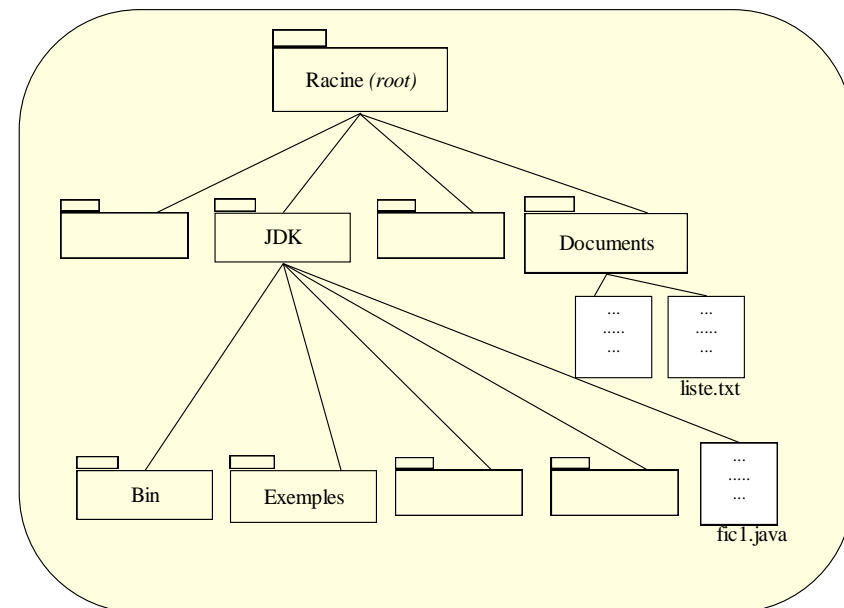
# Organisation des fichiers

- ❑ Les nœuds de l'arbre = *répertoires (directory)* ou *dossiers (folder)*
- ❑ Les feuilles sont appelées *fichiers*.
- ❑ Les répertoires sont eux même des fichiers contenant l'ensemble des noms des fichiers et des sous-répertoires qu'ils contiennent.



# Organisation des fichiers

- ❑ Tout répertoire contient deux fichiers un peu particuliers notés « . » et « .. », permettant d'accéder à l'ensemble des fichiers de manière relative (*depuis le répertoire courant*) au lieu de donner le chemin absolu (*depuis la racine* – *chemin démarrant par le caractère « / »*).
- ❑ Chemin absolu
  - windows/dos : `C:\JDK\fic1.java`
  - unix : `/JDK/fic1.java`
- ❑ Chemin relatif depuis *Exemples*
  - `../fic1.java`
  - `../../Documents/liste.txt`



# Accès et stockage des fichiers

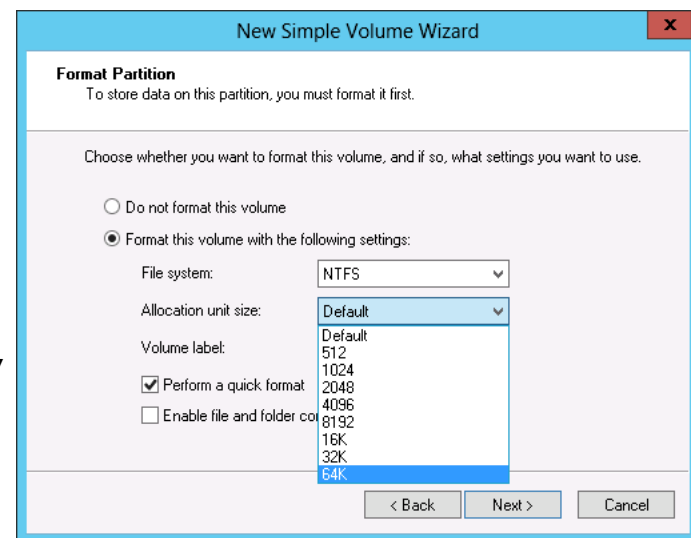
---

- ❑ Les fichiers stockés sur une unité de stockage ne sont pas enregistrés d'un seul tenant mais sont découpés en blocs de taille fixe (nombre entier et fixe d'octets), selon un principe analogue à la pagination des programmes en mémoire.
- ❑ L'accès à un fichier se résume à accéder au premier bloc puis à les lire successivement jusqu'au dernier.
  - soit on connaît à l'avance le nombre de blocs du fichiers,
  - soit un des blocs sera discriminant et signifiera qu'il est le dernier. Il existe pour cela une table (appelée table d'allocation des fichiers) permettant au système de retrouver tous ces blocs.
- ❑ De la même manière qu'il est possible pour un processus d'être « éclaté » en différentes pages à des adresses contiguës, un fichier peut également être éclaté sur le support de stockage
  - fragmentation.
  - plus un support est fragmenté, plus long sera l'accès à l'ensemble du contenu des fichiers



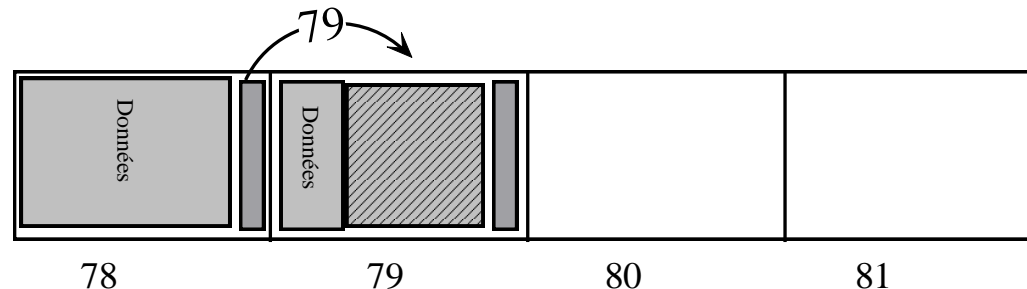
# Découpage en blocs

- ❑ Avant d'écrire sur une unité de stockage, il faut la formater,
  - lui donner une certaine organisation.
  - consiste à donner une unité de taille pour les blocs qui devront être écrits.
- ❑ Deux politiques peuvent être utilisés :
  - des blocs très petits, ça minimise les pertes d'espace dans le cas de fichiers petits, mais ça augmente le temps d'accès (il faudra accéder à plus de blocs, donc plus de déplacements de la tête de lecture), de plus, la table de référencement des blocs permettant de reconstituer les fichiers écrits sur plusieurs blocs sera également plus grande.
  - L'autre politique est de prendre des blocs plus grands, du coup, la table est moins grande, le nombre d'accès est inférieur et donc plus rapide. L'inconvénient est la plus grande perte de blocs.
- ❑ Une taille pour un bloc se situe autour de 32/64Ko.



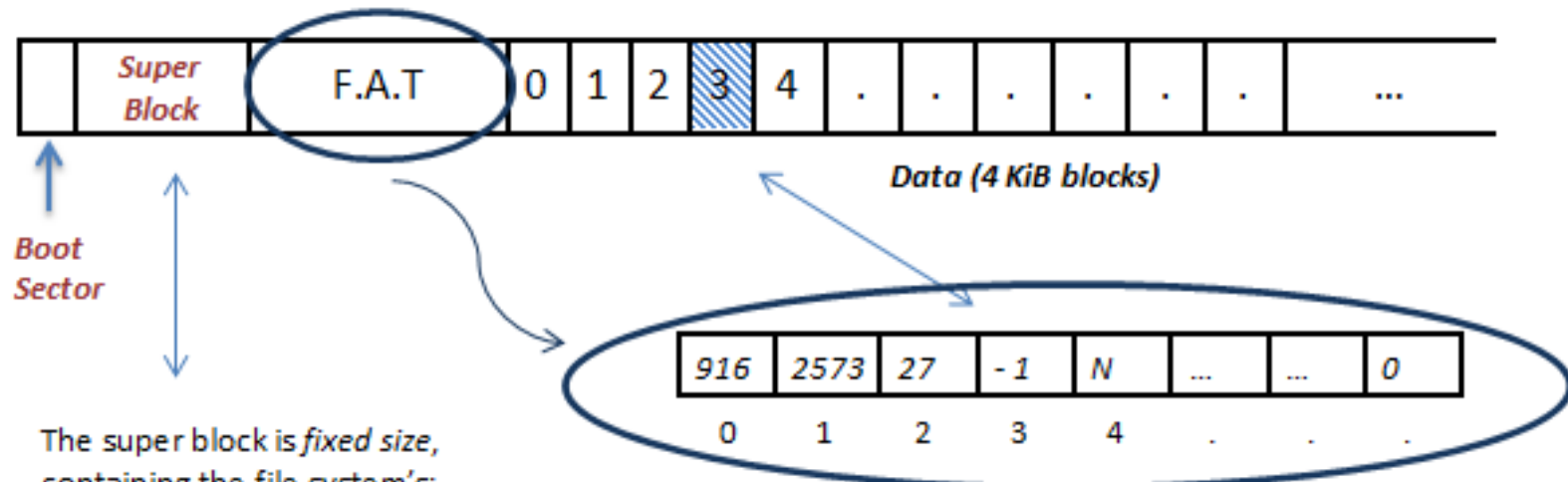
# Exemple de stockage

- ❑ Prenons un exemple avec un fichier appelé *fichier1.txt* de 7440 octets, soit 7Ko et 272 octets restant, avec des blocs de 1 Ko (soit 1024 octets).
  - Fichier en un bloc
    - ❑ *Fichier1.txt* | 47 | 54      8 blocs occupés (pas de 1/2 blocs)
  - Fichier découpé
    - ❑ *Fichier1.txt* | 47 | 48      **2 blocs**
    - ❑ *Fichier1.txt* | 12 | 15      **3 blocs**
    - ❑ *Fichier1.txt* | 120 | 120      **1 bloc**
    - ❑ *Fichier1.txt* | 78 | 79      **2 blocs**
- Plus un fichier est "tronçonné", plus il nécessitera d'entrées dans la table, et plus volumineuse elle sera.



# Exemple de FAT

File Allocation Table (F.A.T) file system



The super block is *fixed size*, containing the file system's:

- Size
- Version number
- Number of blocks in use
- Root directory block #

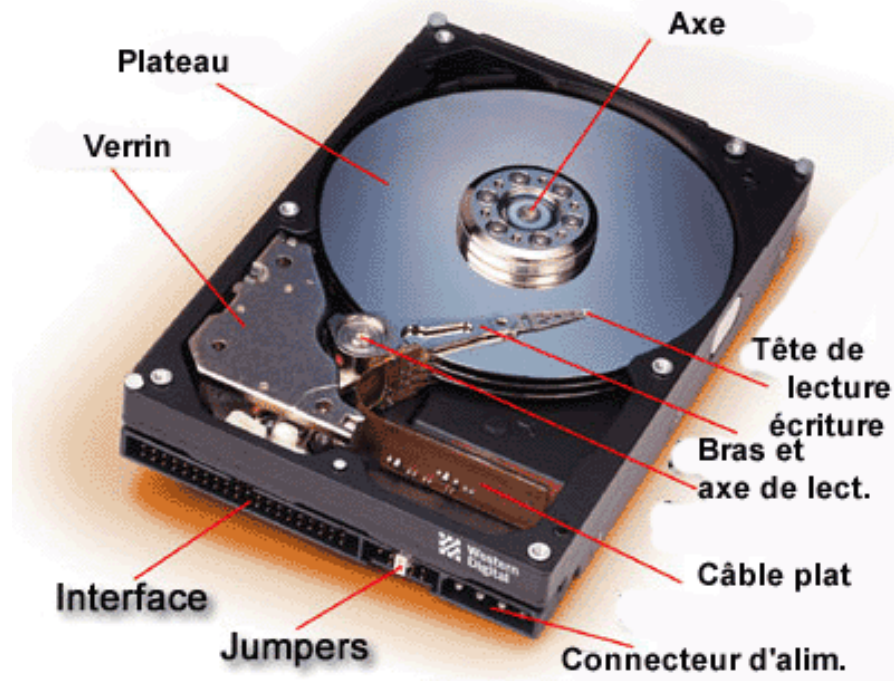
Each block in the F.A.T. is a pointer to the next block in the file system

- 1 Free block in file system
- 0 EOF (last block of current file)
- N Next block in this file is block N

*Having a F.A.T. localizes information and allows data blocks to be a power-of-2 size (instead of leaving 4 bytes in each block for a next field)*

# Exemple de stockage

- Plus important encore peut être, c'est le nombre d'accès nécessaires pour obtenir l'ensemble des informations du fichier (ici, 4 accès). Le temps d'accès d'un disque dur se calcul en *ms*. Enfin, on voit également qu'ici, nous perdons 752 octets (1024-272) puisqu'il est impossible de scinder un bloc et d'y faire contenir d'autres "bouts" de fichiers.

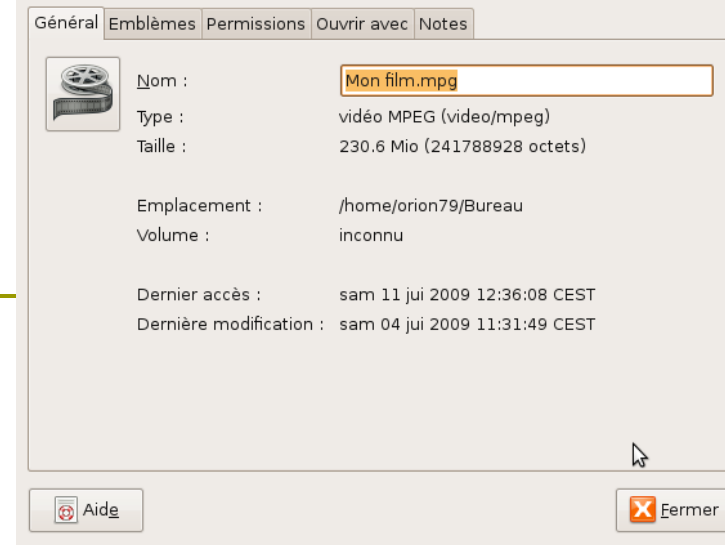


# Exemple de stockage

- Une autre possibilité est de ne pas découper à priori, mais de réserver un espace dans le bloc qui dira si la lecture est terminée, ou s'il faut aller voir un autre bloc.
- **Exemple**
  - Fichier1.txt | 47
  - bloc 47 : 1020 octets de données + 4 octets pour dire que le prochain est le 48ème,
  - bloc 48 : 1020 octets de données + 4 octets pour dire que le prochain est le 12ème,
  - ...
  - bloc 79 : 300 octets de données ( 7440 - ( 7 x 1020 ) ) + un marqueur de fin.
- L'avantage est la simplification de la table d'entrée, l'inconvénient est que pour connaître la taille d'un fichier, il faut parcourir l'ensemble des blocs.
- Il existe de nombreuses autres variantes optimisées de ce type d'organisation, toutes se rapprochant plus ou moins à un mixage de ces deux méthodes.
- **Si on a 4 octets pour donner le nom du bloc suivant, ceci signifie que l'on pourra adresser des fichiers de plus de  $2^{(8 \times 4)} = 2^{32}$  octets (4Go)**
  - Si on veut plus...il faudra baisser la taille des données pour augmenter la taille de codage



# Les fichiers



- ❑ Dans chaque système d'exploitation (Unix, Windows ou autre), les fichiers ne sont pas uniquement représentés par leur nom.
  - Lors de chaque création/modification, un certain nombre d'informations (appelées attributs) sont mises à jour
    - ❑ Le type (exécutable ou non),
    - ❑ la date et l'heure de la dernière modification (ou de création),
    - ❑ la taille,
    - ❑ le numéro du premier bloc,
    - ❑ l'extension (uniquement sous windows)
    - ❑ et éventuellement les droits (Windows NT/2000/XP et Unix).

# Les entrées/sorties

---

- ❑ Ce terme générique d'entrées/sorties est utilisé lorsqu'on parle d'échange d'informations entre un ordinateur et ses organes externes :
  - - clavier,
  - - disque dur, disquettes,
  - - imprimantes,
  - - mémoire
  
- ❑ Un ordinateur via son système d'exploitation ne communique pas directement avec les périphériques (organes externes).
  - il existe une multitude de périphériques (sans compter les modèles et marques différents pour un type de périphérique)
  - il est impossible que le SE les connaisse tous, ne serait-ce que parce qu'il en sort tous les jours des nouveaux.
  
- ❑ Le SE dispose de primitives "standards" afin de communiquer avec des types d'organes (imprimantes, disques durs, etc.)
  - entre les deux se placent un élément essentiel : le pilote (ou driver en anglais).
  - lorsqu'un ordinateur nécessite de réaliser des E/S avec un périphérique il discute via des primitives génériques avec le pilote correspondant, ce dernier réalisant l'interface entre l'ordinateur et le périphérique, c'est à dire qu'il a un rôle de traducteur ordinateur-périphérique.

# Dispositifs d'E/S

---

## □ Les disques

- Les disques sont constitués d'un ou plus généralement de plusieurs cylindres superposés (éventuellement double face).
  - Chacun de ces cylindre est divisé en un certain nombre de pistes,
  - Les pistes sont divisées en secteurs permettant de stocker un ou plusieurs blocs (c'est pourquoi un disque est formaté selon le SE qui va l'utiliser).
  - Lorsqu'on regarde les caractéristiques d'un disque dur, on regarde sa capacité, mais également le temps de positionnement moyen et le temps de rotation (exprimé en tours par minutes, ex. 5900, 7200, 10000, 15000 trs/mn).
    - Prix x 2
- Un bras piloté par le pilote du disque permet d'accéder aux différents cylindres/pistes et blocs contenant au final les données de fichiers.
- CTI -> Dépt. Info : 64 disques de 7200 à 15000 + SSD (cache)



# Dispositifs d'E/S

---

## □ Les terminaux

- représentent en général un écran connecté à un serveur.
- Peuvent être alphanumérique (seuls des caractères sont affichés), ou graphiques.
- Un terminal permet d'accéder à un ordinateur, localement ou à distance.

# UNIX



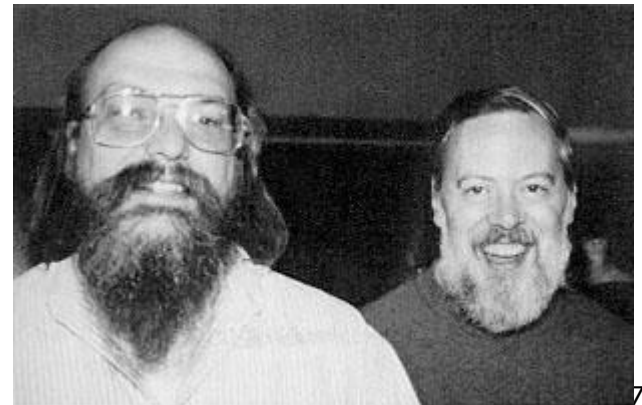
# Présentation d'UNIX

---

- ❑ Premier système "Unix"
  - A partir de 1965
  - Ken Thompson dans les laboratoires Bell AT&T (New Jersey)
  - Objectif : mettre au point un système d'exploitation interactif simple, nommé "Multics" (*Multiplexed Information and Computing System*)

# Présentation d'UNIX

- En avril 1969 les laboratoires AT&T décidèrent d'utiliser le système *GECOS* (*General Electric Comprehensive Operating System*) en lieu et place de *Multics*.
- *Ken Thompson* et *Dennis Ritchie* qui a rejoint l'équipe
  - faire fonctionner le jeu *space travail* sur une plus petite machine (un *DEC PDP-7*, *Programmed Data Processor* possédant uniquement 4K de RAM)
  - ils réécrivirent le système afin de créer une version réduite de *Multics* appelée *UNICS* (*UNiplexed Information and Computing Service*), rapidement contracté en *Unix*.

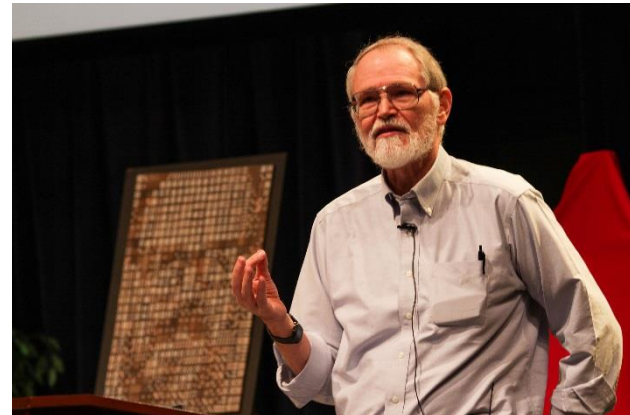


Kenneth Thompson (1943-) et  
[Dennis Ritchie](#) (1941-2011)

# Présentation d'UNIX

---

- ❑ 1er janvier 1970
  - considérée comme étant la date de naissance du système Unix
  - explique pourquoi toutes les horloges système des systèmes d'exploitation Unix démarrent à partir de cette époque.
- ❑ *D.Ritchie* participa également à la définition du C
  - il est considéré comme un de ses créateurs avec *B.W.Kernighan*
  - l'ensemble du système fût entièrement réécrit en langage C en 1973 et baptisé *Unix Time-Sharing System (TSS)*.



Brian Kernighan (1942- )

# Présentation d'UNIX

---

- ❑ Un décret datant de 1956 empêchait l'entreprise ATT, dont dépendait *Bell Labs*, de commercialiser autre choses que des équipements téléphoniques ou télégraphiques,
  - => distribution des sources d'UNIX en 1973 dans les universités à des fins éducatives.
- ❑ Dès fin 1977 des chercheurs de l'Université de Californie re-développèrent une version d'Unix à partir des sources fournies par AT&T afin de faire fonctionner le système sur leurs plates-formes VAX et la baptisèrent *BSD* pour *Berkeley Software Development*.
- ❑ Ainsi deux branches de développement des sources virent le jour :
  - La branche d'AT&T qui allait devenir **System V** de *UNIX System Labs* (USL)
  - **BSD** (*Berkeley Software Development*) développé par l'Université de Californie

# Présentation d'UNIX

---

- Un grand nombre d'UNIX-like furent développés :
  - **AIX**, Unix commercial basé sur *System V* développé en février 1990 par *IBM*
  - **Sun Solaris**, Unix commercial basé sur *System V* et *BSD* développé par *SUN Microsystems*
  - **HP-UX**, Unix commercial basé sur *BSD* développé à partir de 1986 par *Hewlett Packard*
  - **Ultrix**, Unix commercial développé par *DEC*
  - **IRIX**, Unix commercial développé par *SGI*
  - **Unixware**, Unix commercial développé par *Novell*
  - **Unix SCO**, Unix commercial basé sur *System V* développé dès 1979 par *Santa Cruz Operations* et *Hewlett Packard*
  - **Tru64 UNIX**, Unix commercial développé par *Compaq*

# Présentation d'UNIX

- ❑ En 1983 AT&T eût le droit de commercialiser son Unix, ce qui
- ❑ En 1985 un professeur hollandais, nommé *Andrew Tannenbaum*, développa un système d'exploitation minimal, baptisé *Minix*, afin d'enseigner la programmation système à ses étudiants.
- ❑ En 1991 un étudiant finlandais, Linus Torvalds, décida de concevoir, sur le modèle Minix, un système d'exploitation capable de fonctionner sur les architectures de type 386.
- ❑ Il baptisa ce système d'exploitation "[Linux](#)" et posta le message suivant sur le groupe de discussion *comp.os.minix* :
  - Hello everybody out there using minix - I'm doing a (free) operating system (just a hobby, won't be big and professional like gnu) for 386(486) AT clones.



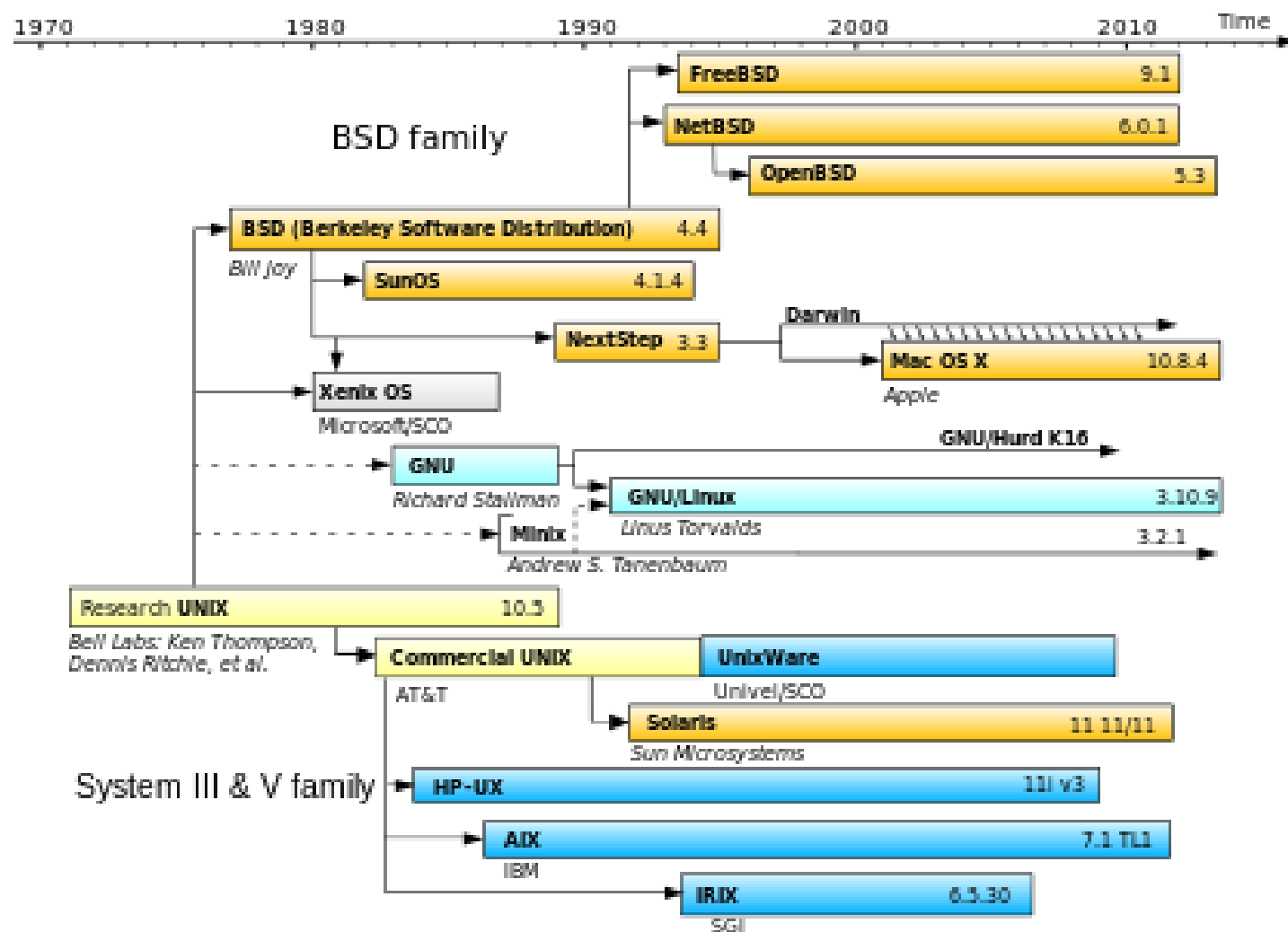
Andrew S. Tanenbaum  
(Hollande, 1944- )

Linus Torvald  
(Suède, 1969 - )





# Présentation d'UNIX



# Présentation d'UNIX

---

- ❑ Sur tout système Unix (et plus généralement, tout système récent), chaque utilisateur doit avoir été déclaré par l'administrateur du système - root (le chef !).
  - Chaque utilisateur possède un identifiant de connexion (*login*) et un mot de passe (*password*).

# Mots de passe

---

- ❑ Des personnes peuvent essayer d'accéder de façon illégale à un système UNIX.
- ❑ Un des moyens dont ils disposent est d'entrer sur un compte utilisateur en déterminant son mot de passe.
- ❑ Il est donc nécessaire que votre mot de passe respecte certaines règles de sécurité :
  - Il doit posséder au moins 7 caractères et contenir au moins une lettre majuscule, un chiffre et un caractère de ponctuation, et ceux-ci à l'intérieur et non en début ou fin de mot de passe. Généralement, seuls les 8 premiers caractères sont pris en compte ;
  - Il ne doit pas contenir des données relatives à votre identité comme votre nom d'utilisateur ou une information livrée par la commande finger (voir plus loin) ;
  - Il ne doit pas appartenir à des dictionnaires, tel quel ou sous sa forme canonique (c'est à dire, épuré de tous les caractères non-alphabétiques), à moins qu'il contienne des majuscules autres que le premier caractère ;
  - Il ne doit pas contenir des répétitions de caractère ;
  - Il doit être suffisamment simple pour s'en rappeler ; il ne faut pas le noter sur papier ou dans un fichier ni le donner à quelqu'un d'autre.
  - Ex. de "bons" mots de passe (*2 mots courts séparés par un ou plusieurs caractères de ponctuation ou chiffres*) : ble!1the ou si%@sol.

# Présentation d'UNIX

---

- ❑ La session de travail (appelé par la suite *session*) début une fois l'identification réalisée et se termine sur demande de l'utilisateur (*logout*).
- ❑ Sous Unix, la fenêtre permettant la saisie des commandes est appelée le terminal et exécute un interpréteur de commande appelé de manière générique le shell.
  - Il en existe plusieurs (*sh, ksh, bash, etc.*).
  - Attention, contrairement d'autres SE, le système Unix réalise une distinction entre les majuscules et les minuscules (la casse des lettres). Ceci est également valable pour la saisie de l'identifiant/mot de passe de connexion.
- ❑ Shell = interpréteur de commandes exécutant la boucle infinie suivante :
  1. Affichage de l'invite de commande ou prompt (« \$ ») d'attente de lecture au clavier.
  2. Lecture d'une commande (validée par RETURN ou ENTRÉE).
  3. Analyse syntaxique (découpage en mot).
  4. Interprétation des caractères spéciaux (\*, ?)
  5. Exécution de la commande et retour au début.

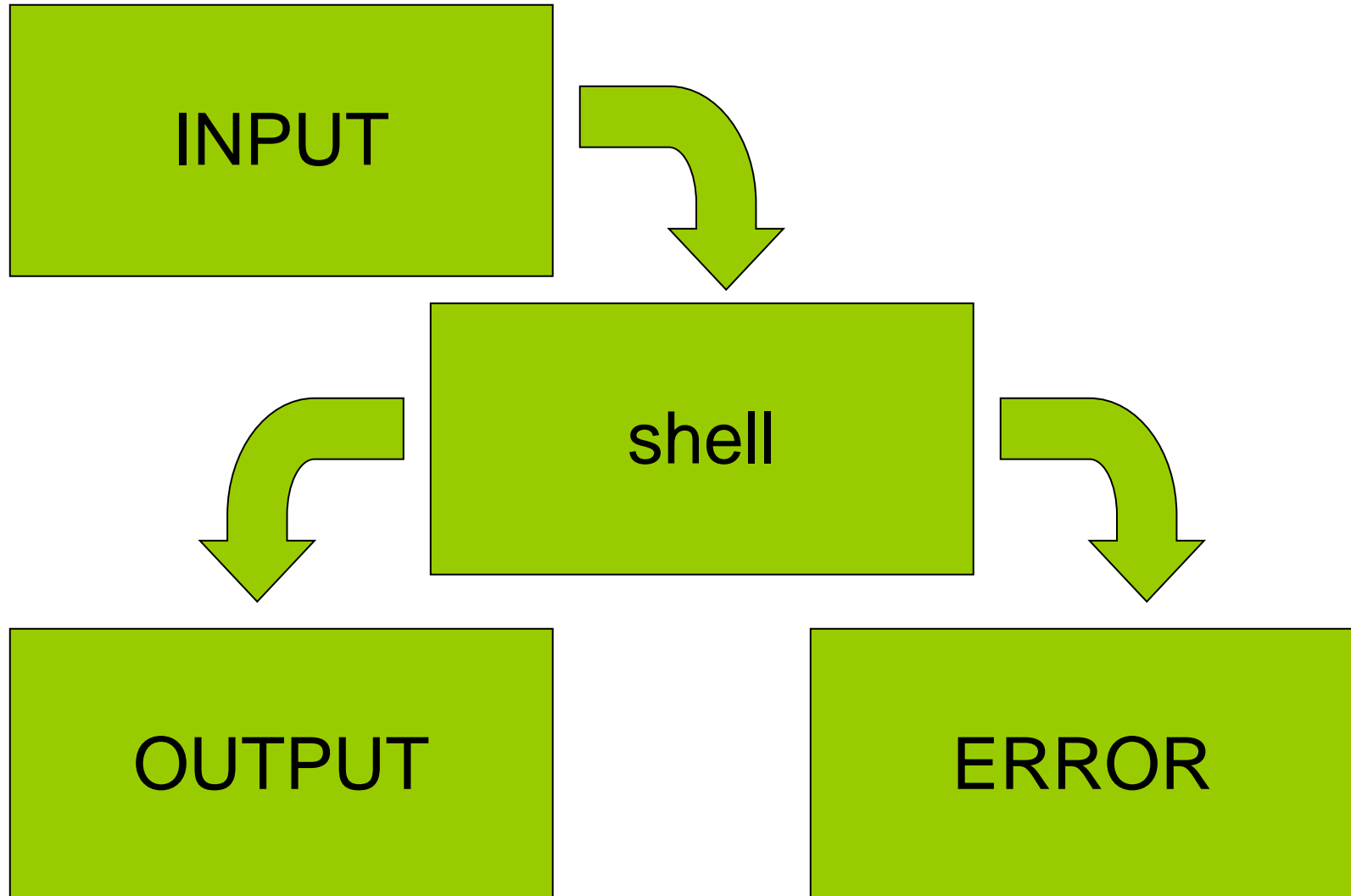
# Le Shell

---

- ❑ À la connexion, le shell dispose de trois flots de communication :
  - Entrée standard : `stdin` (numéro 0)
  - Sortie standard : `stdout` (numéro 1)
  - erreur standard : `stderr` (numéros 2)
- ❑ L'association par défaut de ces flots est l'écran pour *stdout* et *stderr*, et le clavier pour *stdin*.
- ❑ Une redirection est une modification de l'une ou de l'autre de ces associations. Elle est valable uniquement le temps de la commande sur laquelle elle porte.
  - Ainsi, la redirection de la sortie standard permet de récupérer le résultat dans un fichier11 :
  - *commande* > *nom\_fich*.
  - Attention, avec « > », si le fichier de redirection existe, son contenu initial est perdu.
- ❑ La redirection double (*commande* >> *nom\_fich*) permet de ne pas détruire le fichier existant, mais ajoute le nouveau contenu en fin de fichier.
- ❑ Moins utilisée que la redirection de la sortie standard, la redirection de l'entrée standard (*commande* < *nom\_fich*) permet à une commande d'utiliser comme données le contenu d'un fichier à la place d'une lecture clavier.
  - `wc -l .bash_profile`
  - 27 .bash\_profile ==> nombre de lignes suivi du nom du fichier
  - `wc -l < .bash_profile`
  - 27 ==> nombre de lignes seul

# Le Shell

---



# Commandes Unix

- ❑ La syntaxe générale d'une commande Unix est
  - **nom [-options] [argument 1...]**
    - ❑ *nom* est le nom de la commande ;
    - ❑ *options* représente une ou plusieurs options ;
    - ❑ *argument1* est le premier argument.
- ❑ Les options sont composées d'un seul caractère suivant un tiret. Il est possible d'accoler plusieurs options (donc, plusieurs caractères).
  - ❑ Par exemple, *-asli* pour les options *-a -s -l -i*.
- ❑ Si l'option demande un paramètre, ils sont séparés par un espace comme *-o fichier*.
- ❑ Les crochets désignent un élément facultatif, ils ne doivent donc pas être tapés.
- ❑ Dans une commande, chaque mot est séparé des autres par un espace ou une tabulation.

# Exemple de commandes

---

- ❑ Attention, comme TOUTE commande Unix, ces dernières s'écrivent en minuscule.
  - *hostname* : affiche le nom de la machine
  - *uname* : affiche le nom du système d'exploitation
  - *clear* : efface l'écran
  - *who* : liste les utilisateurs connectés sur le même site
  - *exit* : termine le shell (idem CTRL-D si configuré)
  - *su - nom\_connexion* : change temporairement l'identité de l'utilisateur en super-user (administrateur)
  - *man nom\_commande*
  - *mail*



# Commandes de positionnement dans l'arborescence

---

- ❑ *pwd* : affiche le nom absolu du répertoire de travail
- ❑ *cd* : change le répertoire de travail
  - Avec argument : nom du répertoire destination (en absolu ou en relatif)
  - Sans argument : retour au répertoire de connexion
- ❑ *ls* : liste les entrées d'un répertoire
  - Sans argument : les entrées du répertoire de travail
  - Avec arguments : les entrées repérées par les arguments
- ❑ Par exemple, *ls*, *ls /*, *ls /usr*, *ls .bashrc*
- ❑ Dans la commande précédente (*ls .bashrc*), le fichier dont le nom commence par un point (« . ») est appelé un fichier caché. La commande *ls* possède de nombreuses options : par exemple, *-a* pour visualiser les fichiers cachés, *-l* pour tous les attributs (type, droits, liens, propriétaire, groupe, taille, date, nom)

# Commande en relation avec les fichiers textes

---

- ❑ *cat fich, more fich* : affichage simple et page par page ;
- ❑ *head fich, head -n fich* : affichage des 10/n premières lignes ;
- ❑ *tail fich, tail -n fich* : affichage des 10/n dernières lignes ;
- ❑ *wc fich* : affichage du nombre de lignes, de mots, de caractères
  - options *-l, -w et -c* pour les nombres de lignes, de mots et de caractères

# Commandes de manipulation de fichiers

---

- ❑ *mkdir nomrép1 nomrép2, etc.*
  - Création d'un répertoire
- ❑ *cp chemin\_source chemin\_destination*
  - copie le fichier source à la destination (avec changement de nom éventuel)
    - ❑ option *-r*
    - ❑ copie récursive d'un ensemble de fichiers et de leur sous arborescence.
- ❑ *mv chemin source chemin destination*
  - idem précédemment, mais c'est un déplacement au lieu d'une copie.
- ❑ *rm [-r] fichier1 fichier2, ...*
  - suppressions des fichiers
  - option *-r* : supprime récursivement un ensemble de fichiers et leur sous arborescence.
- ❑ *rmdir*
  - Suppression d'un répertoire, doit être vide
- ❑ *ln*
  - **ln -s exemple.c toto.c**
  - crée un lien symbolique sur le fichier "exemple.c", de nom "toto.c"

# Droits d'accès

---

- Toute opération sur une entrée est contrôlée à partir des droits  $r$ ,  $w$  et  $x$ .
  - La création, le renommage, le déplacement et la suppression, d'une entrée dans un répertoire sont contrôlés par le droit  $w$  des répertoires.
  - Pour la création, il faut avoir le droit de modifier le contenu du répertoire pour ajouter une entrée.
  - Pour le renommage et le déplacement, il faut avoir le droit de modifier le contenu du répertoire pour modifier le nom d'une entrée ou/et ajouter une entrée.
- Pour un répertoire, le droit  $x$  autorise la traversée et le positionnement dans celui-ci.
  - Pour garder confidentiel le contenu d'un répertoire, un utilisateur enlève le droit  $r$  pour les autres ou le groupe ou lui-même.
  - Pour empêcher de se déplacer dans une partie de son arborescence, un utilisateur enlève le droit  $x$  au répertoire à la racine de cette arborescence.

- ❑ À tout élément de l'arborescence sont associés trois droits d'accès :
  - En lecture : r pour read
  - En écriture : w pour write
  - En exécution : x pour execute
- ❑ Pour trois catégories d'utilisateurs
  - Le propriétaire : u pour user
  - Le groupe : g pour group
  - Les autres : o pour others
  - Par exemple : `rwX r-x r--` s'interprète comme ceci :
    - ❑ Lecture, écriture et exécution autorisées pour le propriétaire
    - ❑ Lecture et exécution autorisées pour le groupe
    - ❑ Lecture seule autorisée pour les autres
- ❑ Les droits sont modifiables avec la commande *chmod* (*change mode*) selon 2 méthodes :
  - ❑ Valeur numérique calculée sur le poids de r, w et x pour chaque catégorie : r = 4, w = 2 et x = 1
  - ❑ Par exemple :
    - `chmod 741 fichier` ==> `rwX r-- --x`
    - `chmod 600 fichier` ==> `rw- --- ---`
  - Valeur littérale :
    - ❑ Catégories : u, g, o ou tous (a = all = ugo)
    - ❑ Opérations : ajout (+), retrait (-) ou affectation (=)
    - ❑ Par exemple : (fichier avec les droits `rw- --- ---`)
      - `chmod g+r fichier` ==> `rw- r-- ---`
      - `chmod ug+x fichier` ==> `rwX r-x ---`
      - `chmod a=r fichier` ==> `r-- r-- r--`
      - `chmod u+wx fichier` ==> `rwX r-- r--`
      - `chmod go-r fichier` ==> `rwX --- ---`

# Droits d'accès

---

- ❑ Limites des droits « traditionnels » UNIX
- ❑ Les trois niveaux de permission, **Read**, **Write** et **eXecute**, sont codés chacun sur un bit et sont appliqués sur les trois uniques classes d'utilisateurs : **User**, **Group**, et **Other**.
- ❑ User représente le propriétaire du fichier, Group les utilisateurs du même groupe que User, c'est-à-dire partageant les mêmes GID (group id, défini dans /etc/group), et Other, tous les autres utilisateurs, c'est-à-dire appartenant à un groupe différent.

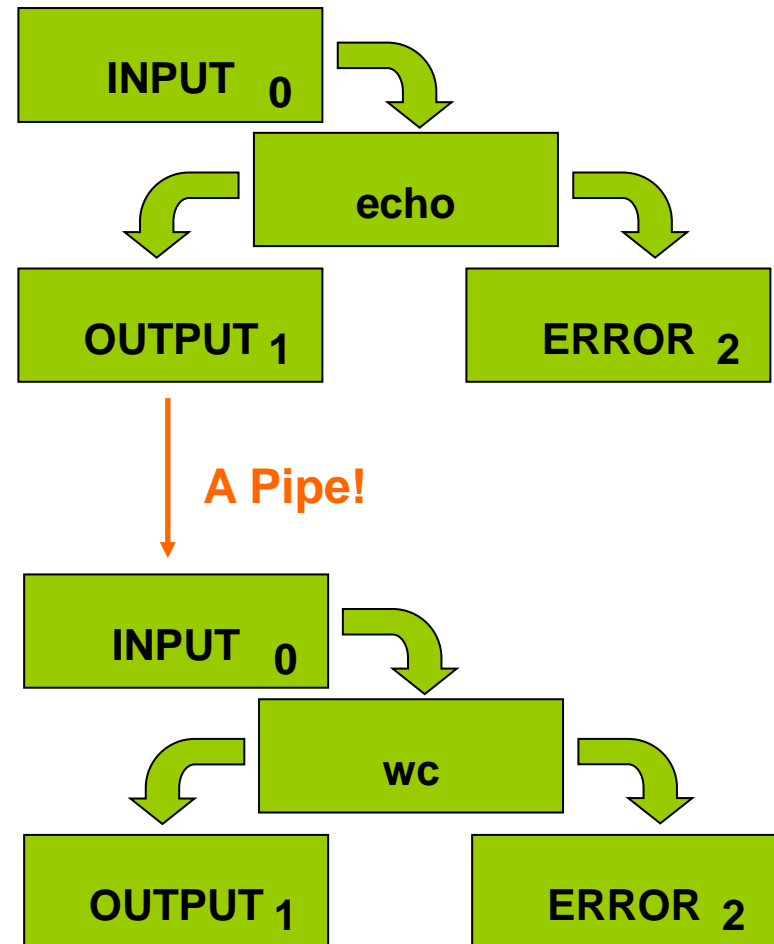
# Tube/Pipe (1)

---

- ❑ Un tube (pipe) est une zone mémoire permettant à deux processus d'une même machine de communiquer tout en étant synchronisés.
- ❑ L'objectif d'utiliser un tube est de faire agir une commande sur le résultat d'une autre sans fichiers intermédiaires.
- ❑ Le symbole « | » placé entre deux commandes redirige la sortie standard de la première sur l'entrée standard de la seconde.

# Tube/Pipe (2)

```
$ who | wc -l  
3
```





# Caractères spéciaux

- ❑ Les caractères dits « Jokers » servent à référencer des noms de fichiers :
  - \* ==> une chaîne de caractères quelconque (même vide) ;
  - ? ==> UN caractère quelconque ;
  - [...] UN caractère quelconque dans l'ensemble défini entre « [ » et « ] » ;
  - [!...] UN caractère quelconque hors de l'ensemble.
  - Un ensemble est défini par une liste de caractères ([adefgw]) ou un intervalle ([0-9]) ou toute combinaison des deux ([ab0-9A\;]).
- ❑ Par exemple,
  - \*.c ==> noms suffixés .c
  - \*. [csp] ==> noms suffixés .c, .s ou .p
  - [!0-9]\* ==> noms ne commençant pas par un chiffre
- ❑ Le caractère tilde (« ~ ») permet de repérer un nom de connexion ~ seul pour le répertoire de connexion de l'utilisateur
  - ~nom pour le répertoire de connexion de l'utilisateur nom
    - ❑ ls ~roose/WWW liste le sous-répertoire WWW de roose.

# Exemple d'utilisation des jokers

---

- ❑ `ls *.c`
- ❑ `ls fichier?.c`
- ❑ `ls java.*`
- ❑ `cd ~roose`
- ❑ `cd ~`

# Variables globales/de session

---

- ❑ Il existe sous Shell quelques variables pré-définies :
  - HOME : répertoire de connexion (cd et cd ~ sont égales à cd \$HOME)
  - PATH : liste des répertoires de recherche des commandes séparés par « : »
- ❑ L'ordre des répertoires est significatif, ce qui permet le choix de la version d'une commande.
- ❑ Pour afficher le contenu de ces variables
  - echo \$NOMVAR

# Variables globales/de session

---

- ❑ PWD : Répertoire courant
- ❑ USER : nom de l'utilisateur courant
- ❑ PATH : chemins où aller chercher les commandes
- ❑ DISPLAY : L'écran sur lequel les programmes X travaillent. Cette variable est souvent de la forme : machine.somehost.somewhere:0.0 Si cette variable est vide, c'est qu'il n'y a pas d'affichage graphique possible.
- ❑ SHELL : contient le nom de votre shell.
- ❑ HOME : contient le nom de votre répertoire personnel.

# Commandes liées aux processus

---

- ❑ La commande *ps* affiche les caractéristiques des processus à un instant donné.
- ❑ Par défaut, *ps* affiche les processus de l'utilisateur en cours dans le terminal en cours. L'option *-u nom utilisateur* permet de connaître l'ensemble des processus d'un utilisateur
- ❑ La commande *pstree* permet de visualiser l'arborescence des processus.
- ❑ La commande *top* permet de visualiser dynamiquement les caractéristiques des processus (l'affichage est actualisé périodiquement).
- ❑ En plus des informations sur les processus, *top* donne des indicateurs sur l'état du système : occupation de la mémoire, de l'unité centrale... *top* montre l'évolution de ces indicateurs en « temps réel ».
- ❑ Si l'on désire supprimer un processus en cours d'exécution (par exemple s'il ne répond plus), il est nécessaire de connaître sonr PID (avec la commande *PS* par exemple). Ensuite, la commande *kill* forcera sa terminaison en envoyant le signal 9 de destruction (SIGKILL).
  - *kill -9 numéro(s) de PID du ou des processus*

# Tâche de fond

---

- Une commande peut aussi s'exécuter en arrière-plan (en anglais background). Par exemple, l'utilisateur saisit *date &*.
  - Le shell crée un processus enfant et n'attend pas qu'il se termine.
  - Le processus enfant exécute la commande *date*.
  - Les deux processus, parent et enfant, s'exécutent alors « simultanément ».



# Autres commandes

---

- ❑ La commande *grep* affiche les lignes d'un fichier de texte contenant une expression.
- ❑ Cette commande est très utilisée pour rechercher un mot donné dans un fichier.

```
$ grep elaphe /etc/hosts
```

- ❑ Pour afficher les lignes ne contenant pas *elaphe*, il faut utiliser l'option *-v* :

```
$ grep -v elaphe /etc/hosts
```

# Autres commandes

---

- ❑ *La commande sort trie les lignes d'un ou plusieurs fichiers de texte. Par défaut, le tri se fait suivant l'ordre lexicographique. Par exemple, la commande suivante trie le fichier /etc/passwd*

```
$ sort /etc/passwd
```

- ❑ *La commande paste fusionne des fichiers. Par exemple, la commande suivante fusionne les fichiers fic1 et fic2 en un fichier fic3 :*

```
$ paste fic1 fic2 > fic3
```



# Autres commandes

---

- ❑ La commande *find* recherche les fichiers satisfaisant un ensemble de critères de sélection à partir d'un répertoire.
- ❑ Le principe de *find* est de parcourir récursivement l'arborescence du système de fichiers, et pour chaque élément, de tester chaque critère :
- ❑ La syntaxe de *find* est la suivante :

```
$ find / -name core -print  
$ find /usr -name '*.c' -print
```

⇒ affichage des fichiers de nom core  
⇒ affichage des fichiers sous /usr  
dont le nom se termine par .c<sup>105</sup>

# Autres commandes

---

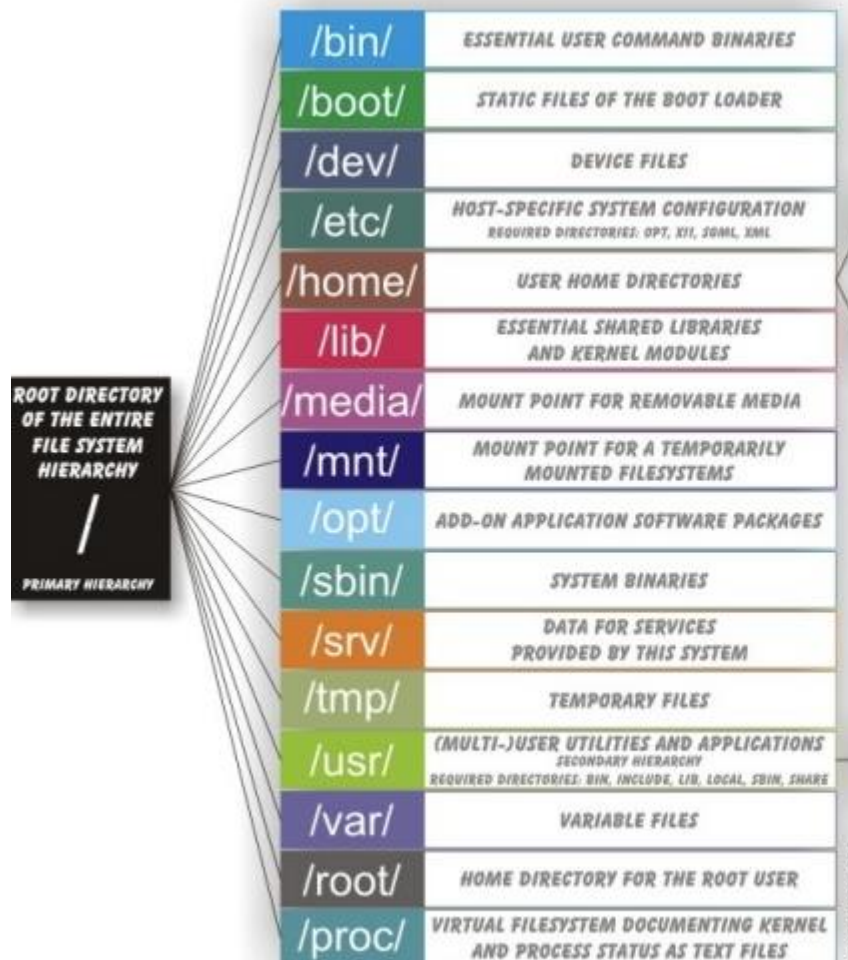
## ❑ tar

- Archivage
- Exemple : `tar cvf tpc.tar *.c`
  - ❑ crée une archive appelée *tpc.tar* contenant tous les fichiers d'extention `.c` du répertoire courant.
- Exemple : `tar cvf tpc.tar *.c`
  - ❑ Décompression

## ❑ gzip/gunzip

- Compression de fichier
  - ❑ `gzip tpc.tar ->`
    - `tpc.tar.gz`
- Décompression
  - ❑ `gzip -d tpc.tar.gz`

# Arborescence



- Le tronc de l'arbre s'appelle "root" (la racine). De là part toute l'arborescence.
- Il est noté "/".  
Quelque part dans cet arbre se trouve un répertoire particulier nommé "répertoire principal" ou "home directory" qui correspond au point d'entrée de l'utilisateur au moment de sa connexion sur la machine.
- Il existe deux manières de définir un chemin d'accès à un point de l'arborescence:
  - le chemin absolu
    - indique l'énumération des répertoires à suivre depuis la racine jusqu'au lieu désiré. Ce chemin commence toujours par "/".  
Ex: /home/bernel
  - le chemin relatif
    - idem mais en partant du répertoire courant.  
Ex: si je me trouve dans le répertoire /home/bernel, tmp/cours est le chemin relatif pour atteindre /home/bernel/tmp/cours, qui n'a a priori rien à voir avec /tmp/cours !

# Quelques répertoires importants

---

- `.` : le répertoire courant. Son chemin absolu peut être connu avec la commande `pwd`
- `..` : le répertoire parent
- `/` : la racine de toute l'arborescence.

# Types de fichiers

- A la base, toute l'arborescence Unix est constituée de fichiers qui sont de différents types.
- Ces derniers peuvent être affichés via la commande

- "ls -al":

```
$ ls -al /tmp/bernel/Cours
```

```
total 24
drwx----- 2 berne1 root 4096 Dec 12 18:03 .
drwx----- 3 berne1 root 4096 Dec 12 18:02 ..
-rwxr-x--x 1 berne1 root 0 Dec 12 17:18 file1
mrwx----- 1 berne1 root 2807432 Dec 12 18:05 file2
-rw-r----- 1 berne1 root 0 Dec 12 17:18 file3
lrwxrwxrwx 1 berne1 root 5 Dec 12 18:10 file4 -> file1
```

- \$ Un fichier peut donc être: un répertoire (Directory) - un fichier "normal" | un Lien