**Chapitre 1 : notion de processus, de coopération, de compétition et de parallélisme.**

# 1 Introduction aux S.E

Un système informatique comprend du matériel (processeur; mémoire), des programmes systèmes (Windows, Unix, Linux,…) et des programmes Applications (logiciels comme Microsoft Office Word; winrar; jeux…).

Les programmes systèmes sont appelées systèmes d`exploitation.

Un S.E est un ensemble de programmes qui coopèrent à la gestion des ressources (processeur , mémoire, périphériques) de la machine (ordinateur).

L'objectif principal d'un S.E est la gestion correcte et optimale des ressources de l`ordinateur (Mémoires, CPU, Périphériques,…..) entre les différents programmes qui y font appel. Par exemple: gestion de la mémoire, la gestion de processus, …

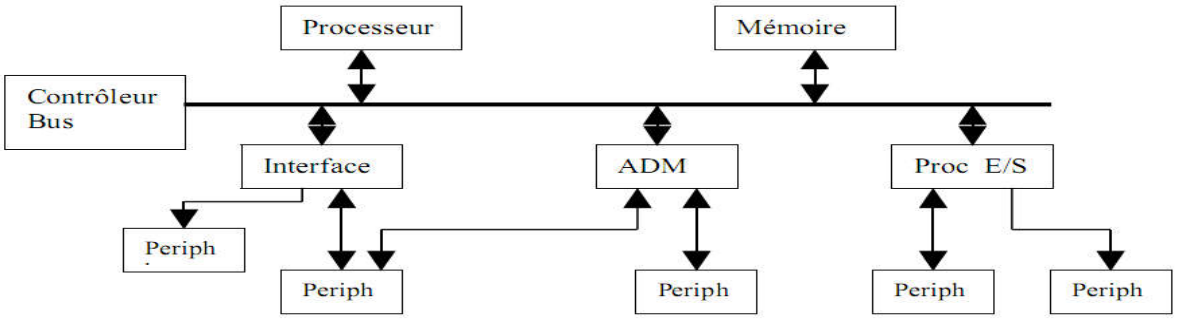



**Figure 1 - Vue générale : composants d'ordinateur.**

# 2 Notion de processus

Dans un système, plusieurs activités se déroulent simultanément, ou en séquentiel et présentent des interactions mutuelles. Ces activités résultent de l`exécution de programmes (programme system et programme utilisateur).

**2.1 Définition :** Un processus représente l'exécution d'un programme comportant des instructions et des séquences. C'est une entité dynamique (active) créée à un instant donné, qui disparaît en général au bout d'un temps fini.

Processus = instructions +ressources

**Note** : Différence entre processus et programme : le programme est une description statique ; le processus est une activité dynamique (il y a un début, un déroulement et une fin, il a un état qui évolue au cours du temps).

**2.2 Exemple de processus**

- Le calcul de la moyenne des notes

- l`impression des relevés de notes

- les commandes DOS : copy, find; del…..

## 3 Bloc de contrôle de processus

Pour leur évolution, les processus ont besoin de données (fichiers), mémoire destiné à les contenir, de l'unité centrale, de périphériques…Nous appelons toutes ces entités : Ressources.

Une ressource peut être critique ou non. Une ressource critique est celle qui ne peut être utilisé que par un seul processus à la fois.

Vu qu'un processus est décrit via une diversité d'informations. Chaque processus est représenté dans le système d'exploitation par un bloc de contrôle de processus appelé en anglais : Process Control Bloc « PCB ».

***Le PCB contient les informations suivantes :***

- L'identificateur de processus : IDP
- L'état actuel du processus
- La valeur du compteur ordinal qui indique la prochaine instruction à exécuter.
- Information sur la gestion de la mémoire (les tables de pages mémoires utilisées par le processus)
- Priorité du processus

***Table des processus :*** L'ensemble des blocs de contrôle de processus forme une table dite Table des processus. La création d'un processus revient à réserver et initialiser une entrée dans cette table.

## 4 Opérations sur les processus

### *a. Création de processus*

Pendant son exécution, un processus peut créer plusieurs nouveaux processus par l'intermédiaire d'un appel système de création de processus. Le processus effectuant la création s'appelle père et le processus créé est dit fils.
Le processus père peut répartir ses ressources entre plusieurs de ses fils. Il peut transférer des données d'initialisation au fils.

### b. Terminaison de processus

La terminaison d'un processus peut avoir lieu de deux façons :

- Terminaison normale : à la fin de l'exécution de sa dernière instruction.

- Terminaison anormale : si un mauvais fonctionnement est détecté ou que le processus n'est pas utile. Une commande spéciale est utilisée à cette fin : **KILL nom-processus**

La terminaison d'un processus entraîne :

- la destruction de sa descendance
- La libération de l'entrée correspondante dans la table des processus
- La libération des ressources occupées par le processus.

### C. Processus dans Unix :

- Un processus est identifié par un numéro PID,

- La commande ps donne la liste des processus en cours d'exécution,

- La commande fork() crée un nouveau processus,

- getpid() : obtenir le numéro du processus courant.

- getuid() : obtenir le numéro d'usager auquel appartient le processus,

- kill(PID) : tuer un processus spécifié par PID.

- sleep(n) : se bloquer pendant n secondes.

- exit() : terminaison d'un processus.

- wait() : mise en sommeil sur la terminaison d'un fils.

**NB** : Lors du démarrage de système Unix, deux processus sont crées :

- Swapper(PID = 0) qui gère la mémoire,
- Init(PID = 1) qui crée tous les autres processus.

## 5 Etats d`un processus

Chaque processus évolue par l`exécution de ses instructions respectives en utilisant un ensemble de ressources telles que : la mémoire, processeur, imprimante, fichier de données….

L`évolution des processus passe par différents états allant de la naissance à la destruction au bout d`un temps fini (Figure 2).

- **Etat Prêt**.
  si le processus dispose de toutes les ressources nécessaires à son exécution à l'exception du processeur.

- **Etat Elu** (en Exécution). Processus en cours d`exécution sur le processeur

- **Etat Attente**. Si le processus ne dispose pas du processeur et d`autres ressources ou de données nécessaires à son exécution.

La création et la destruction d`un processus est souvent faite par d`autres processus qui déclenchent ces opérations en conséquence d`une commande directe ou indirecte de l`utilisateur.

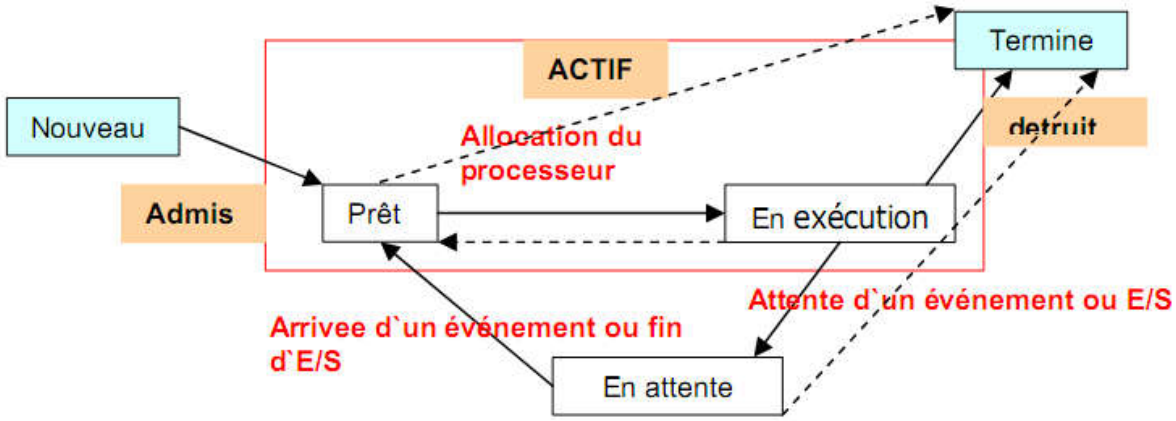



***Figure 2.*** *Diagramme d`état d`un processus*

## 6 Relation entre processus Compétition/ Coopération :

Dans un système, plusieurs processus peuvent se dérouler simultanément. Ces processus résultent de l'exécution de programmes. Durant leur évolution, les processus d'un système interagissent les uns avec les autres. Selon que les processus se connaissent mutuellement ou pas, ces interactions en deux types qui sont : Coopération / Compétition.

a) **Compétition** : C'est la situation dans laquelle plusieurs processus doivent utiliser simultanément une ressource critiques critique à accès exclusif (ressource ne peut être utilisée que par un seul processus à la fois ).

**Exemples :** Imprimante, variable globale.

Une solution possible (mais non la seule) pour gérer la compétition : faire attendre les processus demandeurs jusqu'à ce que l'occupant actuel ait fini (premier arriver, premier servi). Fifo

b) **Coopération :** Si les processus se connaissent mutuellement, on parle alors de coopération. C'est la situation dans laquelle plusieurs processus collaborent à une tâche commune et doivent **synchroniser** pour réaliser cette tâche. Un processus est dit coopératif s'il peut affecter les autres processus en cours d'exécution dans le système ou être affecté par exécution.

***Exemples :***

- P1 produit un fichier, p2 imprime le fichier,
- P1 met à jour un fichier, p2 consulte le fichier,

Note : La synchronisation se ramène au cas suivant : un processus doit attendre qu'un autre processus ait franchi un certain point de son exécution.

**Remarque** : Notons que, pour les deux types de relations (compétition ou coopération), on doit faire attendre un processus. Comment réaliser cette attente ?

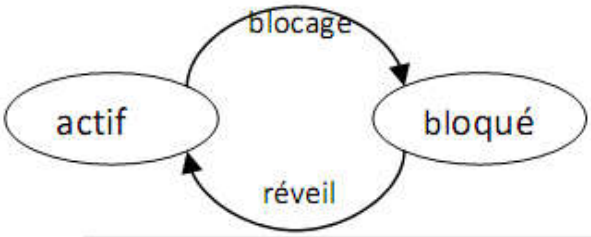
- **Solution 1 : attente active**

| P1 | P2 |
|---|---|
| `while (resource occupée)`<br>`{}`<br>`ressource occupée = true;` | `ressource occupée = true;`<br>`utiliser resource;`<br>`ressource occupée = false ;` |

**Inconvénient :** l'attente active monopolise le processeur et donc dégrade la performance globale de la machine.

- **Solution 2 : blocage du processus**

On définit un nouvel état pour les processus, l'état bloqué. L'exécution d'un processus bloqué est arrêtée, jusqu'à son réveil explicite par un autre processus ou par le système.




```
…
 sleep(5);/* se bloquer pendant 5 secondes */
…
```

## 7 Notion de parallélisme

Soit le programme suivant :

I1: lire(a)

I2: lire(b)

I3: c=a+b

I4: ecrire(c)

L'instruction I2 et I1 peuvent s'exécuter en même temps; on dit alors que il y a un parallélisme entre eux;

L'instruction I4 n'est peut s'exécuter avant les autres instructions car il y une contrainte de dépendance entre eux.

Des langages de programmation ont été mis en ouvre pour spécifier deux instructions ou bloc d'instruction parallèles

**Exemple :** Dans langage Pascal, l'instruction PARBEGIN et PAREND permettant de délimiter un bloc d'instructions en parallèles

```
PARBEGIN
    lire(a)
    lire(b)
PAREND
    c=a+b
Écrire(c)
```