INF2610
Noyau d'un système d'exploitation

Chapitre 1 - Concepts généraux

Département de génie informatique et génie logiciel

POLYTECHNIQUE MONTRÉAL
AFFILIÉE À L'UNIVERSITÉ DE MONTRÉAL

Automne 2019

Sommaire

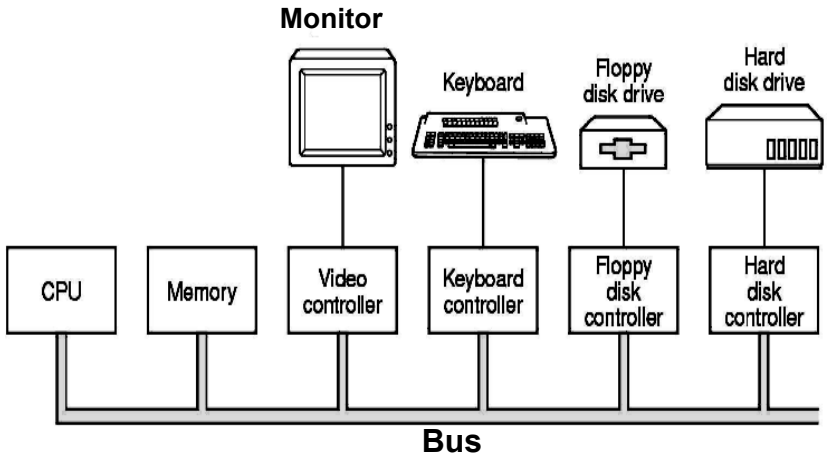
- Qu’est ce qu’un système d’exploitation ?
- Concepts de base
- Interface avec le matériel
- Interactions utilisateur/système
- Evolution du mode d’exploitation
- Structure des systèmes d’exploitation

Qu'est-ce qu'un système d’exploitation ?

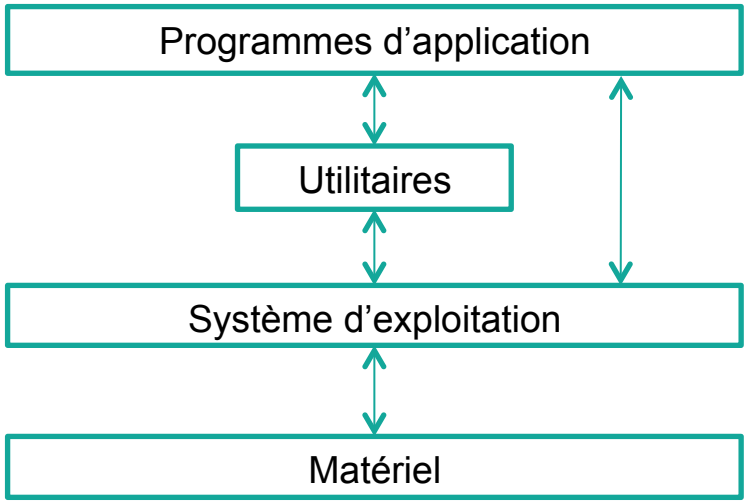
Qu'est-ce qu'un système d'exploitation ?

Ordinateur

- Matériel :



- Logiciels :



Qu'est-ce qu'un système d’exploitation ? (2)

Un système d’exploitation est un logiciel qui :

- gère les composants de l’ordinateur, et
- fournit une base (une machine virtuelle) sur laquelle sont construits les programmes d’application et les utilitaires.

  ➔ services = {appels système}

But : Développer des applications sans se soucier des détails de fonctionnement et de gestion du matériel.

Qu'est-ce qu'un système d’exploitation ? (3)

Fonctions principales d’un système d’exploitation :

- chargement et lancement des programmes,
- gestion du matériel (processeurs, mémoire physique, bus et périphériques),
- gestion des processus (programmes en cours d’exécution),
- gestion des fichiers,
- protection contre les erreurs et détection des erreurs, etc.

Qu'est-ce qu'un système d’exploitation ? (4)

- Parmi les systèmes d’exploitation qui ont été ou sont actuellement populaires, on peut citer :
  - Windows (et MS-DOS),
  - Mac-OS,
  - Unix (AIX, Xenix, Ultrix, Solaris, BSD, etc.), et
  - GNU/Linux.

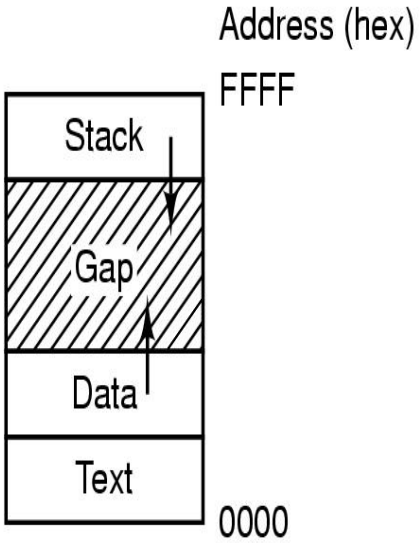
- On retrouve également le système d’exploitation éducatif *Minix 2* développé par *Andrew S. Tanembaum* et le simulateur de systèmes d’exploitation Nachos 3.

- Systèmes d'exploitation écrits majoritairement en langage de haut niveau.

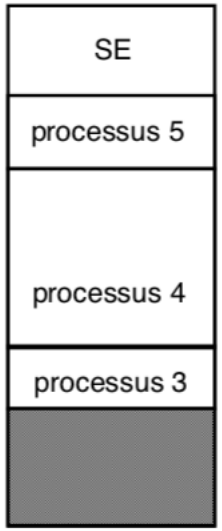
Concepts généraux

Concepts généraux

Processus : programme en cours d'exécution (espace d'adressage, état, compteur ordinal, sommet de pile, etc.).



Espace d'adressage d'un processus



Mémoire physique

Concepts généraux (2)

Mémoire virtuelle :

- Espace d'adressage d'un processus est divisé en blocs (pages ou segments).
- Lors de l'exécution du processus, les blocs sont chargés à la demande dans la mémoire physique.

➔ Taille de l'espace d'adressage peut être plus grande que celle de la mémoire physique.

Concepts généraux (3)

Fichiers

Fichiers ordinaires

Données stockées sur certains périphériques, comme le disque dur (ex: un fichier texte)

Fichiers spéciaux

Périphériques d'E/S (clavier, moniteur, disque, réseau, etc.) listés dans le répertoire **/dev**.

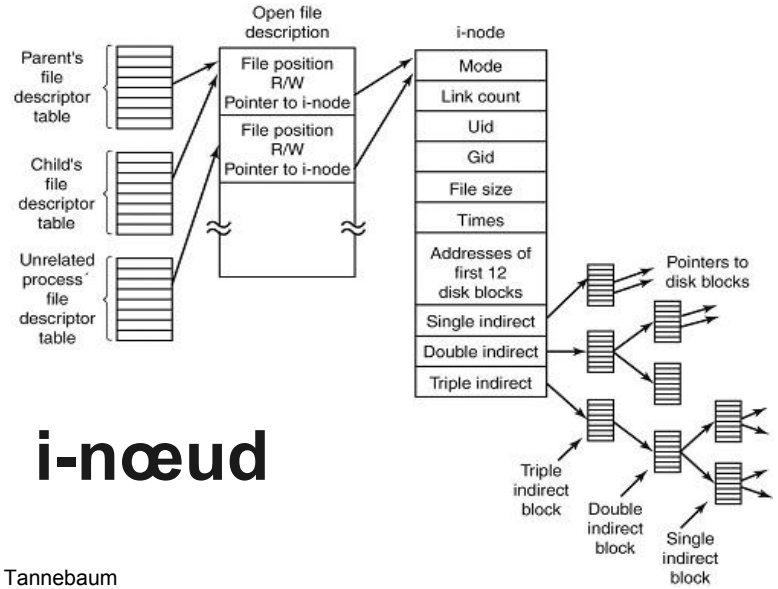
```
$ echo INF2610 > /dev/tty
```

Affichera INF2610 à l'écran

Si vous naviguez dans `/dev/`, vous trouverez des fichiers nommés stdin, stdout et stderr. Ce sont les E/S standards!

Concepts généraux (4)

- Sous UNIX/Linux, un fichier est **représenté** par une structure de données appelée i-nœud (i-node)



i-nœud

L'i-nœud contient :

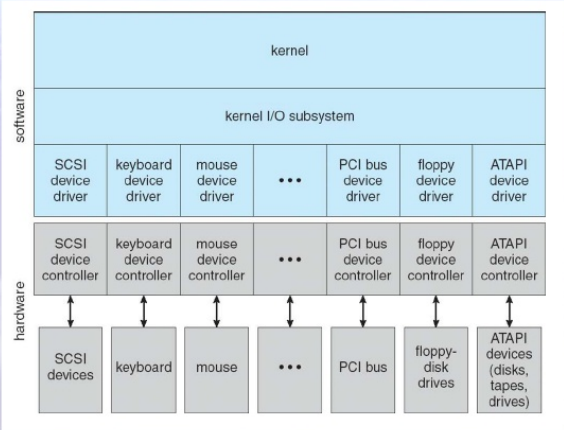
- type et mode du fichier,
- le nombre de liens vers le fichier,
- l'identifiant du propriétaire,
- l'identifiant du groupe du propriétaire,
- la taille du fichier,
- le périphérique représenté par ce fichier (si le fichier représente un périphérique,
- ...

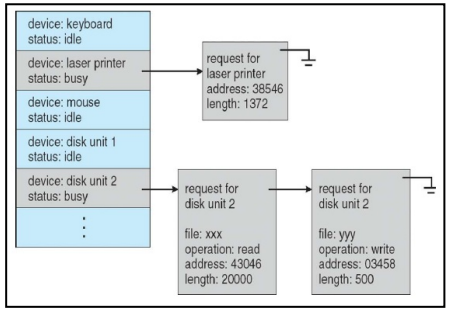
Tannebaum

http://man7.org/linux/man-pages/man7/inode.7.html

Interface avec le matériel

Interface avec le matériel





http://iips.icci.edu.iq/images/exam/Abraham-Silberschatz-Operating-System-Concepts---9th2012.12.pdf

- **Contrôleur d'un composant** : assure son fonctionnement et les interactions avec les autres composants.

- **Pilote** : procure une **interface** au noyau pour communiquer avec un périphérique via des fonctions simples (open, read, write, close, etc.)

Interface avec le matériel (2)

Techniques de gestion des E/S :

- PAR SCRUTATION: Le système vérifie régulièrement l'état des périphériques afin de détecter un changement d'état et d'agir en conséquence.

- PAR INTERRUPTIONS: Le périphérique a la responsabilité de signaler une interruption lors de son changement d'état. Le système reçoit cette interruption et agit en conséquence.

➔ E/S dirigées par des interruptions.

Interface avec le matériel (3)

Analogie de la gestion des E/S: Travail d'équipe

Supposons que vous devez savoir quand votre coéquipier, travaillant loin de vous, a terminé sa tâche.

- **Par scrutation**: Vous décidez d'appeler votre collègue à toutes les 20 minutes pour lui demander s'il a terminé jusqu'à ce qu'il ait terminé.

- **Par interruption**: Avant de quitter l'école, vous demandez à votre coéquipier de vous appeler lorsqu'il aura terminé sa tâche.

Interface avec le matériel (4)

Interruptions :

Interruptions matérielles générées par des :

- horloges (interruptions après un certain délai),
- périphériques (signalées lors du début ou de la fin d'une E/S), etc.

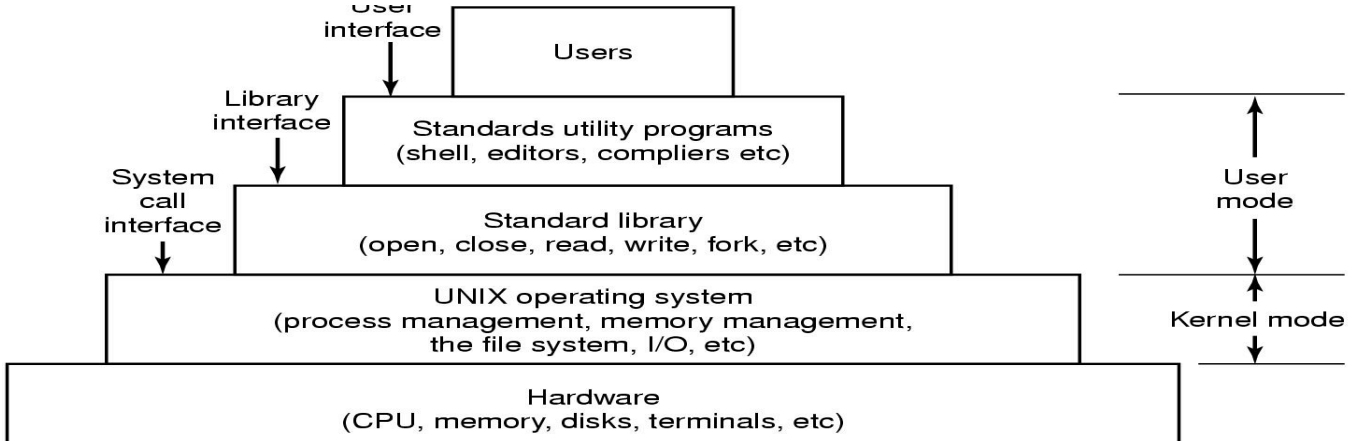
Interruptions logicielles générées par des :

- erreurs arithmétiques (division par 0),
- données non disponibles en mémoire (défaut de page),
- appels système.

Interactions utilisateur/système

Interactions utilisateur/système

- Système d'exploitation : un logiciel qui offre un ensemble de services (appels système).
- **POSIX** (*Portable Operating System Interface*) : normes IEEE de standardisation des interfaces de programmation système basées sur le système d'exploitation Unix.

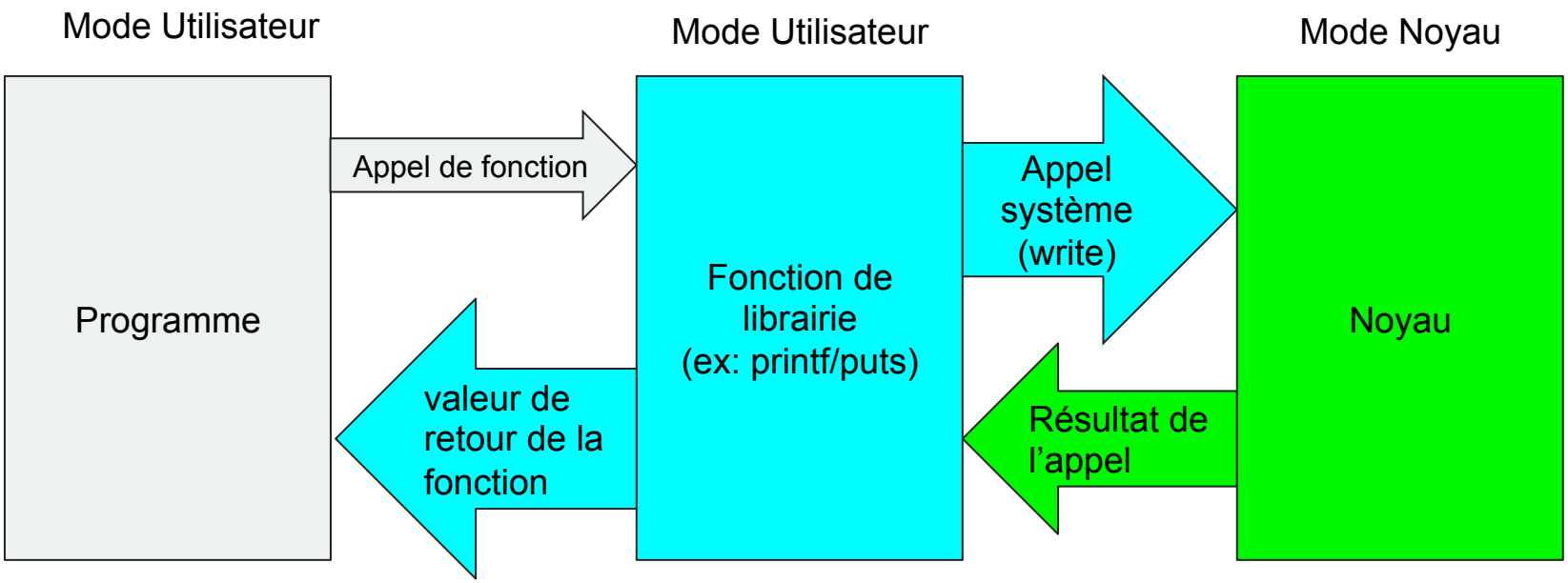


Interactions utilisateur/système (2)

- En général, les processeurs ont deux modes de fonctionnement :
  - le mode noyau (superviseur ou maître) réservé au système d'exploitation, où toutes les instructions sont autorisées.
  - le mode utilisateur (esclave) imposé aux programmes des utilisateurs et les utilitaires, où certaines instructions ne sont pas permises.

- Ces modes de fonctionnement visent à assurer la protection du système d'exploitation contre les intrusions et les erreurs.

Interactions utilisateur/système (3)



Interactions utilisateur/système (4)

- Un appel système consiste en une interruption logicielle qui a pour rôle d'activer le système d'exploitation :
  - changer le mode d'exécution pour passer du mode utilisateur au mode noyau (exemple, sous Linux: syscall);
  - récupérer les paramètres et vérifier la validité de l'appel;
  - exécuter la fonction demandée;
  - récupérer la (les) valeur(s) de retour;
  - retourner au programme appelant avec retour au mode utilisateur.

Interactions utilisateur/système (5)

- Les appels systèmes permettent de:
  - créer, attendre et terminer des processus,
  - établir une communication interprocessus (segments de données, partagés, fichiers, tubes, etc.)
  - synchroniser des processus, etc.

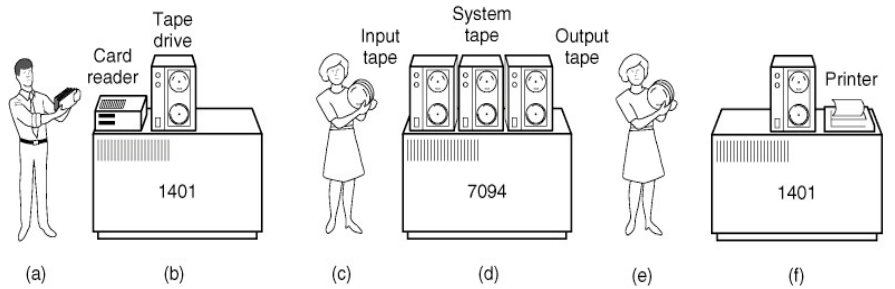
- Pour lister les appels système : man syscalls

Évolution du mode d’exploitation

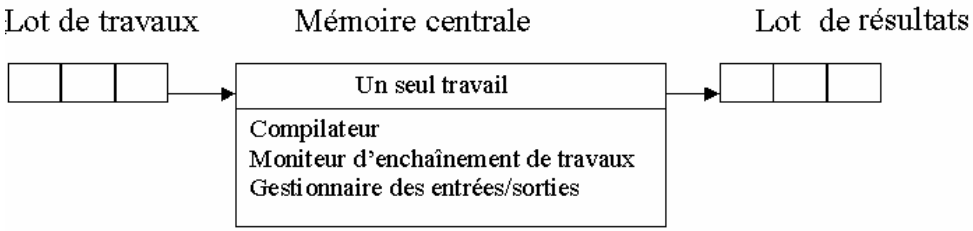
http://www-igm.univ-mlv.fr/~dr/NCS/node5.html

Traitement par lots (transistors, 1955-1965)

- Les programmes étaient écrits **en Fortran ou en assembleur** sur des cartes perforées.
- Ce mode d'exploitation nécessitait deux types de machines dont la plus puissante était réservée aux calculs et l'autre, moins chère, s'occupaient des périphériques lents.



Tannenbaum

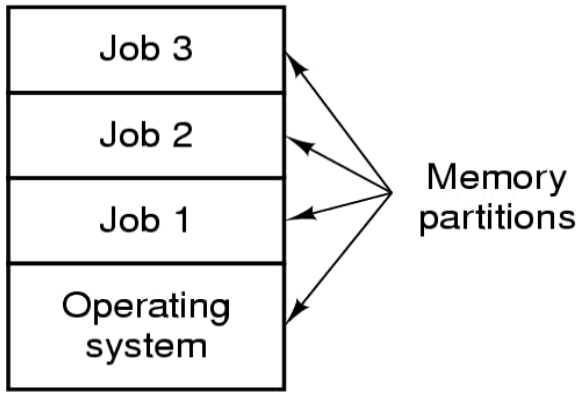


Organisation de la mémoire

Comment maximiser le taux d'utilisation du processeur ?

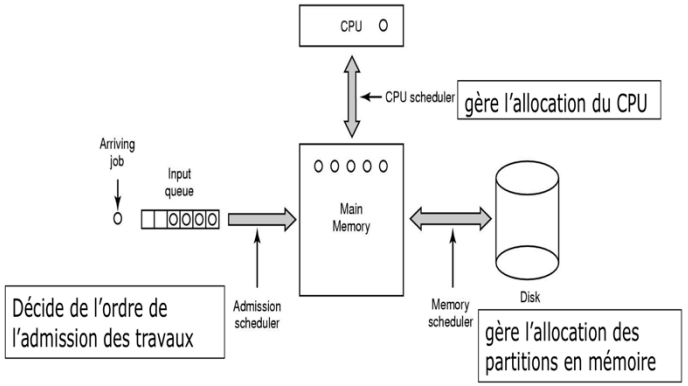
Traitement par lots + multiprogrammation (circuits intégrés, 1965-1980)

- MULTIPROGRAMMATION: Permet de conserver en mémoire plusieurs travaux et de gérer le partage du processeur et des périphériques entre-eux.
- La mémoire est organisée en un ensemble de partitions (1 travail par partition).
- L'introduction des unités de disque permettent des accès direct aux travaux.
- L'arrivée des contrôleurs DMA (Direct Memory Access) qui se chargent de gérer les E/S des périphériques, **libère le processeur de cette tâche**.



Traitement par lots + multiprogrammation (circuits intégrés, 1965-1980) (2)

- Le système alloue le processeur à un travail en mémoire (premier arrivé, premier servi).
- Lorsque le travail en cours demande une E/S, le processeur est alloué à un autre travail en mémoire.
- À la fin de l'E/S, il y a une interruption et le système d'exploitation reprend le contrôle pour traiter l'interruption et reprendre l'exécution d'un travail.
- Dès qu'un travail est terminé, le système d'exploitation charge un nouveau travail dans la partition libérée.

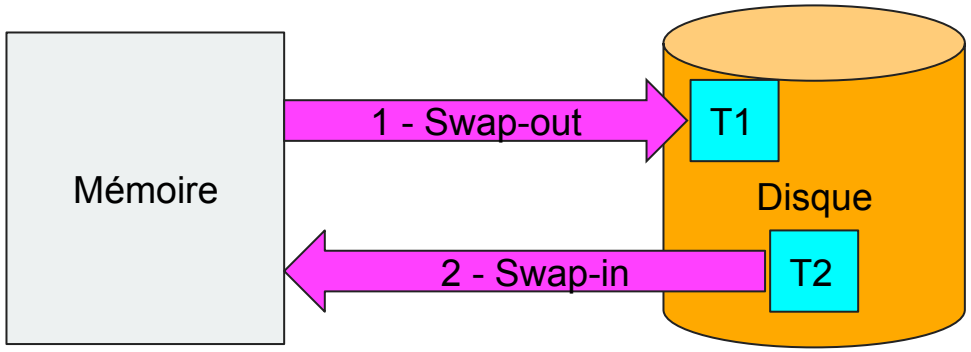


Problème : si tous les travaux en mémoire sont en attente d'E/S, le processeur est inactif.

Va-et-vient (swapping)

- L'idée est de retirer de la mémoire principale les travaux **bloqués** (en attente par exemple d'une E/S) pour les remplacer par des travaux en attente d'exécution (état **prêt**).

- Durant son exécution, un travail pourrait subir plusieurs va-et-vient entre la mémoire et le disque (**zone de swap** du disque).



Problème : Un travail peut monopoliser le processeur (une boucle de traitement infini).

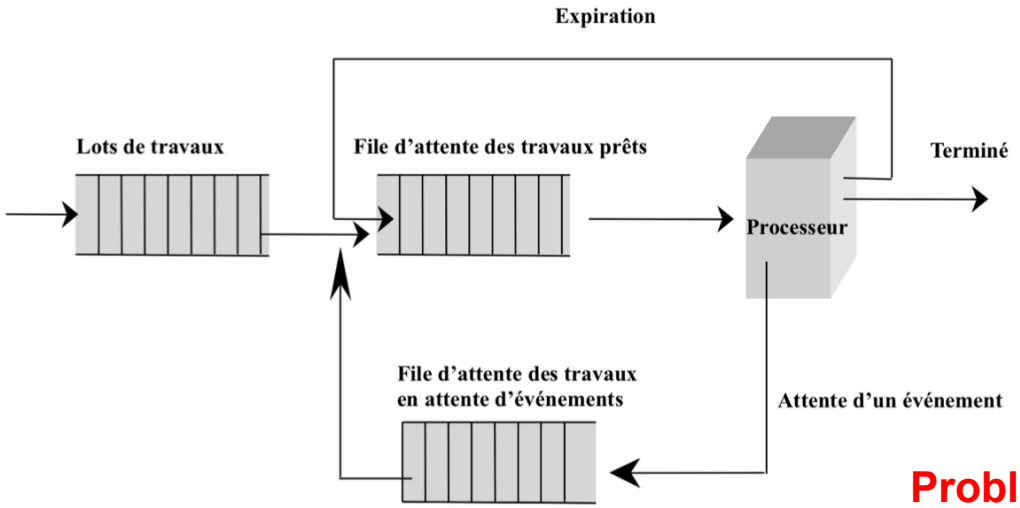
Multiprogrammation et partage de temps (1965-1980)

Concept de partage équitable de temps:

- Le processeur attribue à chaque travail un certain temps limité.
- Au bout de ce temps, si le travail n'est pas terminé, il retourne dans la file des travaux prêts
- Le CPU passe alors au traitement du travail suivant avec la même procédure.

Ce mode d'exploitation donne **l'impression** que les travaux s'exécutent en même temps (pseudo parallélisme).

Multiprogrammation et partage de temps (1965-1980) (2)

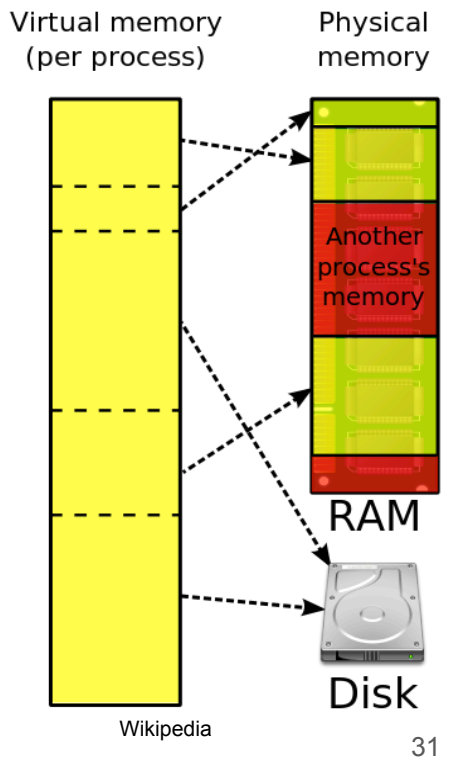


Problème : La taille de l'espace d'adressage d'un travail est limitée par celle de la mémoire physique.

Mémoire virtuelle (recherche de 1965-1975)

But: Exécuter des programmes dont la taille dépasse celle de la mémoire physique

- Chaque processus a son espace virtuel privé.
- Lors de l'exécution, une partie de cet espace virtuel est chargé en mémoire physique (chargement à la demande).
- **Adresses virtuelles versus adresses physiques**.
- Le mécanisme de translation d'adresses est intégré au processeur.



Wikipedia

http://hautrive.free.fr/reseaux/architectures/systemes-exploitation-reseaux.html

https://waytolearnx.com/2018/11/difference-entre-le-systeme-dexploitation-reseau-et-le-systeme-dexploitation-distribue.html

Exploitation en réseau, distribuée, en temps réel

- **Exploitation en réseau** (modèle **client-serveur**) : Un ensemble de machines qui communiquent via un réseau. Le système d'exploitation de chaque machine est doté d'une interface réseau.

- **Exploitation distribuée** : Un système d'exploitation réparti gère et contrôle plusieurs ordinateurs (leurs processeurs, mémoires, disques, etc.). Ce réseau d'ordinateurs apparaît aux utilisateurs comme une seule machine monoprocesseur.

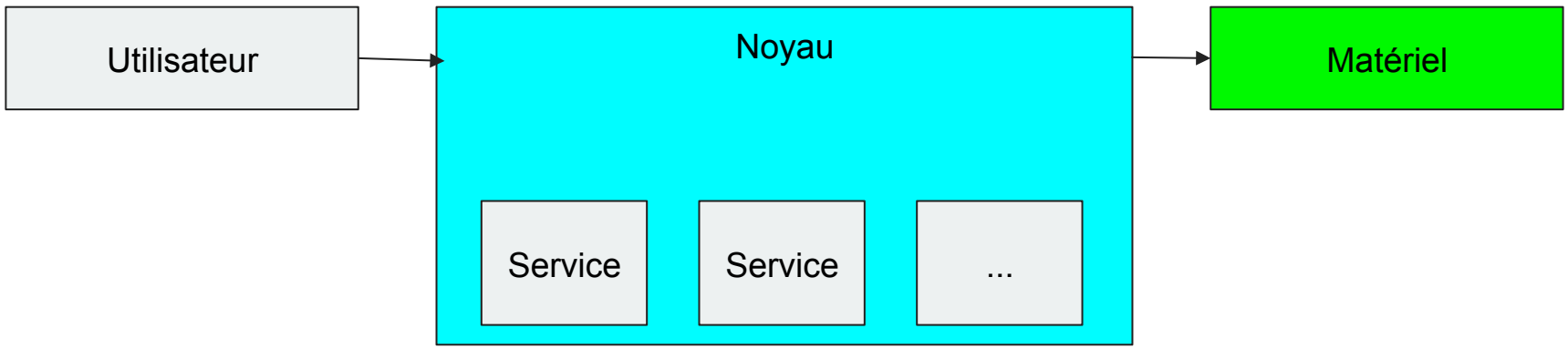
- **Exploitation en temps réel** : Un système d'exploitation temps réel met l'accent sur le temps de réponse et permet de prendre en compte les contraintes temporelles imposées par son environnement. SE temps réel sont utilisés dans des domaines où le respect du temps réel est critique (aviation, commande de processus, etc.).

INF3405 (Réseaux informatiques)
INF8601 (Systèmes informatiques parallèles)
INF4404 (Systèmes répartis et infonuagiques)
INF3610 (Systèmes embarqués)

Structure des systèmes d’exploitation

Noyau monolithique modulaire (Linux, BSD, Solaris)

- Un noyau monolithique modulaire est un **ensemble de modules** qui peuvent être **chargés dynamiquement en mémoire principale** et qui s'exécutent en mode noyau.



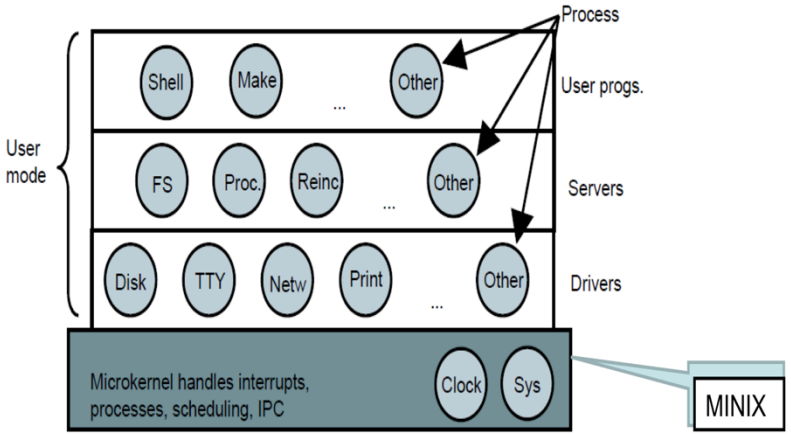
https://fr.wiktionary.org/wiki/micronoyau

https://www.geeksforgeeks.org/operating-system-microkernel/

Micronoyau (MINIX)

- Un Micronoyau contient un minimum de fonctionnalités dans le mode noyau et déplace un grand nombre de services appartenant normalement au mode noyau vers le mode utilisateur.

- Comme la plupart des services sont dans l'espace utilisateur, cela permet de facilement modifier/ajouter des services (modularité).

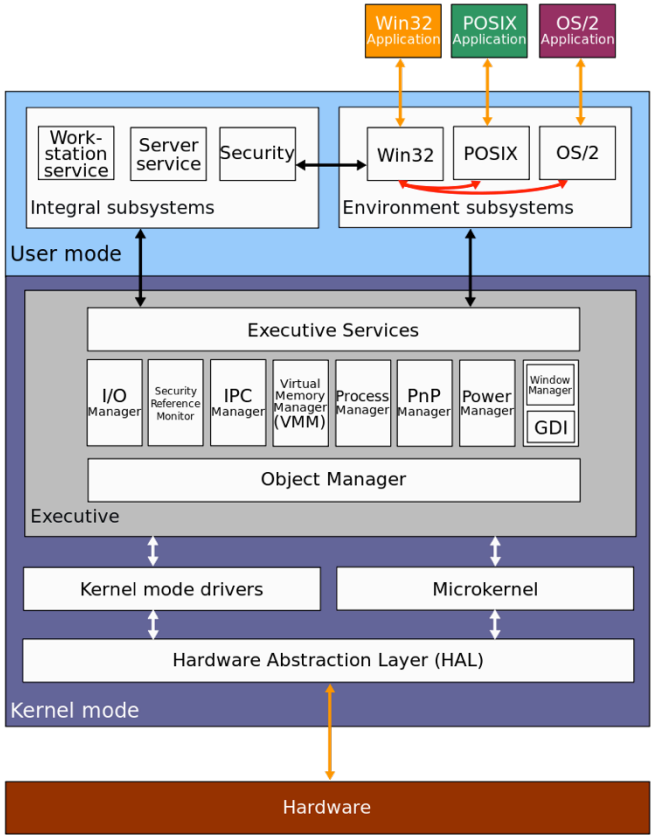


https://www.techopedia.com/definition/27004/hybrid-kernel

Noyau Hybride (Windows, Mac OS X)

- Hybride entre le Micronoyau et le noyau monolithique modulaire.
- Ce type de système possède donc l'efficacité du noyau monolithique modulaire (chargement de services dynamiquement en mémoire) et la modularité du micronoyau

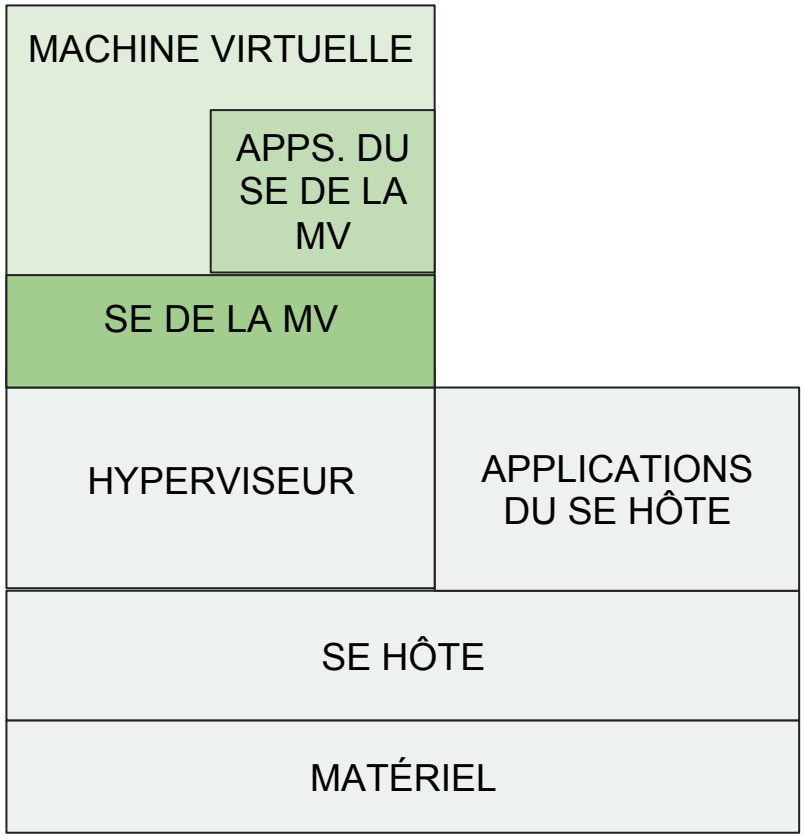
Windows NT



https://azure.microsoft.com/fr-ca/overview/what-is-a-virtual-machine/

Machines virtuelles

- Les machines virtuelles sont des fichiers (images) qui définissent un ordinateur (avec un système d'exploitation, des disques, de la mémoire, etc.)
- Un logiciel appelé **hyperviseur** permet de lancer plusieurs machines virtuelles simultanément.
- Les machines virtuelles sont exécutées dans un "bac à sable", ce qui permet de les isoler du SE Hôte



Quelques lectures

- **Chapitre 1 : Introduction**

Introduction aux systèmes d'exploitation - Cours et exercices en GNU/Linux, Hanifa Boucheneb & Juan-Manuel Torres-Moreno, 216 pages, édition ellypses, 2019, ISBN : 9782340029651.

Culture personnelle :

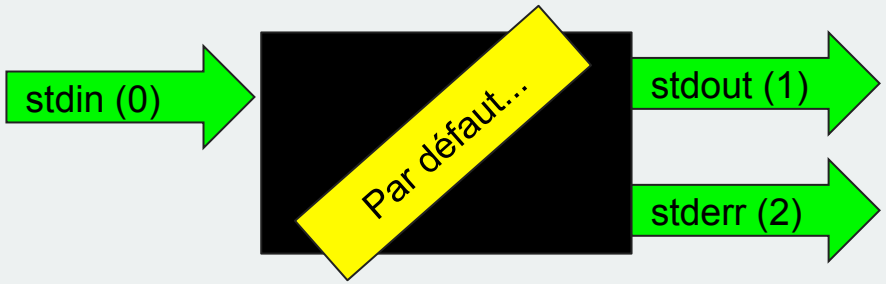
http://fr.wikipedia.org/wiki/Chronologie_informatique

https://fr.wikipedia.org/wiki/Syst%C3%A8me_d%27exploitation#Quelques_exemples

Exercice typique!

printf et write

```
$ ./run.sh
ProcessusEtThreads/
PrintfWrite
```



prinf(format, args…)
fonction prenant comme premier argument une chaîne de caractères à formater puis les valeurs à insérer dans la chaîne comme arguments subséquents. La chaîne sera placée dans un buffer ayant une taille définie **Cette chaîne sera ensuite imprimée sur la sortie standard dès que le buffer sera plein ou qu'il y ait présence d'un caractère de fin de ligne.**

exemple: écrire "bonjour 12":
printf("%s %d\n", "bonjour",12);

write(fd, string, size)
Écrit dans le fichier accessible depuis le descripteur fd la chaîne de caractères string de taille size

exemple: écrire bonjour sur stdout
write(1, "bonjour 12", 10);

Analyser les appels systèmes et de librairies

strace: permet de lister les appels systèmes faits par un programme
ltrace: permet de lister les appels de fonctions de librairies faits par un programme

```
brk(NULL)                               = 0x565557754000
brk(0x565557775000)                     = 0x565557775000
write(1, " ici 1er write \0", 16 ici 1er write )      = 16
write(1, " ici 2eme write \n", 17 ici 2eme write
)      = 17
write(1, " ici 1er printf  ici 2eme printf"..., 33 ici 1er printf  ici 2eme printf ) = 33
exit_group(0)                           = ?
+++ exited with 0 +++
```

Resultat de strace

```
printf(" ici 1er printf ")                                          = 16
write(1, " ici 1er write ", 16 ici 1er write )                                  = 16
printf(" ici 2eme printf ")                                         = 17
write(1, " ici 2eme write \n", 17 ici 2eme write
)                                                   = 17
fflush(0x7efbff4d2760 ici 1er printf  ici 2eme printf )                                                 = 0
+++ exited (status 0) +++
vittorio@lenovo-g50-45:~/Documents/Polytechnique/INF2610/ExercicesINF2610/ressources/codes/ProcessusEtThreads/PrintfWrite$
```

Resultat de ltrace