Principes des systèmes d'exploitation

Fonctions :

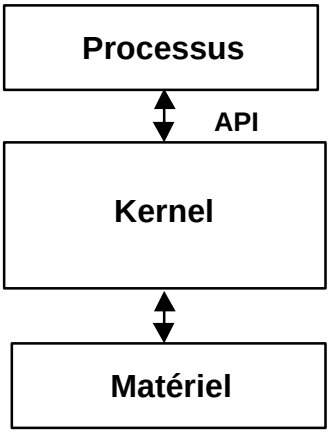
- **Machine virtuelle : ajoute des fonctionnalités (par exemple système de fichier vs accès pistes - secteurs)**
- **Gestion des ressources : processeur, mémoire, espace disque, accès au réseau...**
- **Plate-forme pour le développement d'applications**

Types :

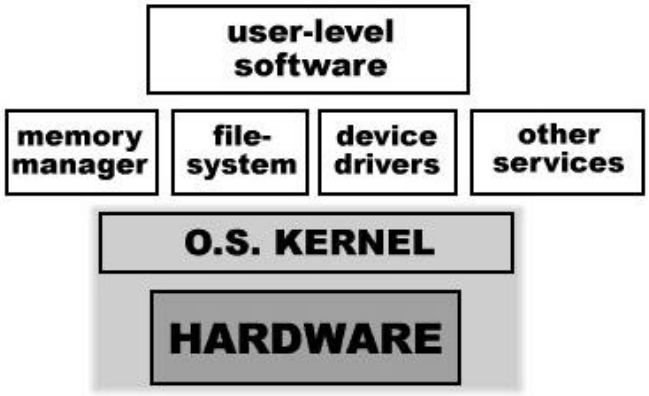
- **Multi-tâches mono utilisateur (Windows 95-98...)**
- **Multi-tâches multi- utilisateur (UNIX...)**
  - **d'usage général (sécurité)**
  - **temps réel**
- **Distribués**

Structure :

- **Monolithique**



- **Micro-kernel + modules**



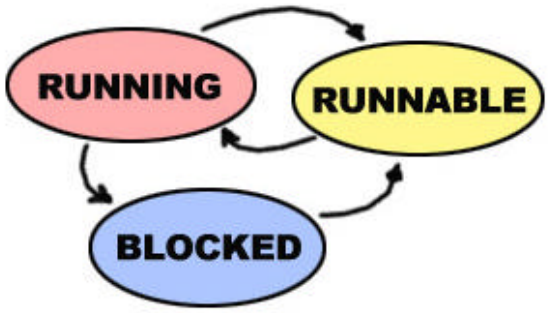
- **Le processeur doit posséder un mode protégé permettant l'exécution de certaines instructions critiques (entrées sorties physiques, manipulation des interruptions...)**
  - **kernel space : routines critiques de l'O.S.**
  - **user space : application**

Définitions :

- **Programme : lignes de code dans un fichier**
- **Processus : instance d'un programme en cours d'exécution**
- **Thread : morceau de programme en cours d'exécution**

Traitement multi-tâches :

- **Ordonnancement des tâches contrôlé par le système d'interruption :**
  - **interruption périodique (timer)**
  - **interruption liées aux entrées sorties**



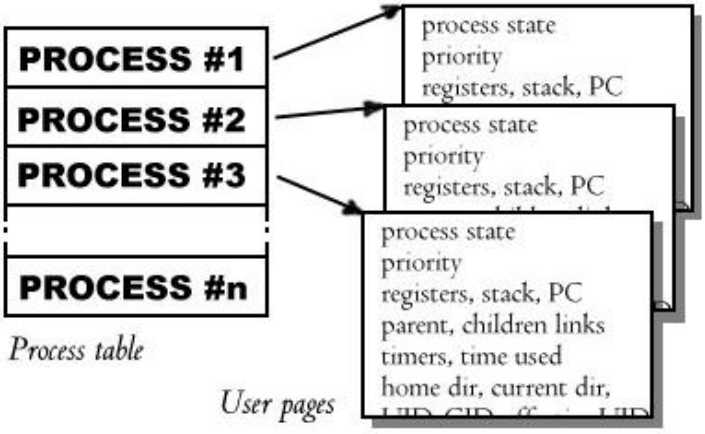
- **Problème des entrées-sorties blocantes**

```
read(int fileID, char *userBuffer, int bufferLength)
{
while (dataNotAvailable()) /* do nothing */;
// Copy bytes to process's buffer, from system buffer.
memcpy(userBuffer, device->buffer, bufferLength);
}
```

- **Mise en veille (état bloqué) :**

```
read(int fileID, char *userBuffer, int bufferLength)
{
if (dataNotAvailable())
  {
  linkProcessToWaitQueue(device->queue, thisProcess);
  thisProcess->status = BLOCKED;
  // Call the scheduler and dispatcher to give up the CPU.
  scheduler();
  }
// By the time we get here, data is available...
// ... and thisProcess->status is once again RUNNING.
// Copy bytes to process's buffer, from system buffer.
memcpy(userBuffer, device->buffer, bufferLength);
}
```

- **Liste des processus :**



Coopération entre les processus

- **Problèmes potentiels : race condition**

'add' process	'remove' process	value of 'count'
temp1 = count		10
temp1++		10
PROCESS SWITCH!		
	temp2 = count	10
	temp2 --	10
	count = temp2	9
PROCESS SWITCH!		
count = temp1		11

➔ **opération atomique**

- **Suppression des interruptions (déconseillé !)**

- **Exclusion mutuelle (MUTEX) par sémaphores : protection par des sections critiques accessibles à un seul processus à la fois**

Les sémaphores.

- **entier non négatif ne pouvant être manipulé que par trois instructions atomiques :**
  - **initialisation**
  - **WAIT : décrémente le sémaphore d'une unité s'il est > 0, sinon bloque le processus**
  - **SIGNAL : incrémente le sémaphore d'une unité**

- **Utilisation des sémaphores :**
  - **Synchronisation**
  - **Mutex**

```
// Global variables (shared between processes):
int count; // The shared variable.

// Create a semaphore for mutual exclusion---initialised to 1.
Semaphore mutex = new Semaphore(1);

void addAUser()
{
register int temp;
wait(mutex); // Start of critical section.
temp = count;
temp ++;
count = temp;
signal(mutex); // End of critical section.
}
```

- **Producteur consommateur**

```
const int BUFFER_CAPACITY = 100; // Say, for example.
Semaphore emptySlots = new Semaphore(BUFFER_CAPACITY);
Semaphore fullSlots = new Semaphore(0);
Semaphore mutualExclusion = new Semaphore(1);


void producer()
{
for(/*ever*/;;)
  {
  Object item = produceItem();
  wait(emptySlots); // Wait for buffer space.
  wait(mutualExclusion); // Enter critical section.
  buffer.insertItem(item);
  signal(mutualExclusion); // Leave critical section.
  signal(fullSlots); // Signal the consumer.
  }
}



void consumer()
{
for(/*ever*/;;)
  {
  wait(fullSlots); // Wait for item in buffer.
  wait(mutualExclusion); // Enter critical section.
  Object item = buffer.removeItem();
  signal(mutualExclusion); // Leave critical section.
  signal(emptySlots); // Signal the producer.
  consumeItem(item);
  }
}
```

- **Blocage (deadlock) - étreinte fatale (deadly embrace)**
  - **Exemple : verrouillage de fichier (file locking)**

Processus A :	Processus B :
Tant que le fichier_1 est verrouillé attendre ; verrouiller le fichier_1 Tant que le fichier_2 est verrouillé attendre ; verrouiller le fichier_2 utiliser fichier_1 et fichier _2 déverrouiller fichier_1 et fichier _2	Tant que le fichier_2 est verrouillé attendre ; verrouiller le fichier_2 Tant que le fichier_1 est verrouillé attendre ; verrouiller le fichier_1 utiliser fichier_1 et fichier _2 déverrouiller fichier_1 et fichier _2

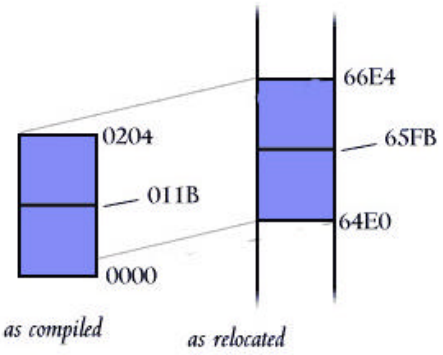
Ordonnancement des processus

- **Scheduler : choisi quel est le processus qui doit tourner**
- **Dispatcher : lance l'exécution du processus choisi par le scheduler**
- **Round robin : liste circulaire, t_{ex} temps d'éxécution, t_{ov} temps nécessaire au scheduler et au dispatcher**
  - **Efficacité : t_{ex} >>**
  - **Temps de réponse : t_{ex} <<**
- **Priorité : plusieurs listes circulaires**
- **Temps réel : temps de réponse garanti**

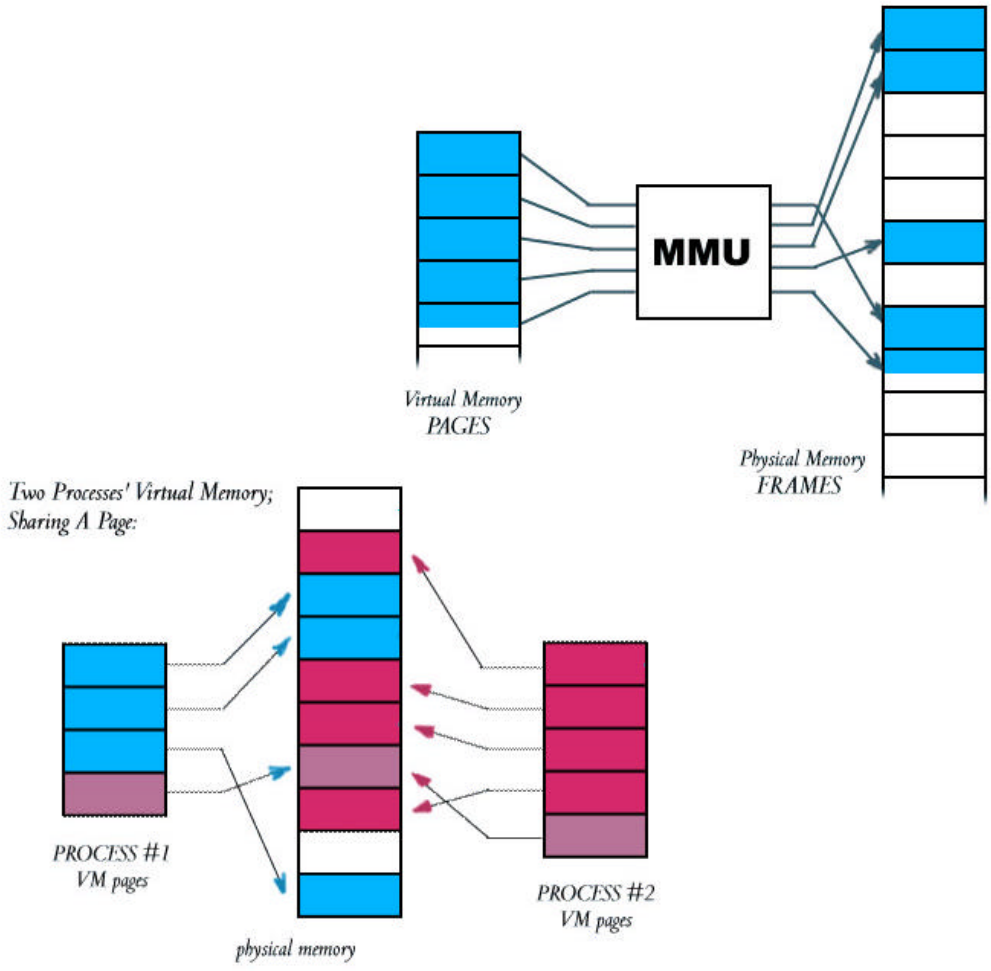
Gestion de la mémoire

- **Un processus ne doit pas pouvoir accéder à l'espace mémoire d'un autre processus (sauf mémoire partagée) -> MMU (Memory Management Unit)**

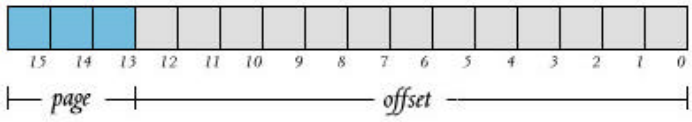
- **relocation : les adresses générées par le compilateur ne correspondent pas à la position réelle dans la mémoire**



- **Utilisation d'une MMU : relocation, protection et partage**

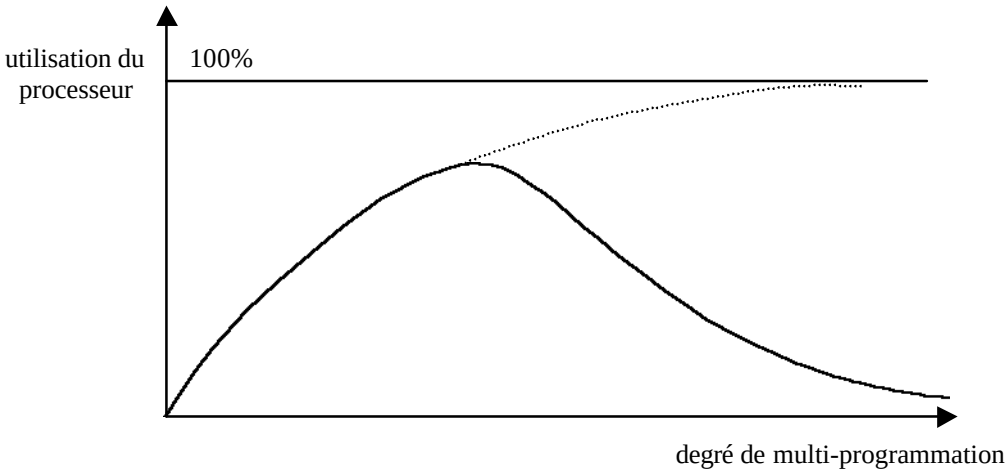


- **Taille mémoire insuffisante :**
  - **Swapping : échange de processus (et de leurs données) entre la mémoire et le disque lorsqu'il n'y a pas assez de place en mémoire**
  - **découpage de la mémoire en pages**



  - **fichier d'échange (swap file - mémoire virtuelle)**
  - **Algorithmes pour l'échange des pages entre la mémoire et le disque :**
    - **Least Rencently Used (LRU)**
    - **Not Recently Used (NRU) avec compteur d'âge**
    - **Paging daemon : maintient un certain nombre de zones libres en mémoire et regarde périodiquement s'il est nécessaire de réaliser un swap.**

- **Performances :**

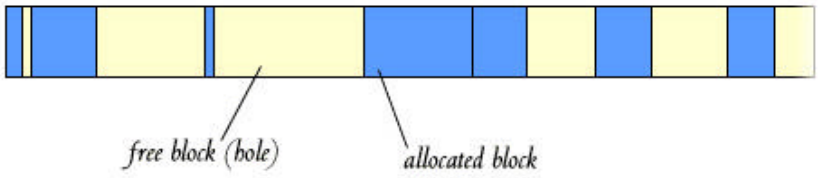


- **Un nombre minimum de page doit être présent en mémoire : sinon phénomène de thrashing (déconfiture) le processeur ne fait plus rien d'utile car le système passe son temps à faire du swap -> working set (espace vital).**

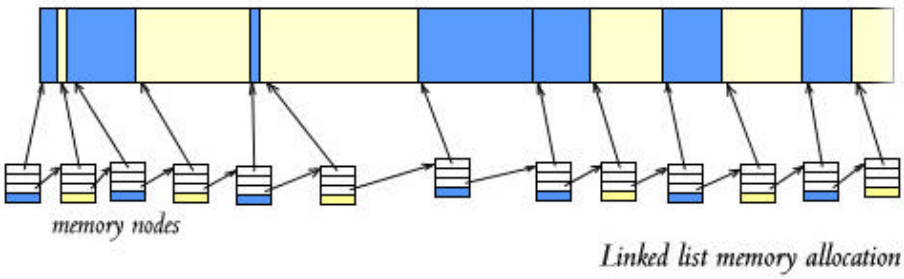
Allocation dynamique de la mémoire.

- **se fait sur le tas (heap)**

- **gestion nécessaire : liste des zones libres et occupées, fusion des petits trous, récupération des zones inutilisées (garbage collection) :**



- **liste des zones libres et allouée**



- **Recherche d'un emplacement libre de la taille adéquate :**
  - **first fit**
  - **best fit**
  - **worst fit**

La gestion des entrées-sorties.

- **Grande diversité de périphériques**
  - **vitesse**
  - **format des données**
  - **opérations possibles**
  - **conditions d'erreur**

- **Nécessité de driver spécifiques (fonctionnant en partie en mode Kernel)**
  - **deux couches de logiciel**
    - **une couche dépendante du matériel**
    - **une couche indépendante du matériel pour une famille de périphérique donnée**
    - **périphériques de type caractère (character device) : accès séquentiel, un octet à la fois (carte audio, réseau, clavier, modem...)**
    - **périphérique de type bloc (block device) : lecture-écriture aléatoire d'un bloc d'octets à la fois (disques, carte vidéo ...)**

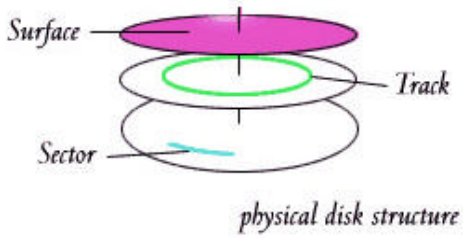
- **Référence au périphérique par un nom de "device"**
- **Intégration dans le système de fichier (tout est fichier sous UNIX)**
- **Mise en service du périphérique par une opération de montage (mount)**

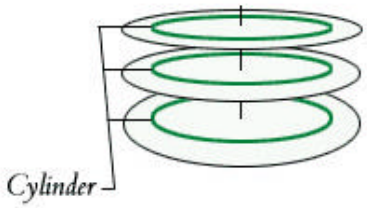
  Exemple : mount /dev/tty au démarrage

- **Les opérations d'entrées-sorties peuvent être bloquantes (synchrones) ou non bloquantes (asynchrones), lorsqu'elles sont bloquantes il y a risque de deadlock**
- **La bufferisation permet de gagner du temps (entrées et sorties) mais problème de la synchronisation**
- **Certains périphériques ne sont pas partageables (imprimante) ® spooling (Simultaneous Peripheral Output On Line)**

Le système de fichiers.

- **Organisation de manière ordonnée (hiérarchique) des informations stockées sur les disques sous forme de fichiers et de répertoires**
- **organisation logique (fichiers - répertoires)**
- **organisation physique (pistes, secteurs, formatage)**





- **Liaison entre la représentation physique et logique : table des inodes**