Architecture des Ordinateurs et Systèmes d'Exploitation

Partie 2 : Système d'Exploitation
Cours

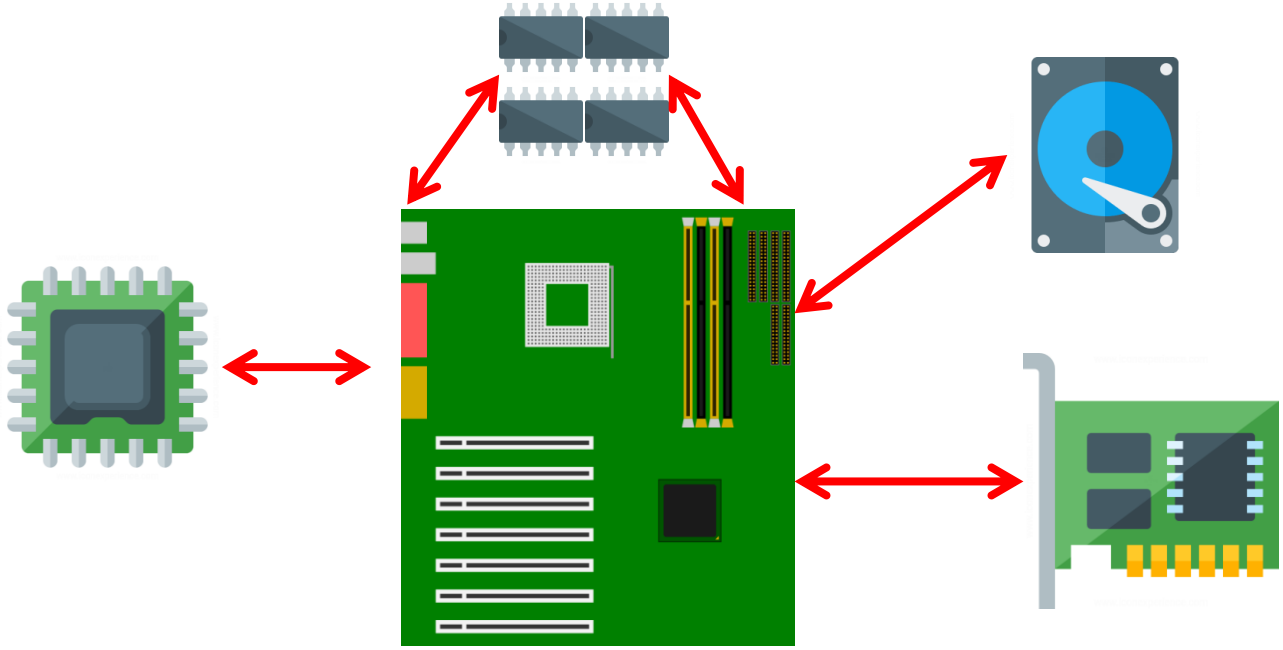
Fabrice BOISSIER & Elena KUSHNAREVA
2017/2018
fabrice.boissier@gmail.com
elena.kushnareva@malix.univ-paris1.fr

Les Systèmes d'Exploitation

Ordinateur

=

Processeur + Mémoire + Périphériques



Les Systèmes d’Exploitation

Comment développer des applications ?

…en lisant la doc des composants…

…donc créer des applications dépendantes du matériel…

Les Systèmes d’Exploitation

Comment développer pour toutes les plateformes ?

…faire une application par plateforme…

…ou…

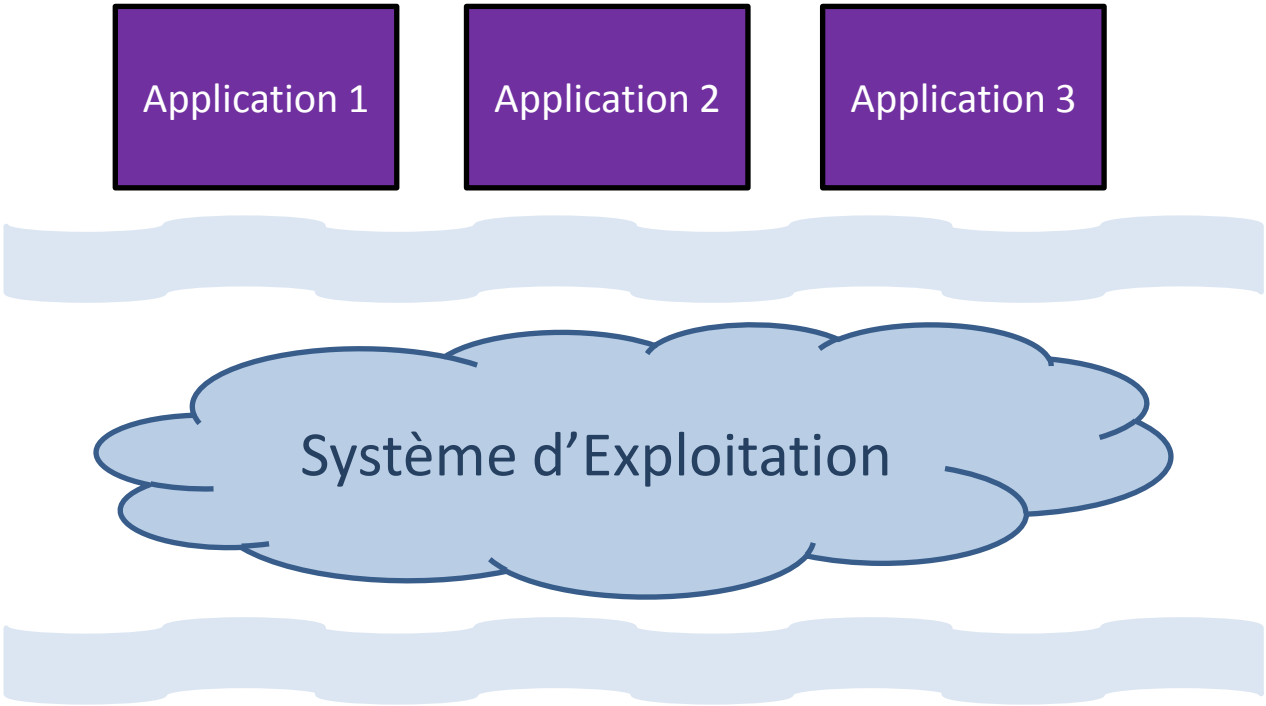
Les Systèmes d'Exploitation

Comment développer pour toutes les plateformes ?

…faire une couche d'abstraction
entre le matériel et les logiciels…

…et offrir des services indépendants
des spécificités du matériel…

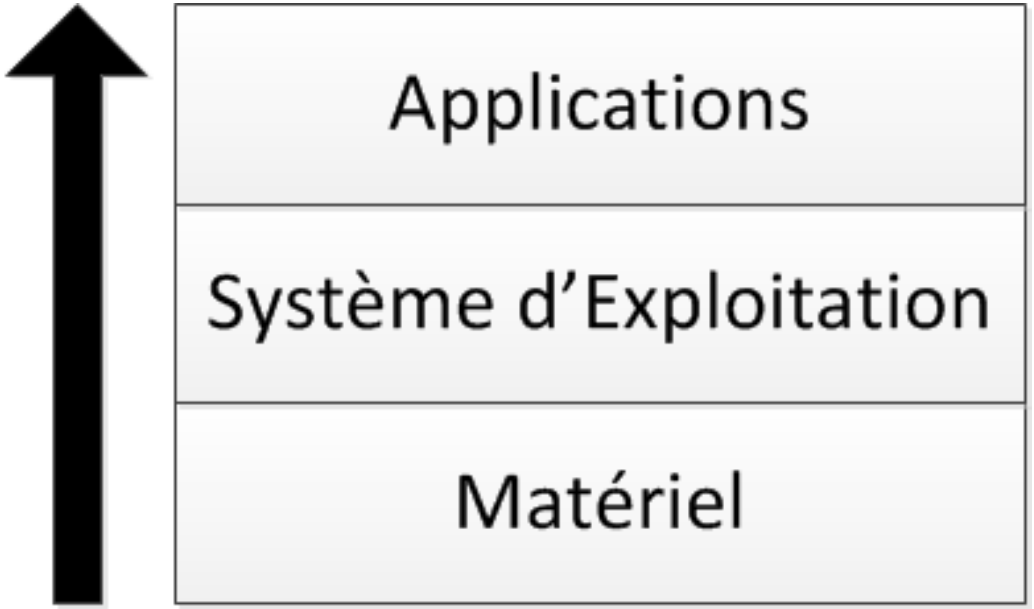
Les Systèmes d'Exploitation



Les Systèmes d’Exploitation

Applications Portables

Applications Spécifiques
(au matériel)



Les Systèmes d’Exploitation

Comment faire cohabiter plusieurs programmes ?

…on ne le permet pas…

…ou…

Les Systèmes d’Exploitation

Comment faire cohabiter plusieurs programmes ?

…chaque programme doit « donner » du temps aux autres programmes…

…ou…

Les Systèmes d’Exploitation

Comment faire cohabiter plusieurs programmes ?

…le système gère le temps accordé aux programmes…

Les Systèmes d’Exploitation

Comment faire…

Il y a beaucoup BEAUCOUP de questions qui se sont posées au fur et à mesure des usages de l’informatique

Nous essayerons de répondre à plusieurs

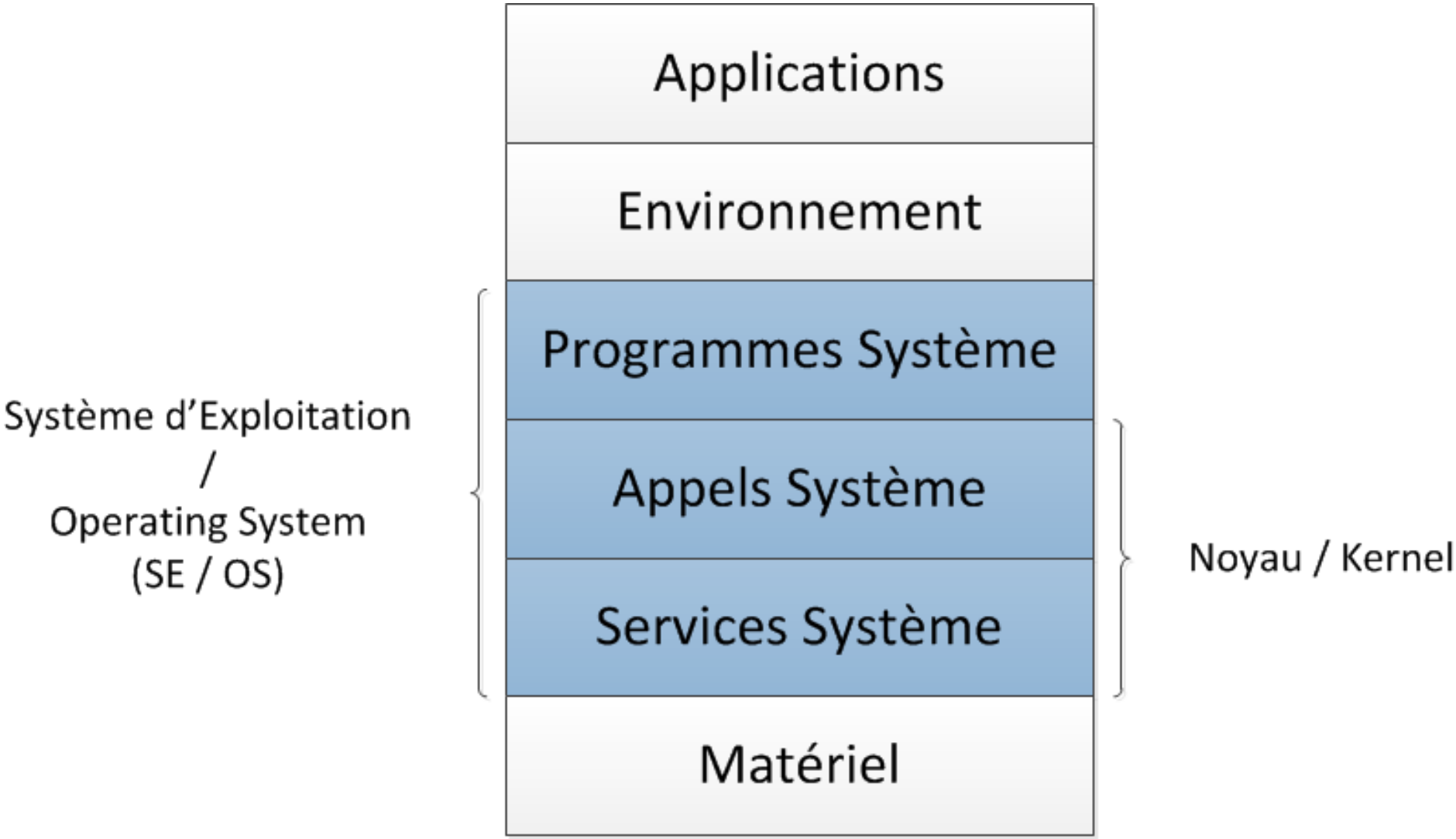
Les Systèmes d’Exploitation

En résumé :

Le Système d’Exploitation gère tout.

C’est un « système » qui « exploite » la machine/plateforme pour rendre des services aux utilisateurs.

Les Systèmes d'Exploitation



Les Systèmes d'Exploitation

- Fonctionnellement :

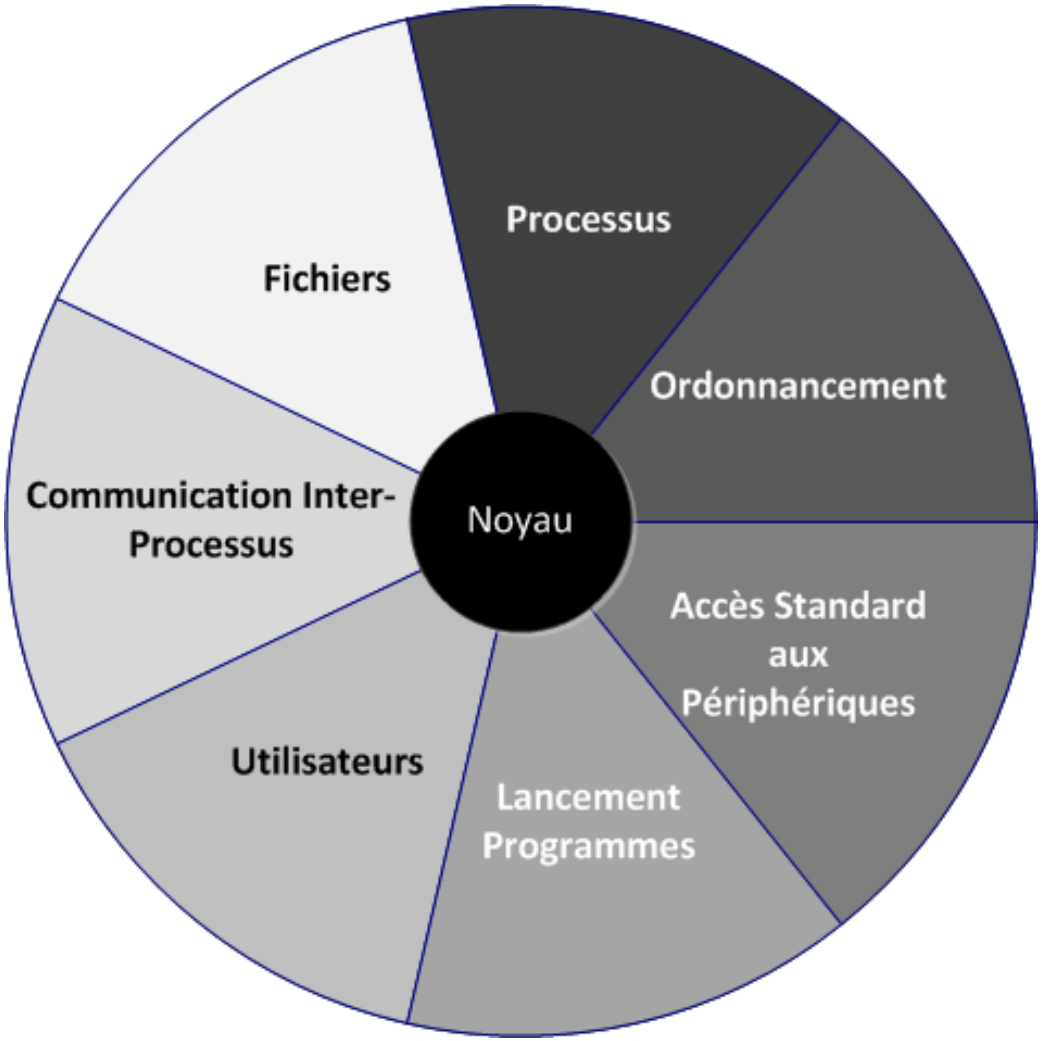
  Fourni des services utiles aux utilisateurs

- Techniquement :

  Utilise le plus efficacement la plateforme

En offrant une surcouche indépendante du matériel

Les Systèmes d'Exploitation



Quelques Systèmes d'Exploitation

Plateforme	Systèmes d'Exploitation disponibles
Mainframe	z/OS, GCOS, Linux, …
Mini	IBM i, Windows Server, UNIX, BSD, Linux, …
Micro	Windows, macOS, UNIX, BSD, Linux, …
Smartphone	iOS, Android, …

Références Bibliographiques

- Modern Operating Systems.
  Andrew S. Tanenbaum Ed. Prentice-Hall, 2001

- Systèmes d'exploitation : systèmes centralisés & systèmes distribués.
  A. Tanenbaum. Ed. Prentice Hall, Dunod, (Traduction).

- Operating System Concepts.
  A. Silberschatz, P.B. Galvin & G. Gagne, Ed. John-Wiley.

- Principes appliqués des systèmes d'exploitation avec Java.
  A. Silberschatz, P. B. Galvin & G. Gagne. . Ed. Vuibert,. (Traduction)

- Operating Systems, Internals and Design Principles.
  William Stallings. Ed. Prentice-Hall.

- Operating systems, A Modern Perspective.
  Gary Nutt. Ed. Addison-Wesley.

Références Bibliographiques

- Linux : Programmation système et réseau.
  Joëlle DELACROIX (CNAM)
- The Open Group
  Single UNIX Specification / SUSv4
- Microsoft
  Windows Internals
- IBM (publib.boulder)
  Introduction to the New Mainframe - z/OS Basics
- FreeBSD Handbook
- Linux Man Pages

Système d’Exploitation

- Le rôle d'un système d'exploitation est d'assurer la transparence d'accès aux différents composants d'un ordinateur

Le système d’exploitation :

- Est un gestionnaire des ressources : il gère et contrôle les composants de l’ordinateur
- Est une machine virtuelle : il fournit une base (machine virtuelle) sur laquelle seront construits les programmes d’application et les utilitaires

But :

- Développer des applications sans se soucier des détails de fonctionnement et de gestion du matériel

Système d’Exploitation

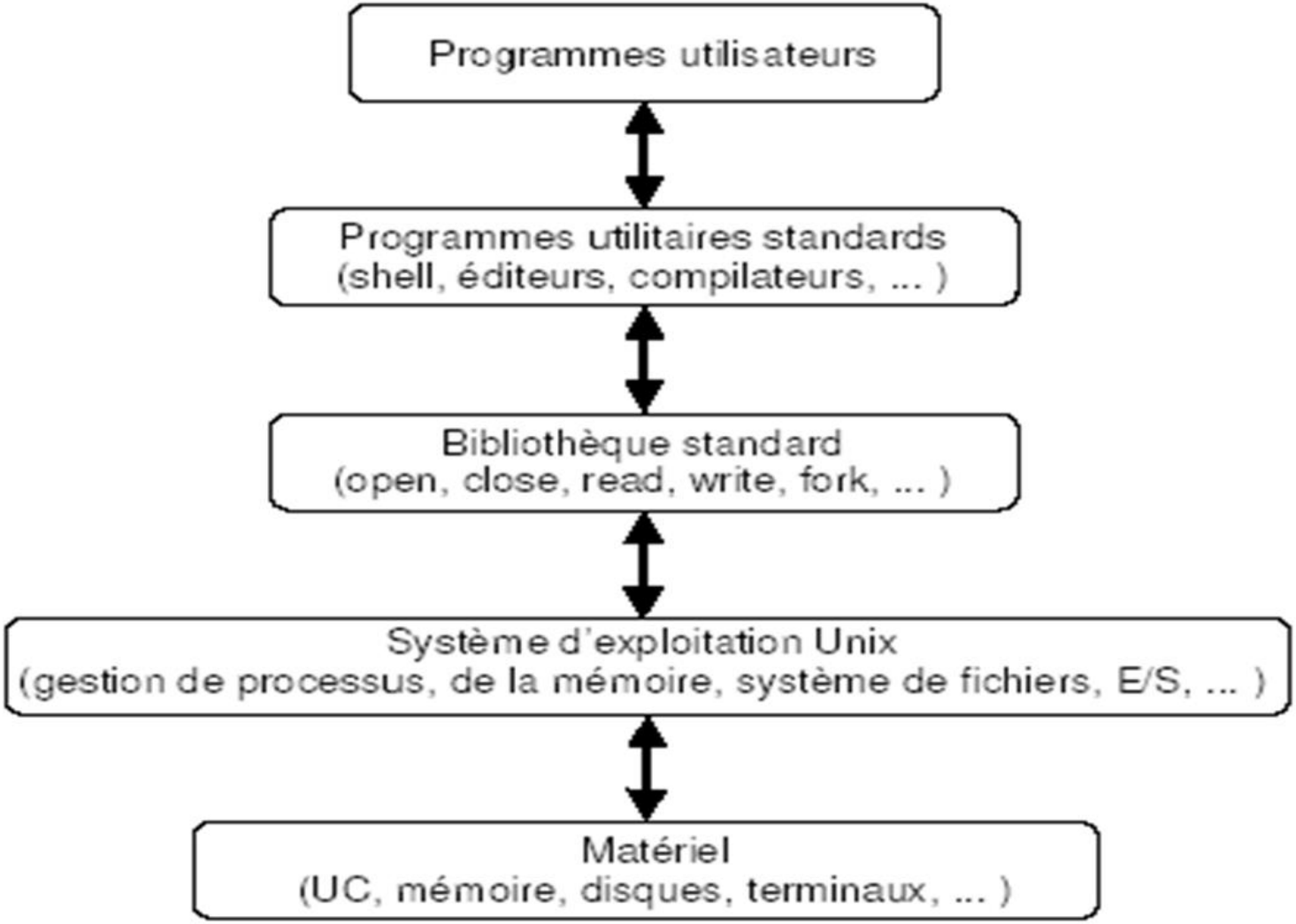
- Ses fonctions principales sont :
  - gérer les différentes ressources matérielles constituant une machine,
  - contrôler l'environnement d'exécution des programmes,
  - offrir à ses utilisateurs un ensemble de services

    (commandes systèmes, langage d'enchaînement de commandes, interfaces graphiques pour la gestion des ressources)
  - proposer aux programmeurs un ense.mble d'API d'accès à ces services pour le développement d'applications

Système d’Exploitation

- Fonctions principales :
  - Gestion des périphériques (des E/S ou I/O)
  - Gestion de la mémoire
  - Gestion des processus et des threads
  - Gestion des fichiers
  - Protection et détection d’erreurs
  - Gestion des communications entre machines (réseau)

Système d'Exploitation



Pilotes / Drivers

- Chaque composant d'un ordinateur possède du code pour communiquer avec les autres composants
  (dans son firmware par exemple)

- Les pilotes (drivers) permettent au système d'exploitation de manipuler de façon optimale les périphériques
  (l'OS ne peut pas deviner sans driver les protocoles/fonctions que le périphérique gère, ni les paramètres qu'ils emploient)

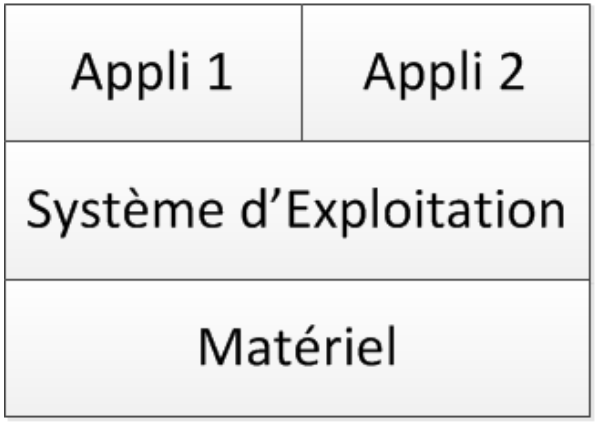
  – Le pilote est du code dans l'OS (logiciel/soft)
  – Le firmware est du code dans le matériel (matériel/hard)

Hyperviseurs

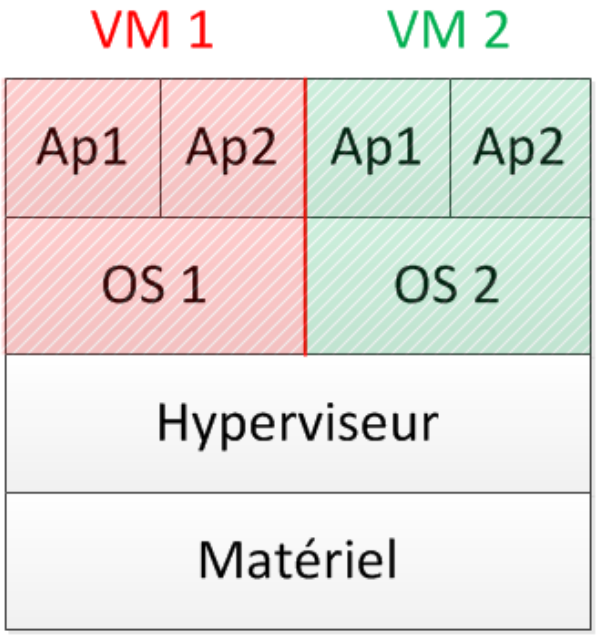
- Conteneur de Systèmes d'Exploitation

- 2 types d'hyperviseurs :
  - Hyperviseur Type 1 : Programme fonctionnant juste au dessus de la machine (plutôt rapide)
  - Hyperviseur Type 2 : Programme fonctionnant dans un système d'exploitation déjà existant (plutôt lent, presque assimilable aux émulateurs)

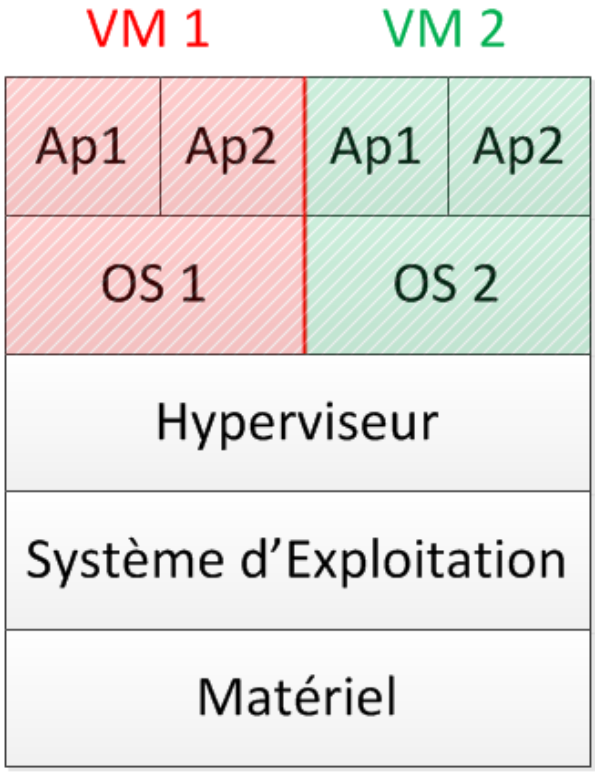
Hyperviseurs



Ordinateur sans Hyperviseur



Hyperviseur Type 1



Hyperviseur Type 2

Système d’Exploitation : 3 concepts

- **Processus** :
  un programme en cours d’exécution, composé de :
  - code + données + pile d’exécution ;
  - un compteur ordinal, autres informations caractérisant son état.

- **Fichiers** :
  ensemble de blocs de données stockés sur le disque.

- **Mémoires virtuelles** :
  espaces d’adressage virtuels des processus (créés par les compilateurs) de taille pouvant excéder celle de la mémoire physique.

Interpréteur de Commandes

- L’interpréteur de commandes (interface utilisateur/système) :
  - est lancé dès la connexion au système ;
  - invite l’utilisateur à introduire une commande ;
  - récupère puis exécute la commande par combinaison d’appels système et d’outils ;
  - affiche les résultats ou les erreurs puis se met en attente de la commande suivante.

Interpréteur de Commandes

Invite de commandes

```
Microsoft Windows [version 6.1.7601]
Copyright (c) 2009 Microsoft Corporation. Tous droits réservés.

C:\Users\Metalman>dir /w
 Le volume dans le lecteur C n'a pas de nom.
 Le numéro de série du volume est B83E-3B45

 Répertoire de C:\Users\Metalman

[.]                 [..]
[.gimp-2.8]         [.jmc]
[.thumbnails]       [.VirtualBox]
[Desktop]           [Documents]
[dwhelper]          [dynamorio]
[Links]             [Music]
[Pictures]          [Roaming]
[Tracing]           [Videos]
[YAKINDU_SCT]
               0 fichier(s)
              33 Rép(s)   3 833 7

C:\Users\Metalman>
```

~

```
Metalman@X220 ~
$ ls
a.exe                   my_script.sh
DL                      my_script.sh~
Ex2.sct                 MyEnv
Ex2_HG_no_labels.png    networkx-1.11-py2.py3-none-any.whl
fdk-aac-0.1.5.tar.gz    ProjExemples
ffmpeg-3.2.2            recherche_mot_fichiers.sh
ffmpeg-3.2.2.tar.bz2    rename_numbers.sh
flickr                  rmn
flickr_downloader.sh    susv4tc1.tar.bz2
fmmidi-1.0              svn
folders_enum.sh         TEST
git                     test1.c
lame-3.99.5.tar.gz      test1.c~
Metalman                test2.c
my_python               VirtualEnv_Python.txt
my_R                    x264-snapshot-20160408-2245-stable.tar.bz2
my_rename.sh

Metalman@X220 ~
$
```

UNIX

Unix est un système d'exploitation

- interactif,
- multi-utilisateurs,
- multitâches, à temps partagé
- multi-langages,

Unix offre

un langage de commande

- séquentiel,
- pseudo-parallèle,
- abréviations,
- re-directions d'entrée-sorties,
- commandes de base,
- programmes,
- communications,
- synchronisation...

une documentation en ligne

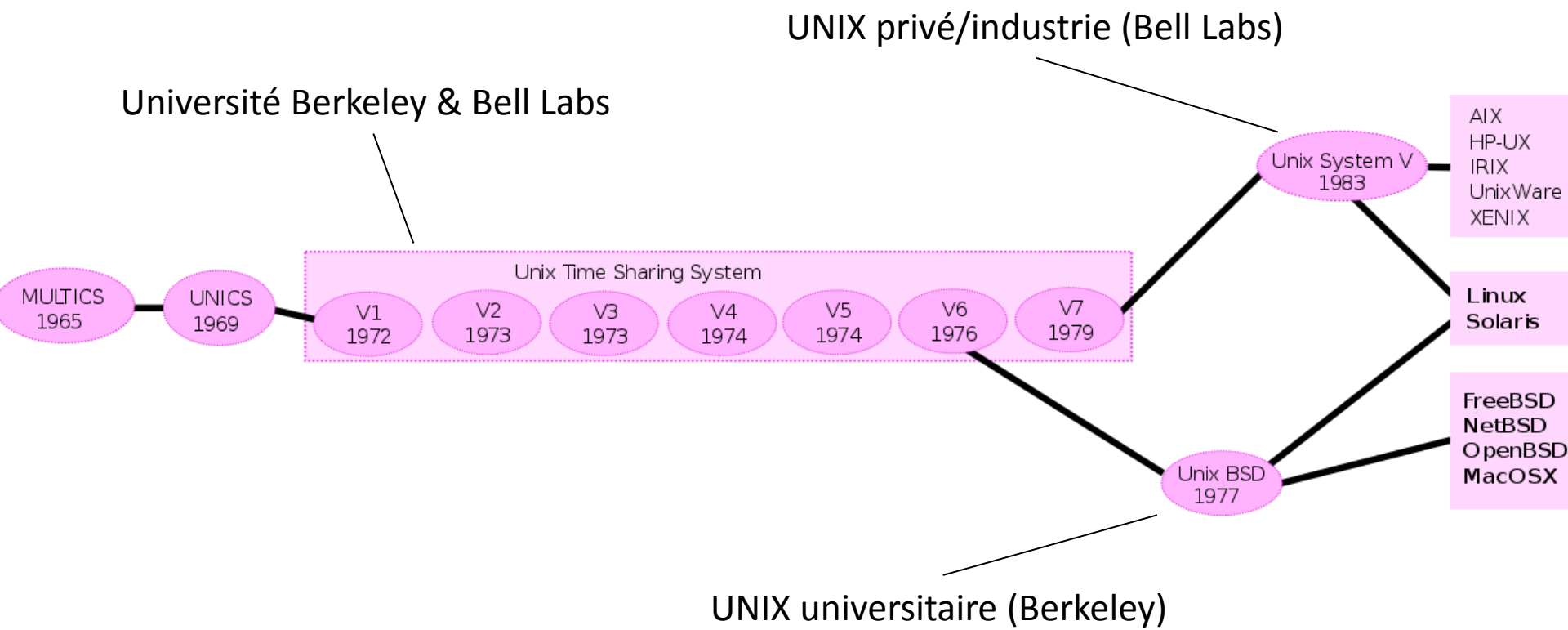
man

des utilitaires

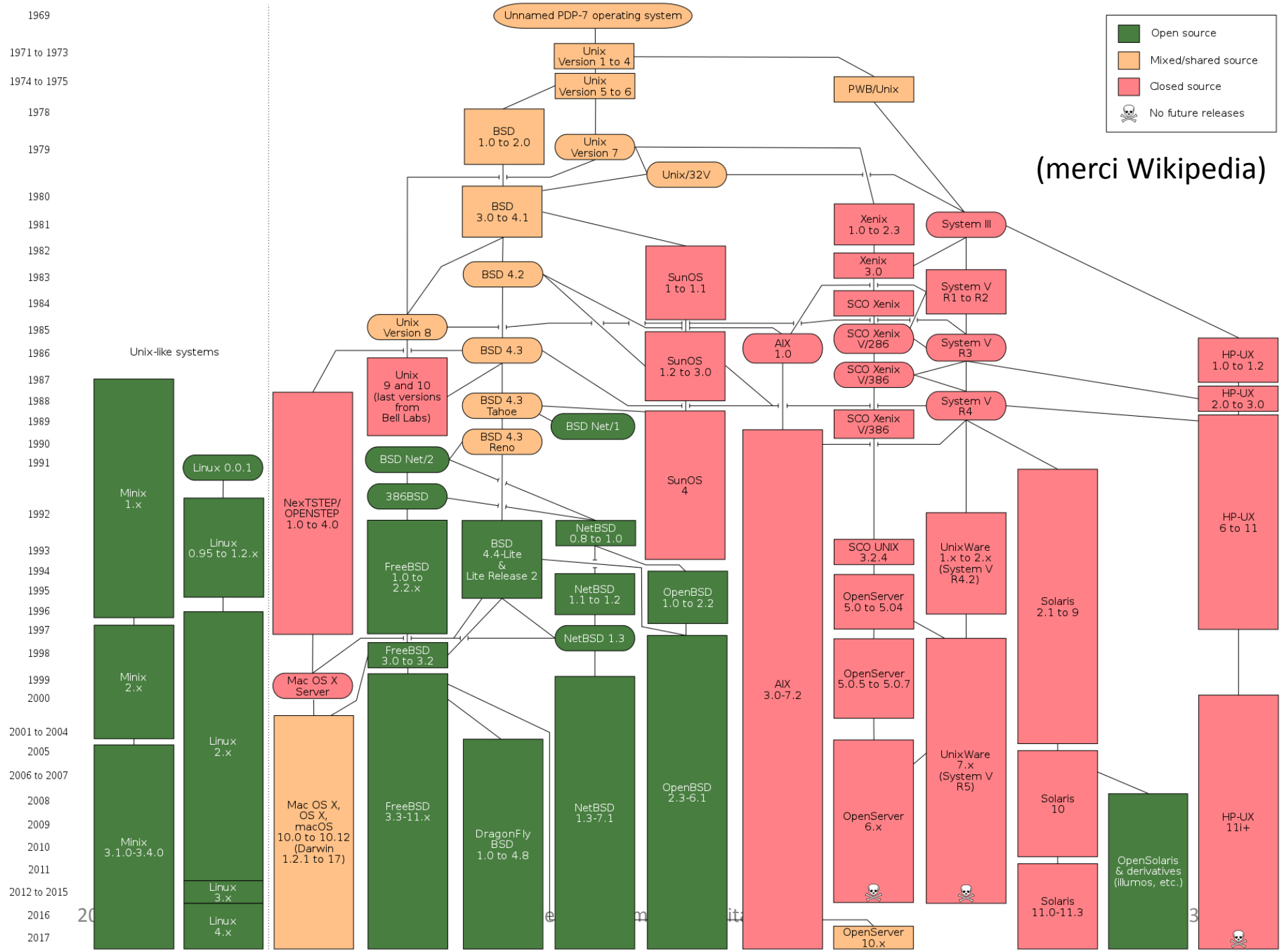
- traitement de texte (*troff, nroff, latex*)
- gestion d'applications (*make*)

- Installation cygwin : http://cygwin.com/

UNIX : Historique



(merci Wikipedia)


1969
1971 to 1973
1974 to 1975
1978
1979
1980
1981
1982
1983
1984
1985
1986
1987
1988
1989
1990
1991
1992
1993
1994
1995
1996
1997
1998
1999
2000
2001 to 2004
2005
2006 to 2007
2008
2009
2010
2011
2012 to 2015
2016
2017
Unix-like systems
Open source
Mixed/shared source
Closed source
No future releases
(merci Wikipedia)
Unnamed PDP-7 operating system
Unix Version 1 to 4
Unix Version 5 to 6
PWB/Unix
BSD 1.0 to 2.0
Unix Version 7
Unix/32V
BSD 3.0 to 4.1
Xenix 1.0 to 2.3
System III
Xenix 3.0
BSD 4.2
SunOS 1 to 1.1
System V R1 to R2
SCO Xenix
Unix Version 8
SCO Xenix V/286
AIX 1.0
System V R3
HP-UX 1.0 to 1.2
BSD 4.3
SunOS 1.2 to 3.0
SCO Xenix V/386
Unix 9 and 10 (last versions from Bell Labs)
BSD 4.3 Tahoe
BSD Net/1
System V R4
HP-UX 2.0 to 3.0
SCO Xenix V/386
BSD 4.3 Reno
BSD Net/2
Linux 0.0.1
Minix 1.x
NexTSTEP/ OPENSTEP 1.0 to 4.0
386BSD
SunOS 4
HP-UX 6 to 11
Linux 0.95 to 1.2.x
FreeBSD 1.0 to 2.2.x
BSD 4.4-Lite & Lite Release 2
NetBSD 0.8 to 1.0
SCO UNIX 3.2.4
UnixWare 1.x to 2.x (System V R4.2)
Solaris 2.1 to 9
NetBSD 1.1 to 1.2
OpenBSD 1.0 to 2.2
OpenServer 5.0 to 5.04
Minix 2.x
NetBSD 1.3
FreeBSD 3.0 to 3.2
OpenServer 5.0.5 to 5.0.7
Mac OS X Server
Linux 2.x
AIX 3.0-7.2
UnixWare 7.x (System V R5)
Minix 3.1.0-3.4.0
Mac OS X, OS X, macOS 10.0 to 10.12 (Darwin 1.2.1 to 17)
FreeBSD 3.3-11.x
DragonFly BSD 1.0 to 4.8
NetBSD 1.3-7.1
OpenBSD 2.3-6.1
OpenServer 6.x
Solaris 10
OpenSolaris & derivatives (illumos, etc.)
HP-UX 11i+
Linux 3.x
Linux 4.x
Solaris 11.0-11.3
OpenServer 10.x

UNIX & Linux

- UNIX : OS d’origine
  - A donné des standards POSIX et SUS
- Minix : copie « faite maison » d’UNIX
  - Créé pour des TP et projets
- Linux : kernel « copié » du projet Minix
  - Ne contient qu’un kernel… pas de logiciels

UNIX & Linux

- GNU : (Gnu is Not Unix)
  - Type de licence demandant à ce que les sources des logiciels soient fournies « avec » le logiciel
  - Ensemble des programmes UNIX recodés « avec » leurs sources accessibles

- GNU/Linux : kernel Linux + programmes GNU
  - Est un système d’exploitation complet

UNIX :
Système Multi-Utilisateurs et Multi-Tâches

Multi-utilisateurs

Plusieurs utilisateurs se partagent les ressources du système dans la mesure des droits dont ils disposent.
Pour ce faire, chacun devra être identifié au sein de ce dernier.

Il existe un utilisateur privilégié appelé SuperUtilisateur (SU pour SuperUser) ou Root qui peut accéder à l'ensemble des ressources présentes.
Son travail consiste à gérer la machine (cohérence et intégrité des données, performances et paramétrages du système, gestion des utilisateurs ...)

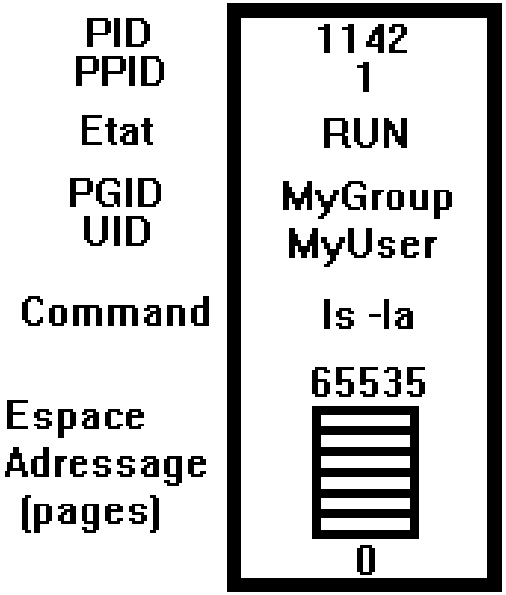
Multi-tâches

Les tâches lancées par plusieurs utilisateurs durant un même intervalle de temps seront exécutées selon un ordre et une gestion du temps définis par le système d'exploitation.
Ce système travaille en **temps partagé**.

Les Processus

Process Identifier	PID
Parent Process Identifier	PPID
Process Group Identifier	PGID
User Identifier	UID

- Un processus est une entité rassemblant toutes les informations nécessaires pour qu'un programme puisse s'exécuter sous le contrôle du système d'exploitation



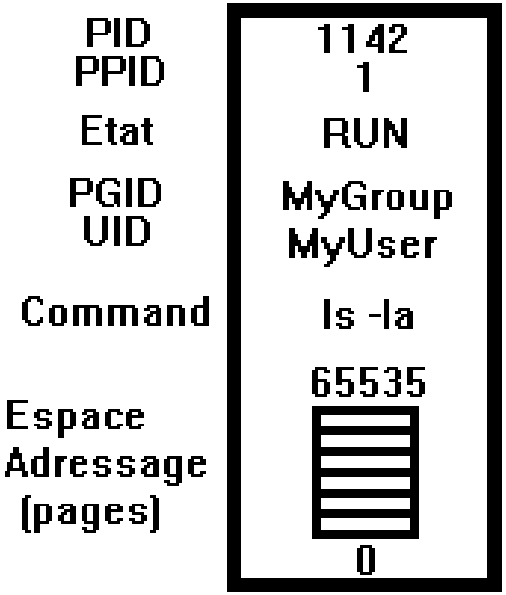
Les Processus

- Chaque processus :
  - est identifié par son **PID**
  - est rattaché à un processus père **PPID**
  - dispose de droits vis-à-vis des fichiers (**UID** et **GID**)
  - connait la commande qui l'a créé (**Command**)

- Le système d'exploitation lui attribut des pages mémoire contenant le code et les autres parties utiles

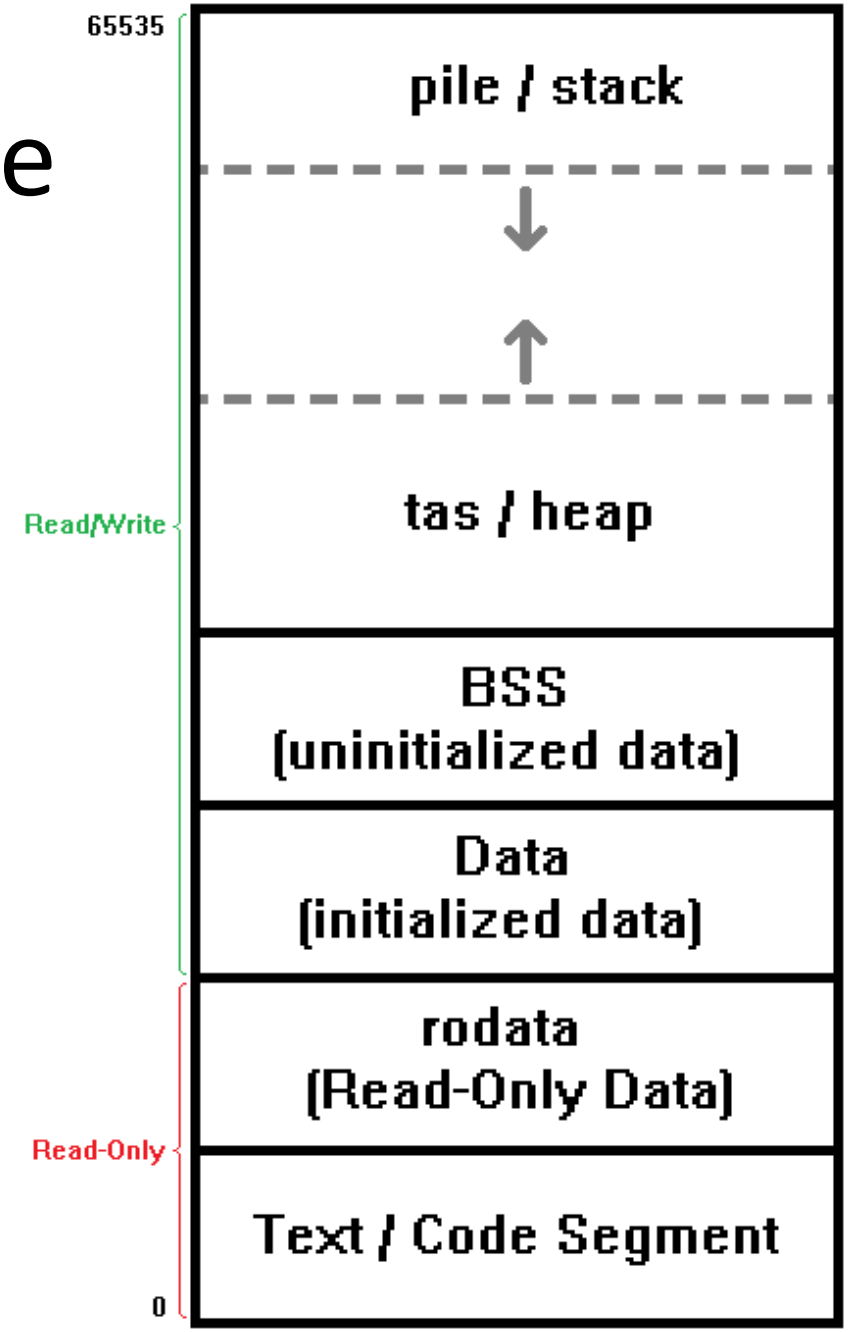
- Chaque processus est dans un état précis (en exécution, attente I/O, stop, …)

Process Identifier	PID
Parent Process Identifier	PPID
Process Group Identifier	PGID
User Identifier	UID



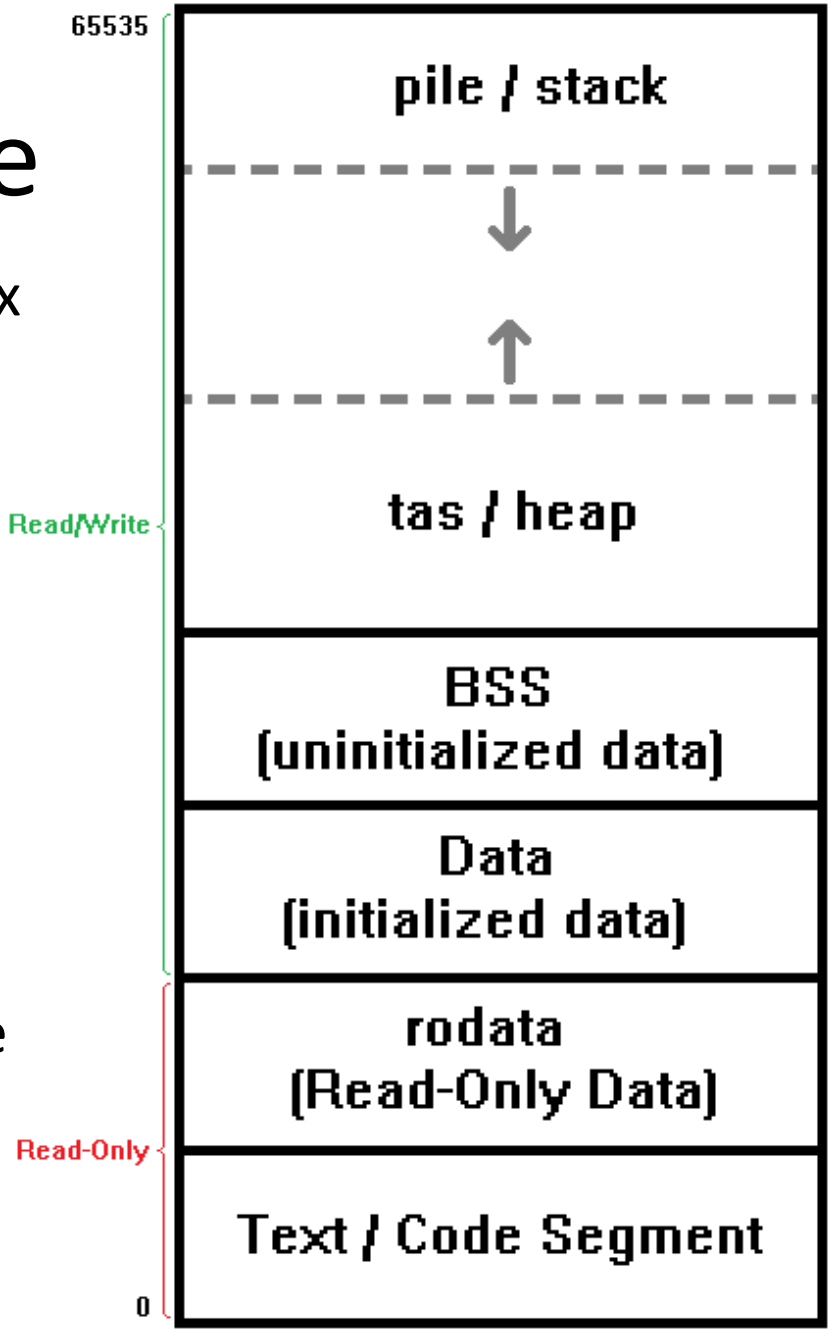
Espace d'Adressage

- Avant d'être mis en mémoire, un programme est découpé en plusieurs zones.
  Certaines zones sont en lecture seule, d'autres en mode lecture et écriture.



Espace d'Adressage

- **Pile/Stack** : les arguments passés aux fonctions
- **Tas/Heap** : les données allouées dynamiquement (malloc, new, …)
- **BSS** (*Block Started by Symbol*) : variables non initialisées et variables à 0
- **Data Segment** : données initialisées (copiées de RODATA vers DATA au démarrage du programme)
- **RODATA** (*Read-Only Data*) : les données initialisées et les chaînes de caractères pré-déclarées
- **Text/Code Segment** : contient le code en langage machine du programme



```
int  main(void)
{
     char *Var = "Test.";
     char *MyStr;
     int i = 0;
     int a = 1337;

     i = addition(21, 42);
     MyStr = malloc(32 * sizeof (char));
     return (0);
}
```

MyStr et i ne sont pas initialisés ou à 0

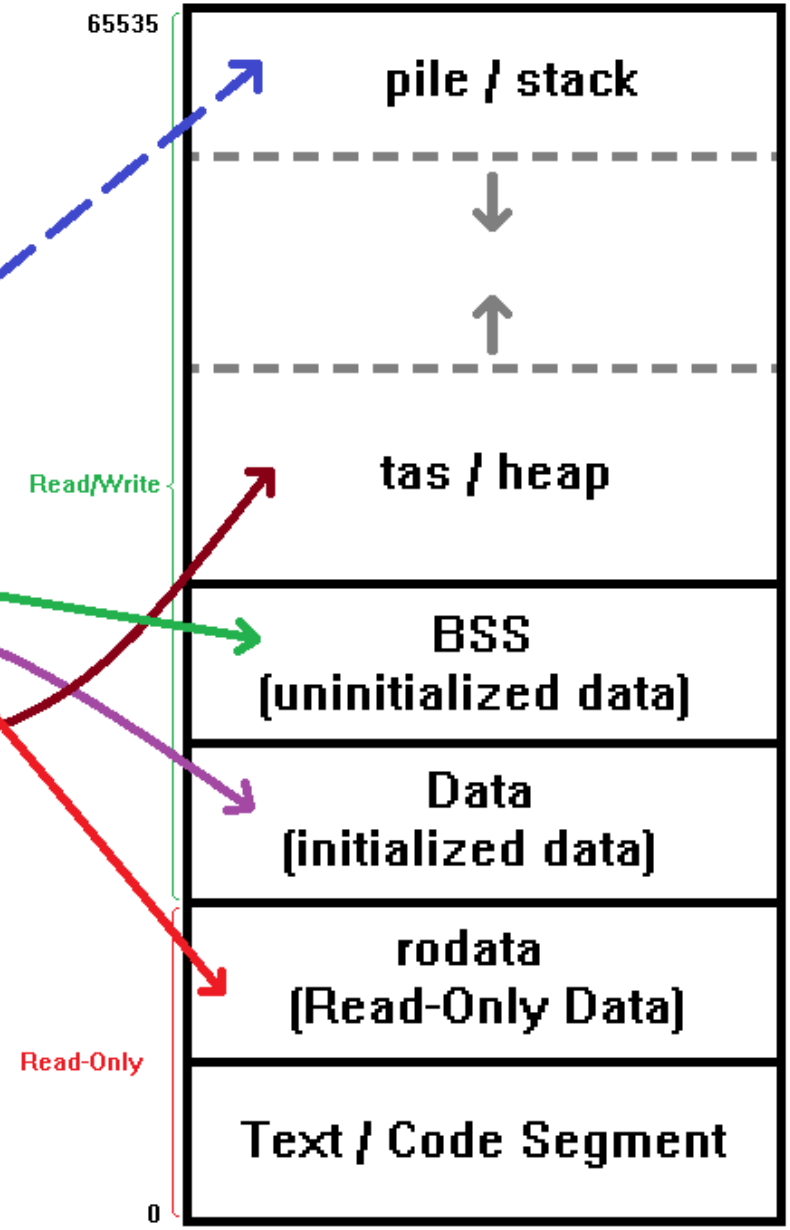
Var et a sont initialisés et modifiables

La chaîne "Test." n'est pas modifiable

Malloc va allouer 32 octets sur le tas

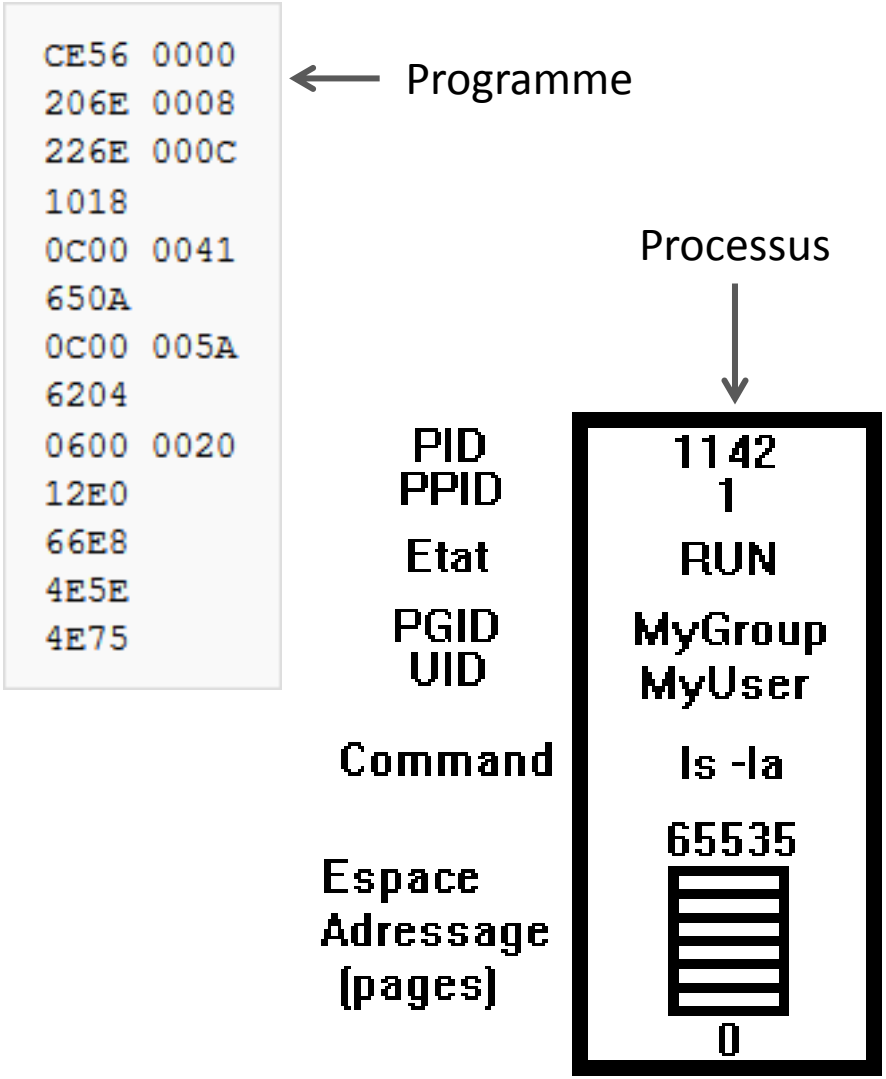
Les paramètres seront passés par la pile

L'ensemble du code est stocké dans Text



Processus vs Programme

- Un programme est un fichier contenant le code exécutable
- Un processus est le code chargé en mémoire et exécuté par le processeur



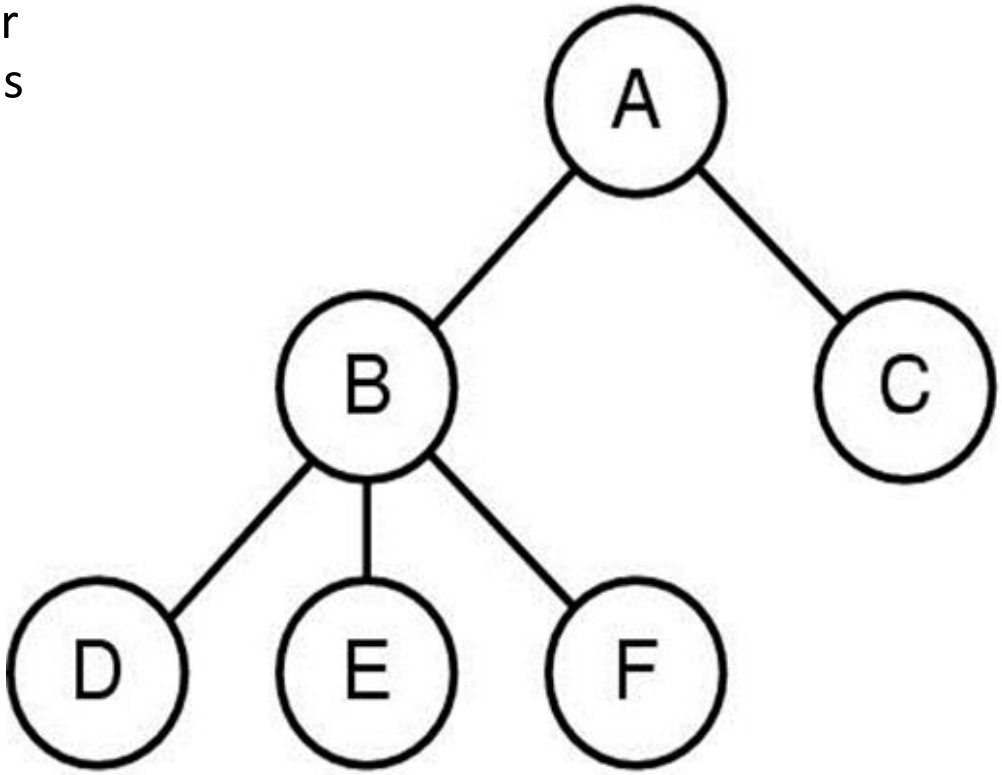
Processus

- Un processus est donc :
  - Un programme (le code en langage machine)
  - Déployé sur plusieurs pages mémoire selon un plan précis (l’espace d’adressage)
  - Avec des caractéristiques supplémentaires connues du système d’exploitation (PID, PPID, …)

Processus : Arborescence

- Un processus peut créer un ou plusieurs processus fils qui, à leur tour, peuvent créer des processus fils (structure arborescente).

- Un processus peut être partitionné en plusieurs threads (processus légers) concurrents partageant un même environnement d'exécution. Les threads sont un moyen de raffiner et de diviser le travail normalement associé à un processus.



Ordonnanceur / Scheduler

- Le système d'exploitation gère autant de processus qu'il y a de programmes en cours d'exécution.

- Comment partager le processeur (qui exécute les programmes) entre tous les processus ?

- Le système d'exploitation va donner un "quantum" de temps à chaque processus pour s'exécuter sur le processeur.

Temps Partagé / Time Sharing

Un utilisateur travaille sur une machine virtuelle. Il a l'illusion d'être le seul sur cette machine et de posséder toutes les ressources (mémoire, processeur).

Pour ce faire, le système d'exploitation utilise une liste de tâches en cours. Pendant un intervalle de temps donné et fixe, le système exécute une tâche. À la fin du temps imparti, la tâche est interrompue, placée dans la liste des tâches en cours et la suivante est alors exécutée.

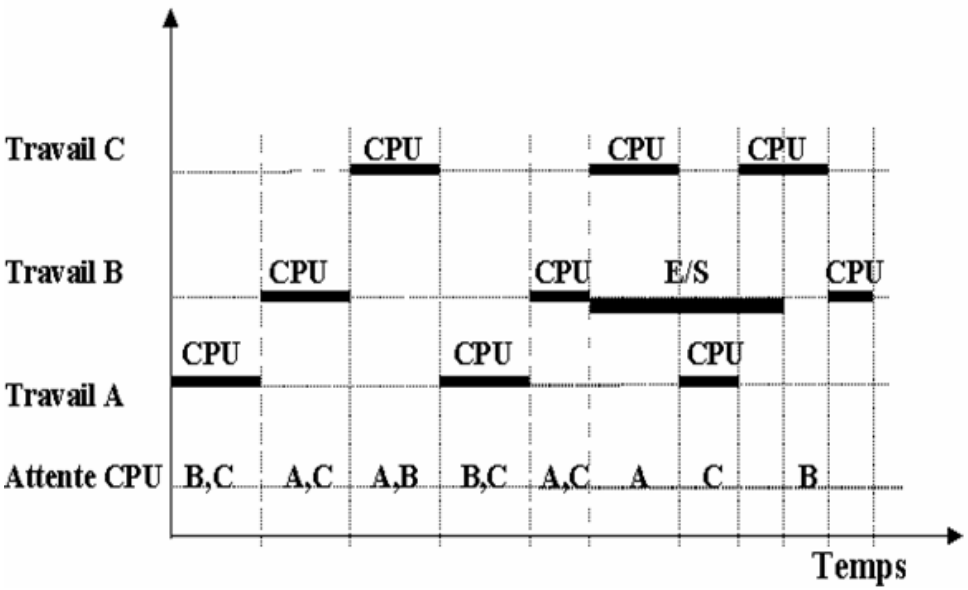
La **préemption** est le mécanisme qui permet au système d'interrompre une tâche à n'importe quel moment pour offrir les ressources de la machine à une autre.

L'utilisateur a aussi la possibilité de lancer des tâches asynchrones (tâches de fonds "interruptibles") afin de reprendre la main pour d'autres traitements. Il pourra ainsi par programmation réaliser des applications mettant en œuvre plusieurs tâches.

Temps Partagé / Time Sharing

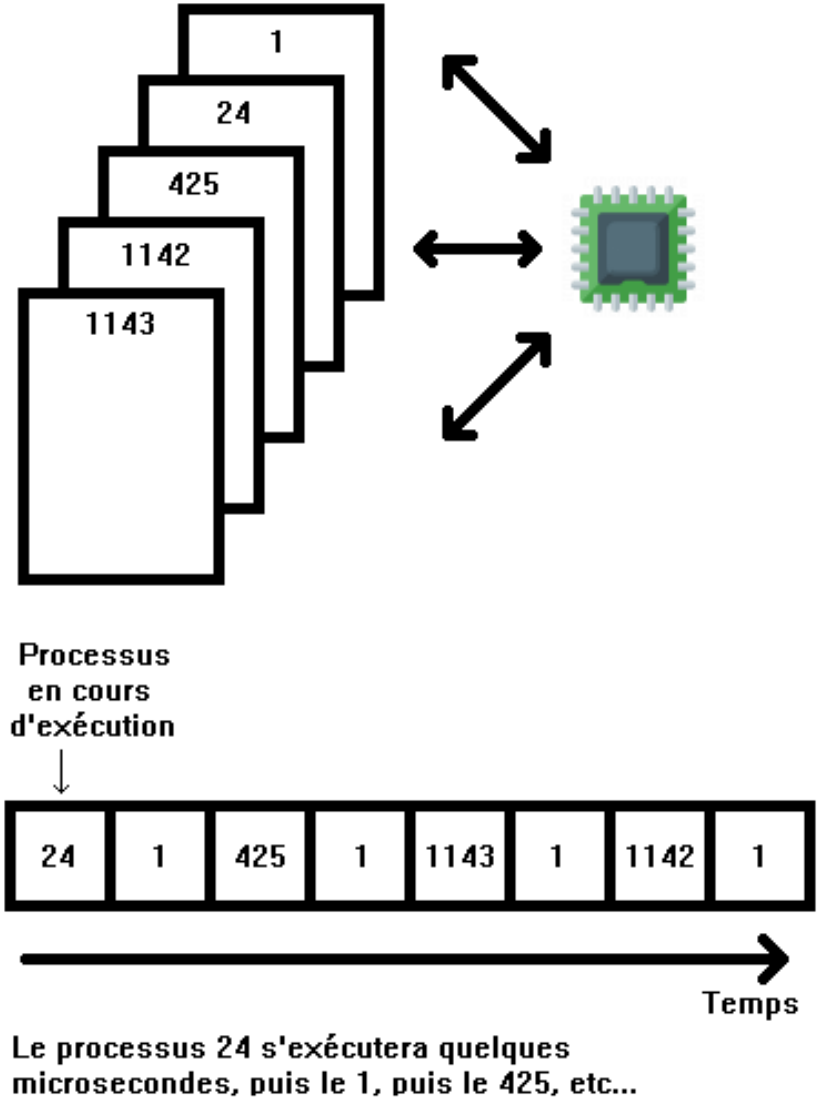
- Le processeur est alloué, à tour de rôle, pendant un certain temps à un chacun des travaux en attente d'exécution. Au bout de ce temps, l'exécution du travail en cours est suspendue. Le processeur est alors alloué à un autre travail.

- Si plusieurs utilisateurs lancent à partir de leurs terminaux leurs programmes simultanément, ce mode d'exploitation donne l'impression que les programmes s'exécutent en parallèle (pseudo parallélisme).

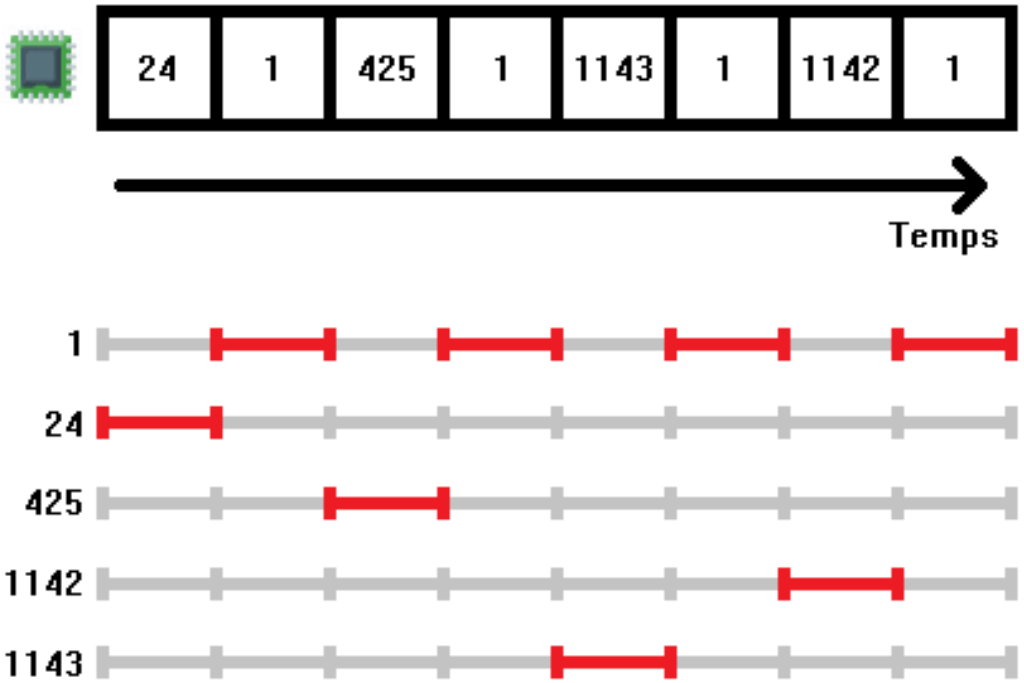


Ordonnanceur / Scheduler

- Le temps est partagé entre chaque processus pendant un court instant, imperceptible pour l'humain, mais suffisant pour gérer :
  - les entrées (souris/clavier)
  - les traitements et calculs (visite de site web, déplacement d'un personnage dans un jeu, mise à jour d'une formule dans un tableur, ...)
  - l'affichage à l'écran (écriture en mémoire vidéo)



Ordonnanceur / Scheduler



- Le processus 1 a été exécuté pendant 4 quantum de temps
- Le processus 24 a été exécuté pendant 1 quantum de temps
- Le processus 425 a été exécuté pendant 1 quantum de temps
- Le processus 1142 a été exécuté pendant 1 quantum de temps
- Le processus 1143 a été exécuté pendant 1 quantum de temps

Si le quantum est de 30ns, le processus 1 a été exécuté pendant 120ns, contre 30ns pour chaque autre processus.

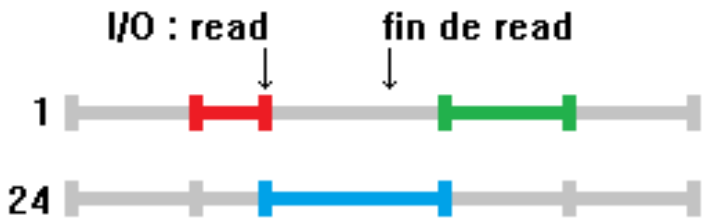
Ordonnanceur / Scheduler

- Un **quartz** sur les cartes mère crée des interruptions à intervalles régulières, les systèmes d'exploitations s'en servent pour calculer l'heure et maintenir des intervalles de temps très précises.

- Ces interruptions permettent de rendre le processeur au système d'exploitation pour "**changer de contexte**" (sauvegarder l'état du programme précédent, et recharger un autre programme à la place).

Ordonnanceur / Scheduler

Si un programme fait un appel système, son quantum de temps est réduit.
Si le syscall produit une I/O, le programme sera mis en attente tant que l'I/O n'est pas terminée (d'autres programmes se partageront le processeur).



Processus 1:
char tab[40];
int var, fd;

var = 40;
read(fd, tab, var);
var = 36 + 4;
...

syscall faisant une I/O

Processus 2:
int sum, a, b;

a = 1337;
b = 4242;
sum = a + b;
...

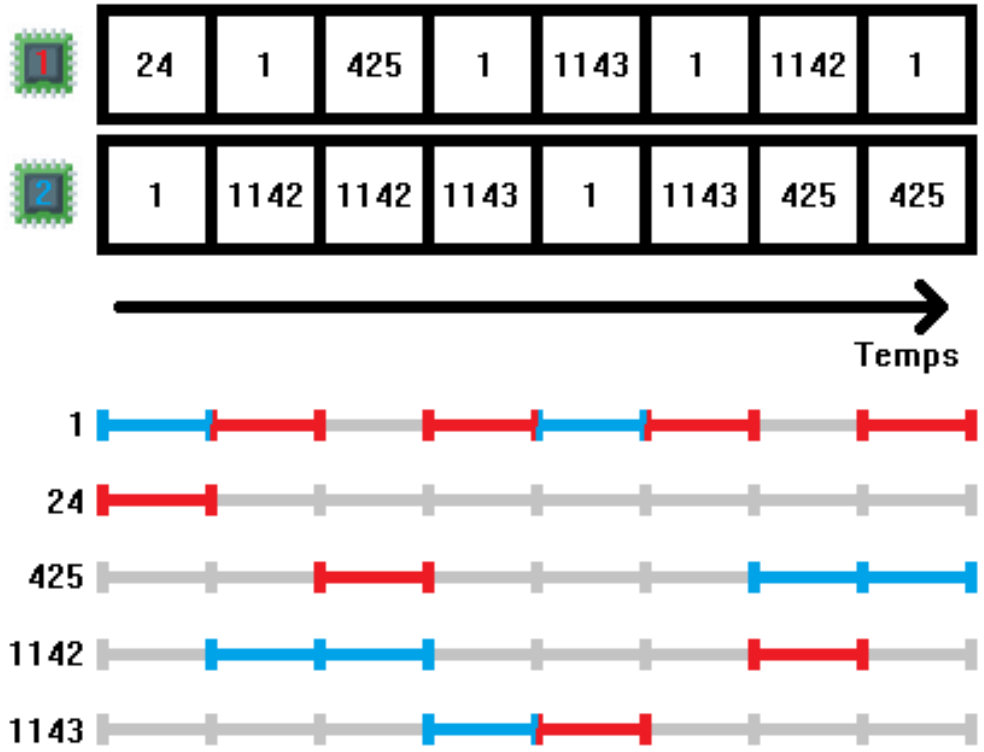
Ordonnanceur / Scheduler

- Dans les machines utilisant plusieurs processeurs, ou dont les processeurs disposent de plusieurs coeurs d'exécution (les processeurs multi-coeurs/multi-cores), les processus peuvent fonctionner en parallèle.

  Attention, il faut que les processus ne partagent aucune ressource (périphérique, page mémoire, fichier, ...).

- C'est le système d'exploitation qui va activer et utiliser ces coeurs/processeurs supplémentaires selon ses algorithmes (si l'OS n'a pas été développé pour disposer de plusieurs coeurs/processeurs, un seul coeur/processeur sera utilisé et visible par l'utilisateur de la machine).

Ordonnanceur / Scheduler



- Le processus 1 a été exécuté pendant 6 quantum de temps
- Le processus 24 a été exécuté pendant 1 quantum de temps
- Le processus 425 a été exécuté pendant 3 quantum de temps
- Le processus 1142 a été exécuté pendant 3 quantum de temps
- Le processus 1143 a été exécuté pendant 2 quantum de temps

Si le quantum est de 30ns, le processus 1 a été exécuté pendant 180ns, le processus 425 pendant 90ns, ...

Ordonnanceur / Scheduler

- La capacité d'un système à partager du temps entre plusieurs processus s'appelle :

  Système multitâches préemptif à temps partagé

- Multitâches : plusieurs programmes sont actifs
- Préemption : passer avant les autres
- Temps partagé : partage entre les processus

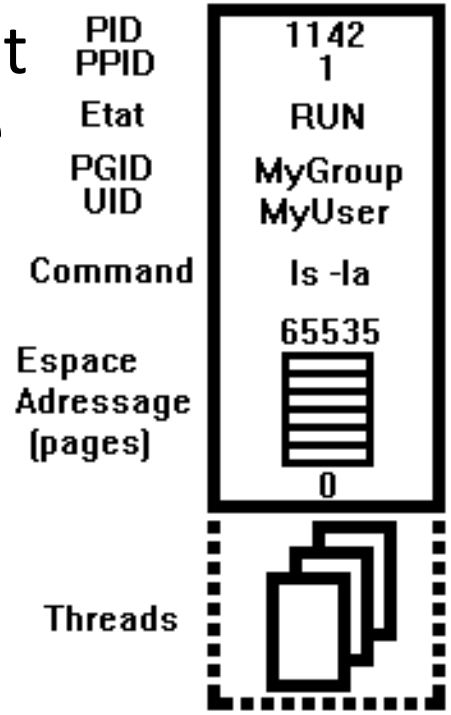
Processus Légers / Threads

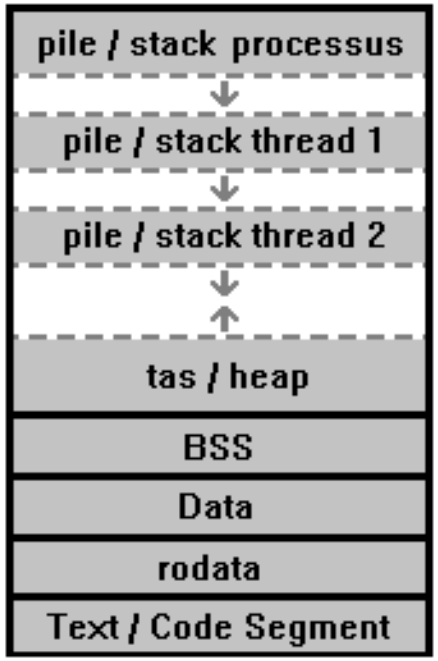
- Certains programmes effectuent parfois de multiples opérations dont les paramètres sont indépendants.

- Pour permettre un usage plus efficace du/des processeurs, on crée des "threads" (processus légers) dans lesquels certaines de ces opérations seront effectuées.

Processus Légers / Threads

Les threads sont attachés à un processus, ils partagent le même espace mémoire que le processus qui les crée (donc les données sont partagées, et les pages mémoires sont identiques), mais disposent de leur propre pile d'appel dans cet espace mémoire.





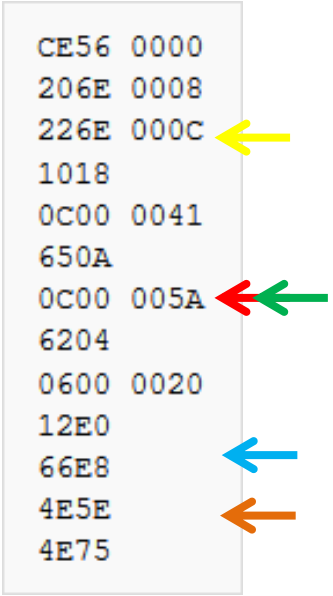
Processus Légers / Threads

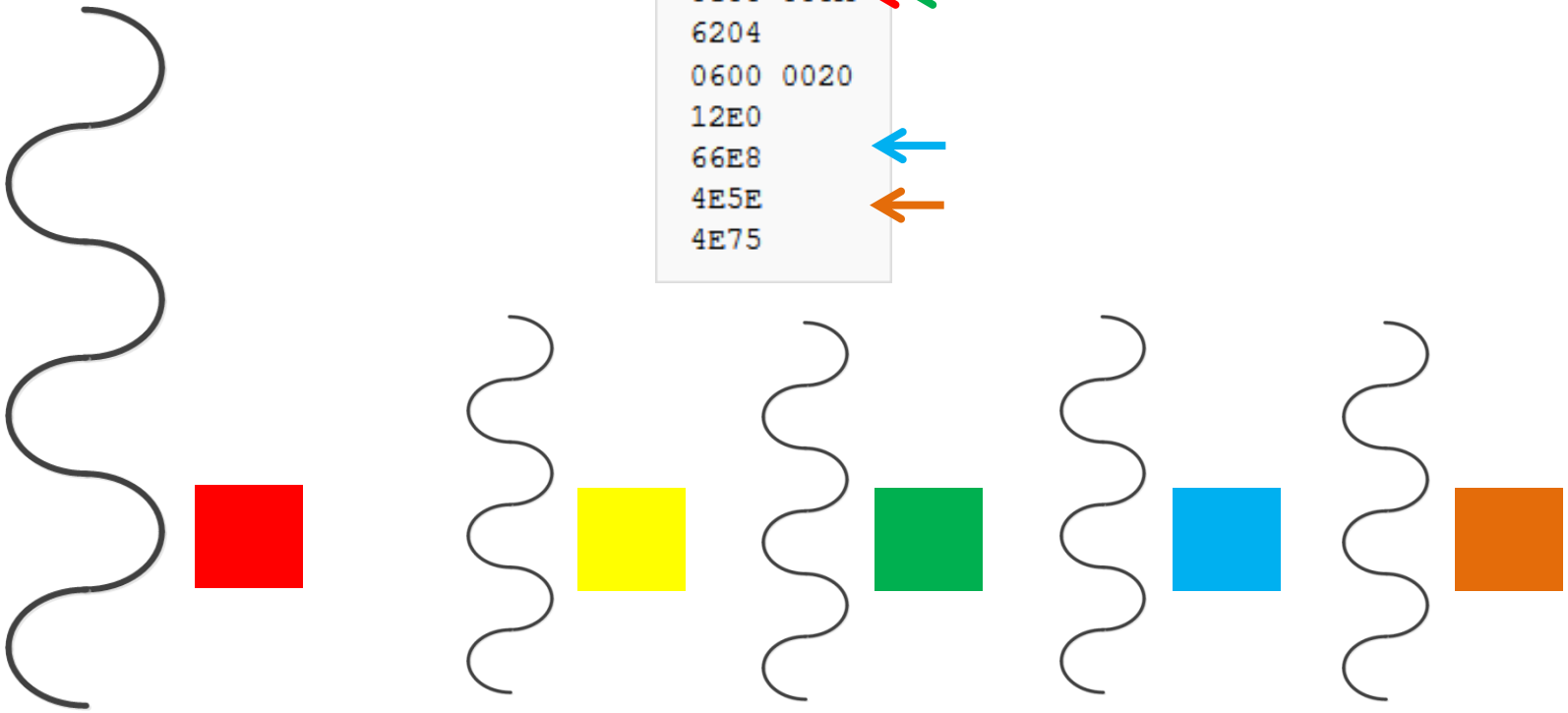
1 Processus = 1 instance du code

```
CE56 0000
206E 0008
226E 000C
1018
0C00 0041
650A
0C00 005A
6204
0600 0020
12E0
66E8
4E5E
4E75
```

4 Threads

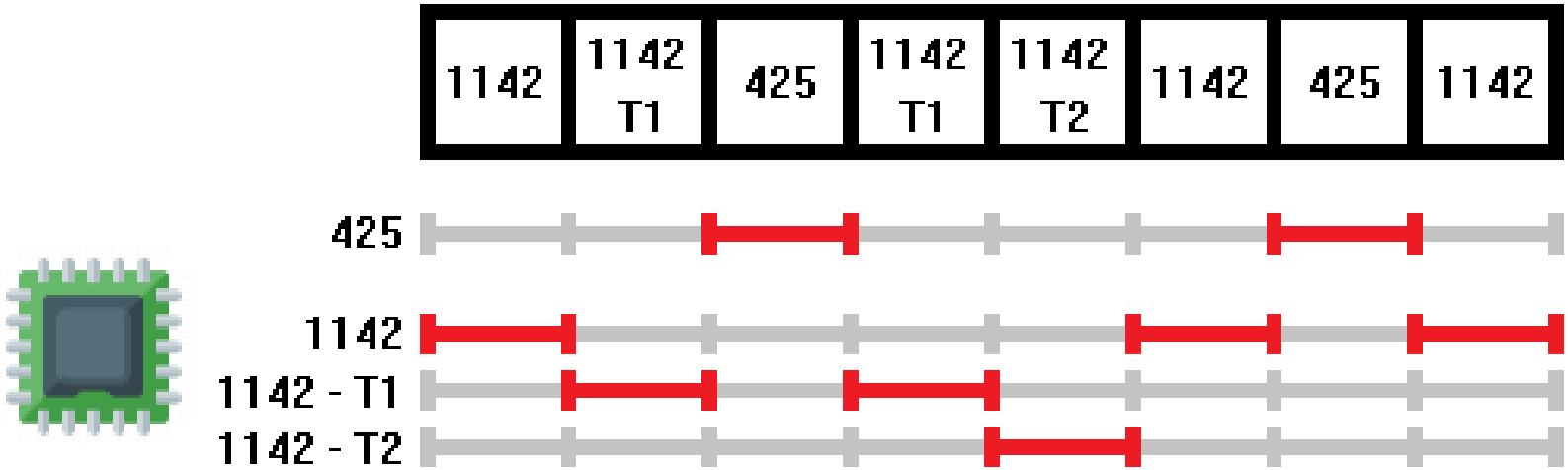
Processus Légers / Threads





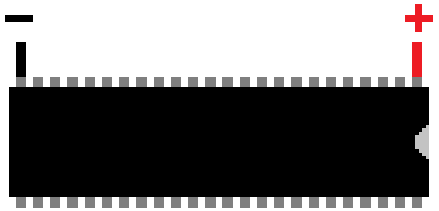
Processus Légers / Threads

- Dans l'ordonnanceur, ils sont vus comme des processus spéciaux/différents.
- Ils peuvent s'exécuter de façon concurrente (en parallèle) aux autres processus et threads.



Fichiers

- La mémoire perd son contenu, si elle est mise hors tension.



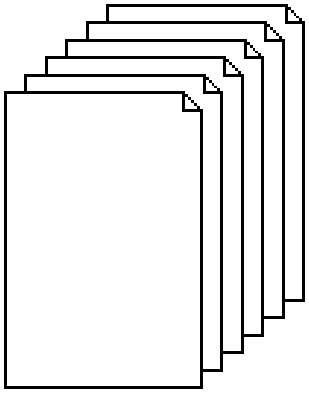
- Comment sauvegarder des données en cas de panne électrique, ou si on veut déplacer la machine, ou plus simplement déplacer les données ?

  Avec un support physique !

- Comment différencier les données entre elles ?

  Avec une organisation en fichiers !

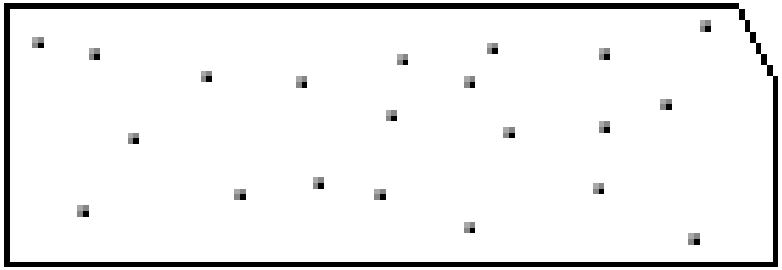
Fichiers



- Historiquement, un « *fichier* » est un meuble stockant des fiches.

- Vers les débuts des traitements automatisés, on utilisait des cartes perforées pour stocker les données et programmes (support physique lisible par un humain).

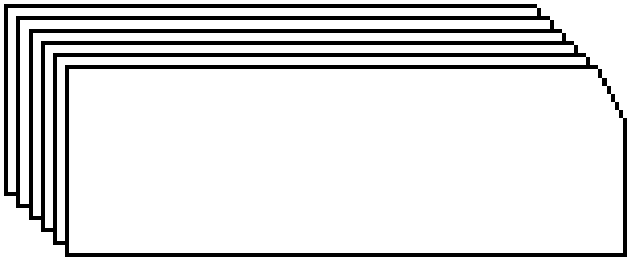
Une carte = 72 ~ 80 caractères



Fichiers

- Comment stocker de GRANDES quantités de données, en réduisant le temps de lecture des cartes ? (et leur tri si on fait tomber le carton les contenant)

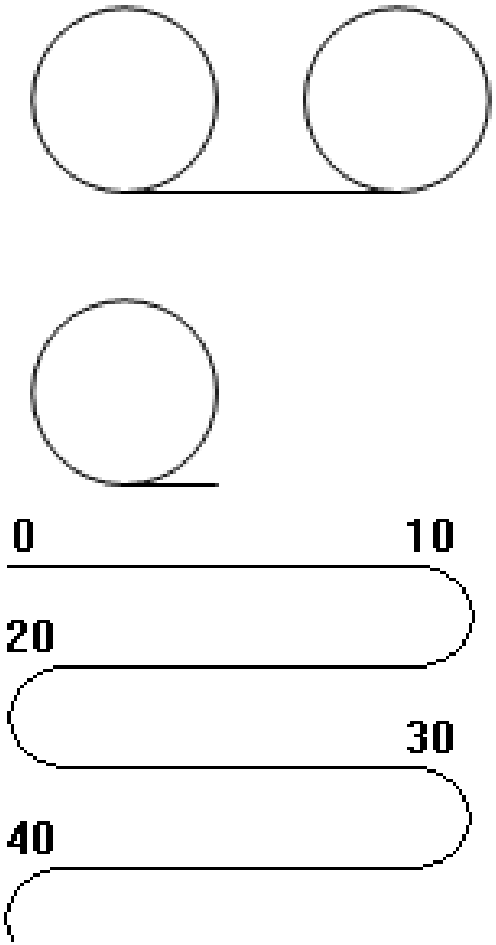
  Avec un support pré-organisé pour la machine.



Bandes Magnétiques

- Les bandes magnétiques sont plus rapides à lire, et ont une densité d'informations beaucoup plus élevée que les cartes perforées... mais elles sont un peu plus fragiles, et il est impossible pour un humain de les lire.

- **Accès séquentiel** : pour lire la case 1250, on doit dérouler toute la bande jusqu'à cette case. Pas génial si on ne veut accéder qu'à une seule donnée précise de la bande, mais parfait pour les traitements répétitifs sur de grandes quantités de données.



Fichiers

- Comment accéder à certaines données précises sans avoir à tout relire/faire passer ?

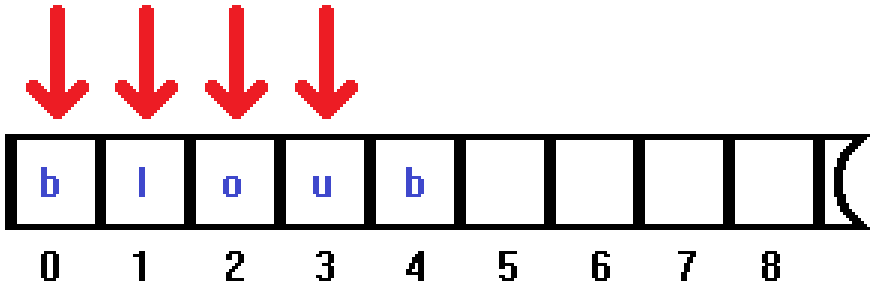
  Avec les supports à **accès aléatoire**.

- Le support des données permet d'accéder à certaines parties sans avoir à lire l'intégralité du support.
- Exemples : les disques (disques durs, disquettes, CD-ROM, ...) et les mémoires flashs.

Accès Séquentiel :

- Lire donnée 3
sauter donnée 0
sauter donnée 1
sauter donnée 2
lire donnée 3
stop, rembobiner

- Lire donnée 9
sauter donnée 0
...



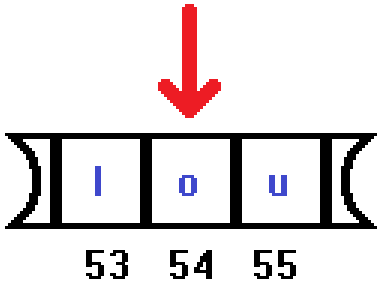
b , l , o , u => u

Accès Aléatoire :

- Lire donnée 54
placer tête sur disque 0, piste 0
lire donnée 54

- Récupérer donnée 1024
placer tête sur disque 0, piste 2
lire donnée 1024

- Récupérer donnée 39
placer tête sur disque 0, piste 0
lire donnée 39

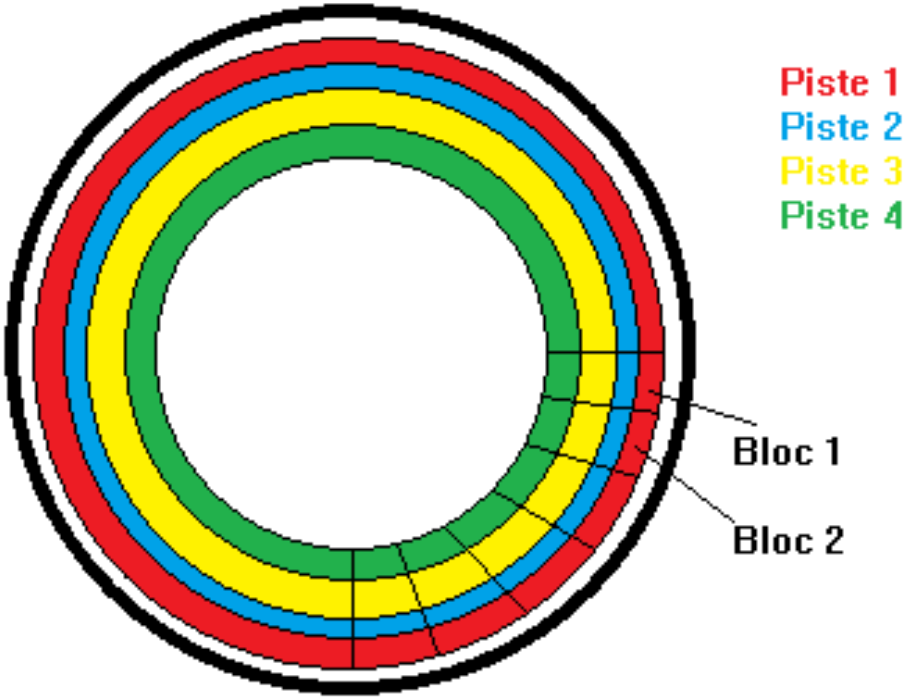


o => o

Fichiers

- Les disques contiennent des pistes qui contiennent des blocs de plusieurs octets.

- Les mémoires flashs se rapprochent des mémoires classiques, mais elles conservent leur état sans être alimentées par un courant.



Piste, Cylindre, Disque…

Cylindre = 1 Piste sur chaque Disque
Disque = toutes les Pistes d'un Disque

Fichiers

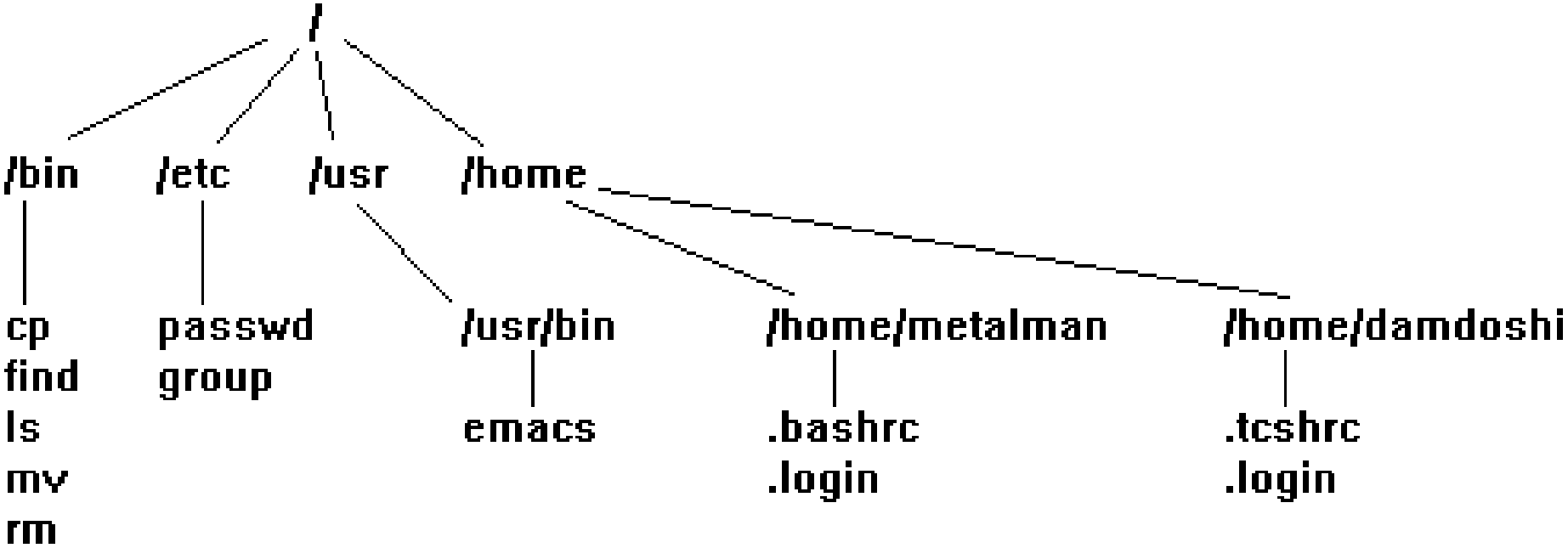
- Accès brut aux blocs de données assez rare aujourd'hui (dédié aux systèmes où les temps de réponse doivent être très réduits).

- Abstraction des *blocs* / *secteurs* / *pistes* / *cylindres* de chaque disque.

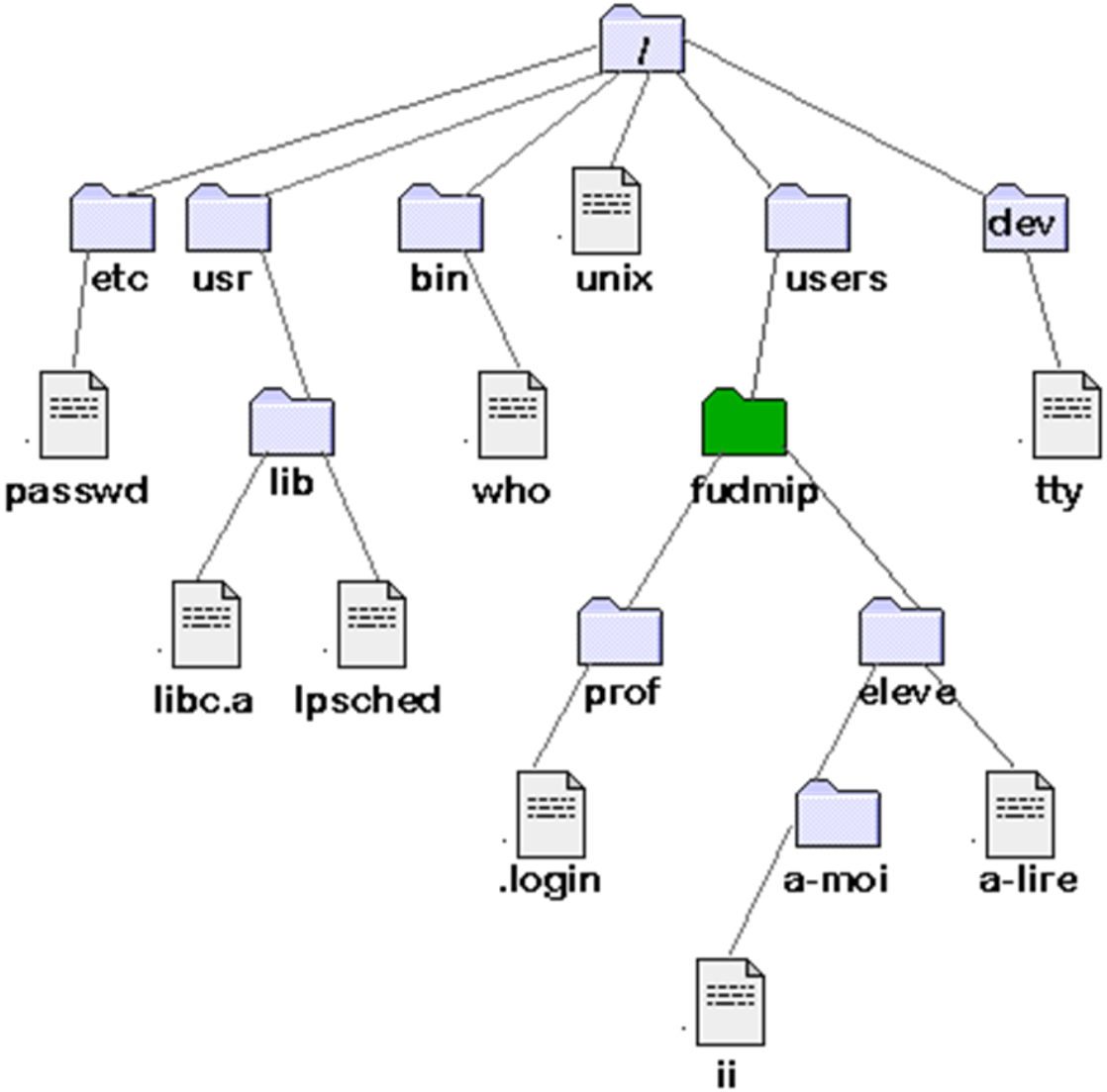
(EN : *clusters / sectors / tracks / cylinders*)

Fichiers : arborescence

- L'utilisateur ne connait pas l'état réel de son disque, il voit l'organisation abstraite sous forme de fichiers et dossiers : **l'arborescence**.



Fichiers : arborescence

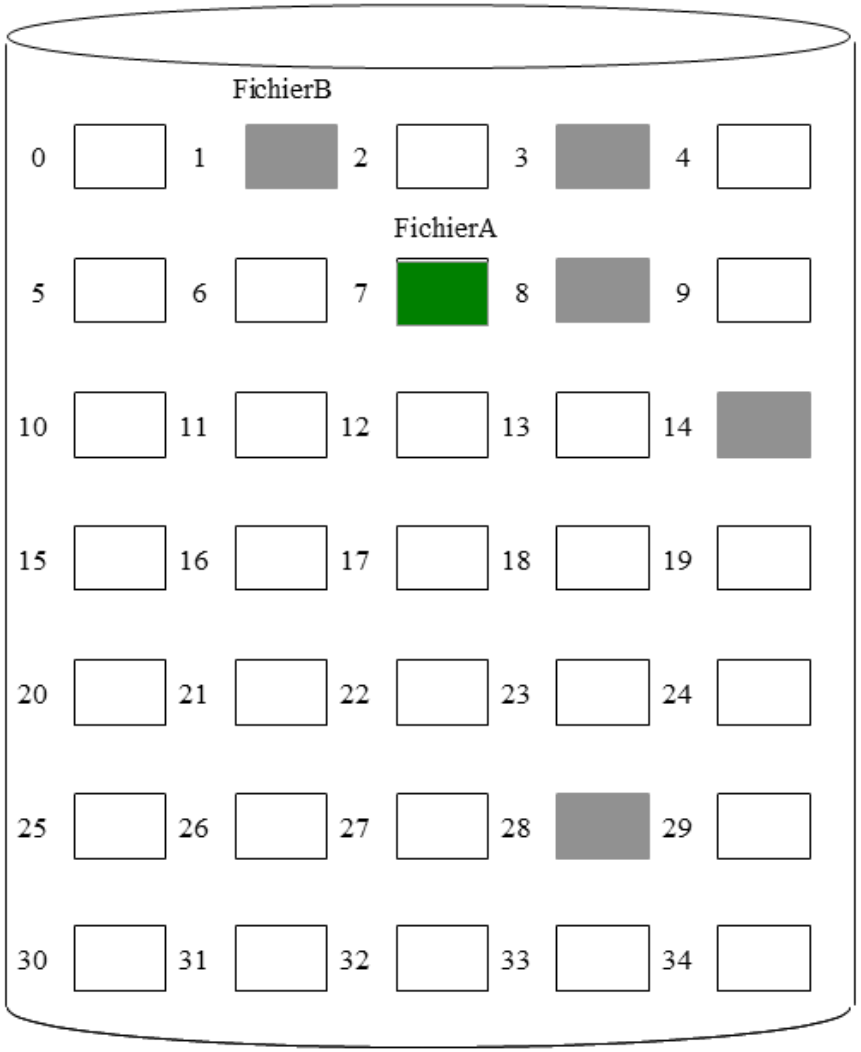


Fichiers : blocs

- Le fichier contient des données, elles sont éparpillées dans des blocs sur le disque (souvent, les blocs sont éparpillés).

	bloc 1	bloc 2	...				
piste 1				mv			
piste 2	ls		cp			find	
...		rm					emacs 4
				.tcshrc			
	emacs 1	emacs 2				.login	
		.login		emacs 3			.bashrc

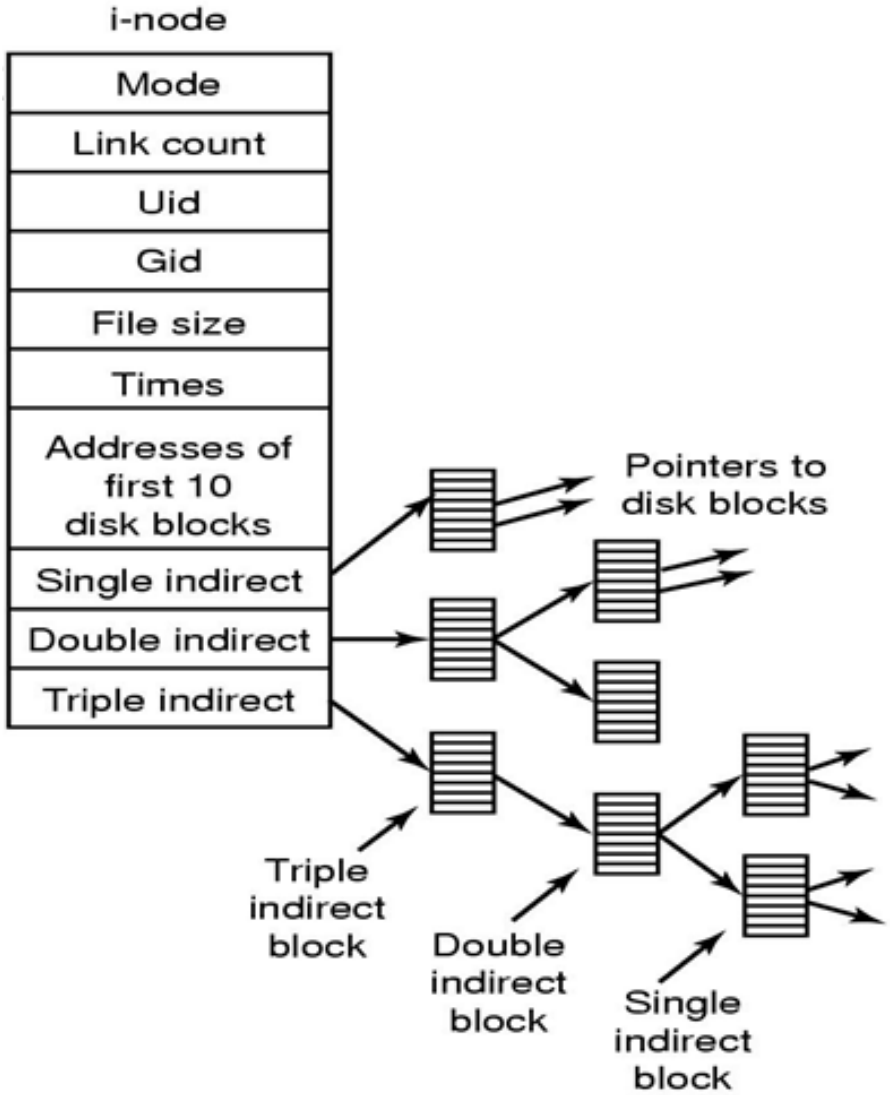
Fichiers : blocs



Nom du fichier	Blocs
Fichier A	7
...	...
Fichier B	1, 8, 3, 14, 28
...	...

Fichiers : i-nodes

- Sur UNIX, chaque fichier est identifié par un **i-node** unique (index node/nœud d'index)
- Les blocs contenant les données du fichier sont indexés dans l'**i-node**
- L'**i-node** est une structure qui contient toutes les propriétés du fichier
- Un **i-node** ne contient pas le nom du fichier



Fichiers : i-nodes

- Un **i-node** rassemble plusieurs informations :
  - La taille (en octets)
  - Les adresses des blocs utilisés sur le disque
  - L’identification du propriétaire du fichier
  - Les droits d’accès (chmod et ACL)
  - Le type du fichier (ordinaire, spécial, …)
  - Un compteur de liens
  - Les dates d’opérations principales (création, modification, consultation)

Fichiers : système de fichiers

- Un **File System** (ou FS, ou système de fichiers) est un système permettant d'organiser et stocker des fichiers
  - Il permet d'exploiter les disques pour y placer des fichiers et les retrouver grâce à leurs noms
  - Il contient des méta-données concernant les fichiers (droits d'accès, data de modification, type de fichier, …)
  - Des limitations sont imposées selon le système de fichier (longueur de nom, taille minimale occupée, …)

- Les systèmes de fichier utilisent divers algorithmes pour organiser et retrouver leurs fichiers (liste chaînée, hash table, B-tree, …)

Fichiers : système de fichiers

- De nombreux systèmes de fichiers existent :
  - FAT, FAT32, exFAT, …
  - NTFS, ReiserFS, …
  - ext2, ext3, ext4, UFS, …
  - ZFS, …
  - Joliet/ISO 9660 (CD-ROM), …

- Il est possible de « couper » un disque en plusieurs parties qui seront indépendantes les unes des autres : les partitions
  - Chaque partition disposera de son propre FS

Fichiers : système de fichiers

- Sur UNIX, toutes les partitions et tous les disques sont abstraits derrière l'arborescence
  - On travaille depuis l'arborescence (/)
  - Même les disques réseaux sont abstraits
  - …cependant, la connaissance du support sous-jacent permet de ne pas être surpris si une opération est refusée par le FS

- *Sur Windows, on travaille depuis les partitions et disques qui démarrent par des lettres (C:, E:, …)*

Fichiers : système de fichiers

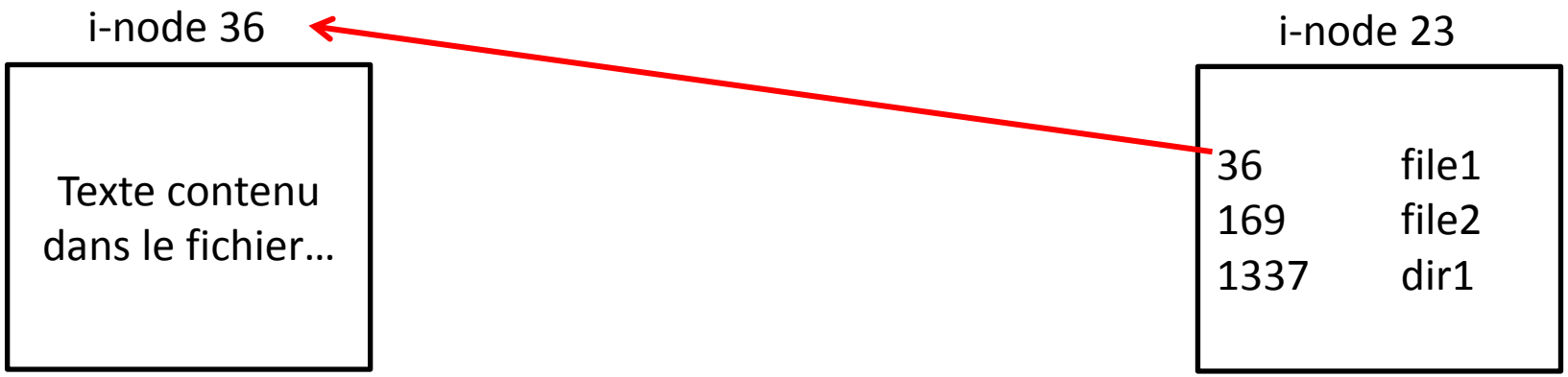
- Les FS pour UNIX contiennent :
  - un superbloc
    (il décrit la table des i-nodes, les informations de la partition dans laquelle se situe le FS, la liste des blocs vides, …)
  - une table d'i-nodes
    (elle contient un nombre fixé d'i-nodes, donc le nombre maximum de fichiers possibles dans la partition)
  - l'espace pour les fichiers et dossiers
    (l'ensemble des blocs pour y stocker le contenu des fichiers et dossiers)

- On ne peut pas toujours élargir une partition après l'avoir créée…

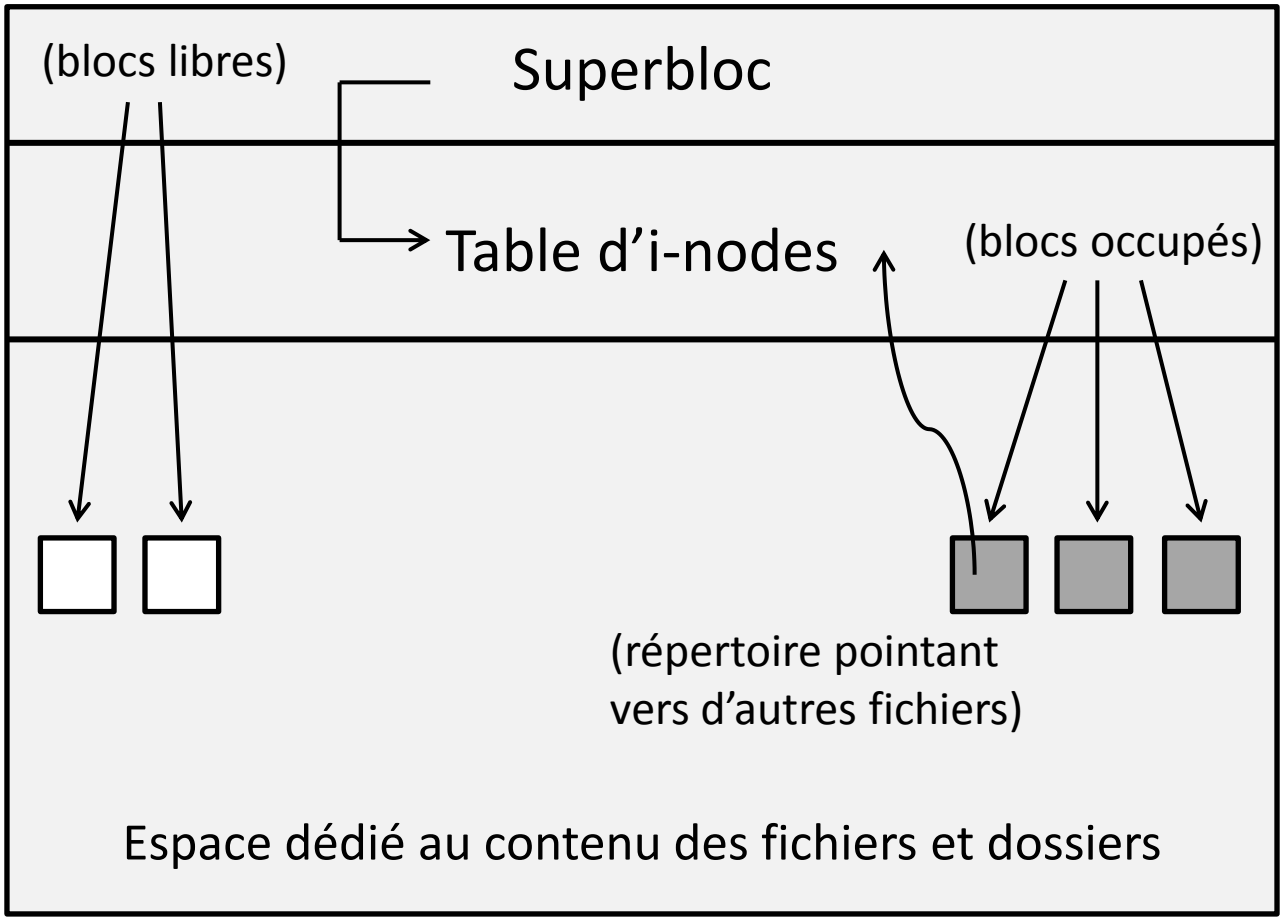
Fichiers : système de fichiers

- Les blocs constituant un fichier contiennent :
  - les données brutes stockées dans le fichier

- Les blocs constituant un dossier contiennent :
  - la liste des noms de fichier contenus dans le dossier
  - le numéro de l'i-node associé à chaque fichier contenu



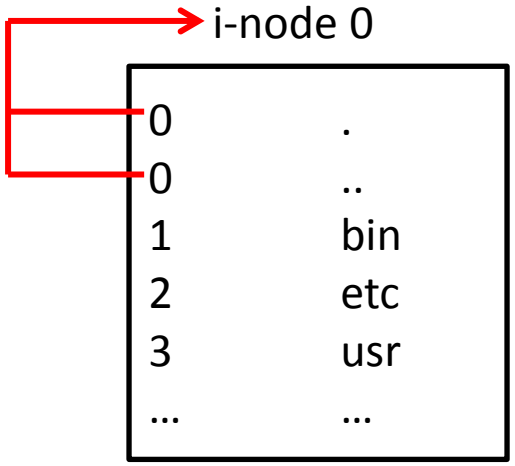
Fichiers : système de fichiers



Partition contenant un système de fichiers compatible UNIX

Fichiers : système de fichiers

- Cas de la racine : /
  - / est son propre père
  - Tous les autres dossiers ont un parent différent d'eux-même



Fichiers : manipulation

- Plusieurs syscalls permettent de manipuler les fichiers :

open	Ouvre ou crée un fichier en lecture, écriture, ou les 2
read	Lit un fichier/Extrait des octets vers un buffer
write	Écrit un fichier/Inscrit des octets depuis un buffer
close	Ferme un fichier
lseek	Déplace le pointeur de lecture/écriture dans le fichier
stat	Obtient les informations du fichier (infos de l'i-node)

Fichiers : manipulation

- Plusieurs programmes (s'appuyant sur les syscalls) permettent de manipuler les fichiers :

ls	Liste les fichiers et dossiers
cp	Copie un/des fichiers ou dossiers
rm	Supprimer un/des fichiers ou dossiers
mv	Déplace un/des fichiers ou dossiers
mkdir	Créer un/des dossiers
rmdir	Supprime un/des dossiers

Liste non exhaustive

Fichiers : types de fichiers

- **Fichiers Ordinaires** : les fichiers contenant des données, du texte, ou des programmes exécutables

- **Fichiers Spéciaux** : les périphériques, les tubes/pipes, ou IPC
  - Mode caractère : E/S réalisées caractère par caractère (terminaux, imprimantes, …)
  - Mode bloc : E/S réalisées par blocs de caractères (disques, …)

- **Répertoires** : contiennent les couples (i-node, nom de fichier)
  - Création, modification, lecture, effacement par primitives systèmes spécifiques
  - Répertoires sont aussi appelés « catalogues » ou « directories »

Fichiers : types de fichiers

Fichier Ordinaire	**-**
Répertoire	**d**
Fichier Spécial (accès en mode caractère)	**c**
Fichier Spécial (accès en mode bloc)	**b**
Tube nommé (FIFO)	**p**
Lien Symbolique	**l**
Socket	**s**

Fichiers : convention de nommage

/	« Slash »	*Racine/Root* de l'arborescence + Séparateur de dossiers
.	« Point »	Répertoire courant
..	« Point Point »	Répertoire parent du dossier courant
~	« Tilde »	« *Home Directory* » Dossier personnel de l'utilisateur courant

Fichiers : convention de nommage

- Le **chemin d'accès** est la chaîne de caractère permettant d'accéder/identifier un fichier précis

- C'est une suite de noms de répertoires se terminant par le fichier ou dossier visé

/usr/bin/convert

- **/** - racine
- **usr** - dossier
- **bin** - dossier
- **convert** - fichier ou dossier

Fichiers : convention de nommage

- Chemin absolu :

  on indique le chemin depuis la racine *(démarre par /)*

  /usr/local/bin/ffmpeg

- Chemin relatif :

  on indique le chemin depuis le dossier courant

  ../../local/bin/ffmpeg

Fichiers : commandes courantes

- ls Lister le contenu d'un répertoire
- cd Changer de dossier/Se déplacer

- cp Copier un fichier (cp -r pour dossier)
- mv Déplacer un fichier/dossier
- rm Supprimer un fichier (rm -r pour dossier)

- mkdir Créer un dossier
- rmdir Supprimer un dossier vide

- touch Mettre à jour la date d'accès (…et créer fichier)

Fichiers : exemple « `ls -al` »

Répertoire

Nombre de liens du fichier

Nom du fichier

```
$ ls -al
total 106
drwxr-xr-x   4  toto  crir     512   Jul 19 10:25   .
drwxr-xr-x 2113 root  root   36352   Jul 22 12:22   ..
drwx------   2  toto  crir     512   Sep  1  1998   .elm
-rw-r--r--   1  toto  crir     157   Aug 18  1998   .profile
-rw-------   1  toto  crir     195   Oct  6  1998   dead.letter
drwx------   2  toto  crir     512   Sep  1  1998   Mail
-rw-------   1  toto  crir   11689   May 27 10:24   mbox
```

Fichier

Propriétaire

Nom du groupe

Taille en octets

Date de la dernière modification

Droits sur Fichiers

- Trois propriétés « classiques » :
  - Propriétaire
    usage générique : créateur du fichier
  - Groupe Propriétaire
    usage générique : ceux qui travailleront avec le fichier
  - Les autres
    usage générique : le fichier est-il public ou non

Droits sur Fichiers

- Trois droits « classiques » :
  - Lire
    - Lit le contenu d'un fichier
    - Lit les fichiers contenus dans un dossier
  - Ecrire
    - Modifier le contenu d'un fichier
    - Modifier le contenu d'un dossier
  - Exécuter
    - « Lancer »/Exécuter le fichier (ex : *./monscript.sh*)
    - ATTENTION ! N'empêche pas de faire : *sh monscript.sh*
    - Autorise de passer dans un dossier (mais voir le contenu)

Droits sur Fichiers

Quels droits :		Pour qui :	
- Exécution : 1	(001)	- Moi (UID)	-XXX- - - - - -
- Ecriture : 2	(010)	- Mon Groupe (GID)	- - - -XXX- - -
- Lecture : 4	(100)	- Le reste	- - - - - - -XXX

- Principe du masque binaire :

Exécution + Ecriture + Lecture = 1 + 2 + 4 = 7 (111)

Exécution + Lecture = 1 + 4 = 5 (101)

- Donner tous les droits à moi, et seulement certains à mon groupe, et exécution au reste :

chmod 751 MonFichier => -rwxr-x- -x MonFichier

Droits sur Fichiers

Quels droits :

- **Exécution : x**
- **Ecriture : w**
- **Lecture : r**

Pour qui :

- **Moi (user) : u**
- **Mon Groupe (group) : g**
- **Le reste (other) : o**

- Méthode avec des lettres :

Donner tous les droits à "moi" : **u=rwx**

Pareil + mon groupe lit et exécute : **u=rwx,g=rx**

Pareil + mon groupe et le reste lisent et exécutent : **u=rwx,go=rx**

Retirer tous les droits aux groupe et autres : **go=**

```
chmod u=rwx,go=rx MonFichier   =>  -rwxr-xr-x    MonFichier
chmod ugo=  Monfichier         =>  ----------    MonFichier
```

Droits sur Fichiers

- Droits avancés : ACL (Access Control List)
  - Maîtrise plus fine des groupes & personnes

    Autoriser des groupes et personnes précis à avoir des droits précis
  - Maîtrise plus fine des droits

    Accéder aux méta-données, changer les droits, modifier le contenu d'un fichier, ajouter un fichier, …

Utilisateurs & Groupes

- Utilisateurs de la machine déclarés dans :
  `/etc/passwd`

- Groupes de la machine déclarés dans :
  `/etc/group`

(toutes ces définitions sont dans le contexte UNIX)

Utilisateurs

- `/etc/passwd` contient dans l'ordre :
  - login
  - mot de passe chiffré (ou x)
  - numéro unique d'utilisateur (UID)
  - numéro unique de groupe (GID)
  - nom complet de l'utilisateur
  - répertoire utilisateur (HOME directory)
  - interpréteur de commande (SHELL)

- Le tout séparé par des **:** (deux points)

Utilisateurs

- /etc/passwd

```
root:JuemgoFbPLY:0:3::/root:/bin/sh

daemon:*:1:5::/:/bin/sh

bin:*:2:2::/bin:/bin/sh

bilbo:gzHCK12Vq99:101:102:Utilisateur1:/home/bilbo:/bin/sh

ygir:emWAhG7HfZ:202:901:YvonneGIRARD,SCIPRE,,:/users/ygir:/bin/sh
```

Utilisateurs

- Utilisateur spécial : `root`
  - « Super-utilisateur » de l'OS
  - N'est PAS le « Super-utilisateur » du processeur !
  - Dispose de tous les droits dans le système de fichiers
  - UID réservé : 0

- Un utilisateur classique peut demander des droits « supérieurs » temporairement avec `sudo`
- Commande `su` permet de changer de user courant

Groupes

- `/etc/group` contient dans l'ordre :
  - nom de groupe
  - mot de passe chiffré (ou x) <= vide en général
  - numéro unique de groupe (GID)
  - la liste des utilisateurs du groupe
    (un utilisateur peut appartenir à plusieurs groupes)

- Le tout séparé par des **:** (deux points)

Groupes

- /etc/group

```
scipre:*:901:bdeco,stage_sa,www,zebulon,yoel,cd,testpth,bontems

mmod:*:907:nicolini,laffeach,heymann,www

iufm:*:908:michaux,www
```

Utilisateurs & Groupes

- Lorsque qu'un utilisateur s'authentifie :
  - L'OS vérifie que le login existe
  - L'OS vérifie que le mot de passe est correct
  - L'OS vérifie que l'utilisateur a le droit d'accéder à la machine (en fait, que le shell existe)
  - L'OS prépare l'environnement (variables HOME, …)
  - L'OS lance le shell (champ SHELL)
  - Le shell lit plusieurs fichiers (*.profile*, *.shrc*, …)

(chaque shell dispose de son fichier : `.bashrc` `.tcshrc`)

Utilisateurs & Groupes

- Généralement, les fichiers `.login` ou `.profile` sont lus lorsque l'utilisateur s'authentifie

- Les `.*rc` sont lus lorsque le shell est (re)lancé

- S'il existe, le fichier `.logout` est lu lorsque l'on se déconnecte

Utilisateurs & Groupes

- */etc/passwd* et */etc/group* servent pour les utilisateurs et groupes « locaux » à la machine

- Dans la réalité des entreprises, les machines sont en réseau, et les utilisateurs & groupes déclarés dans un annuaire global (LDAP en général)
  - Les utilisateurs disposent d'un dossier personnel sur une machine en réseau qui contient les `.profile` et autres

- Les utilisateurs locaux ne deviennent accessibles que si la machine est hors réseau OU qu'elle n'a pas été configurée pour utiliser LDAP