Centre Universitaire de Mila

2 ème année licence LMD Informatique

Module : Systèmes d'exploitation 1

Bessouf Hakim

CHAPITRE 4:

Gestion du processeur central

1. Définition du Scheduling / Scheduleur
2. Objectifs du scheduling
3. Critères de scheduling
4. Niveaux de scheduling (scheduling des Jobs, scheduling des processus)
5. Politiques de scheduling
6. Contrôle de processus (état d'un processus, Bloc de contrôle de processus) PCB, Création de processus, destruction …)

Introduction

- L’espace mémoire alloué à un processus est appelé **image mémoire** (**Memory Map**) du processus , Il est divisé en plusieurs parties :

Le Code : C'est le code des instructions du programme à exécuter. L’accès à cette zone
mémoire se fait en **lecture seulement** (**ReadOnly**).

Les Données (Data) : Il contient l’ensemble des constantes et variables déclarées**.**

La Pile (Stack) : Elle permet de stocker: les valeurs des registres, Variables locales, paramètres de fonctions et les adresses de retour des fonctions.

Le Tas (Heap) : C'est une zone à partir de laquelle l’espace peut être alloué dynamiquement **en cours d’exécution du processus**, en utilisant par exemple les fonctions **new** et **malloc**

Introduction

- Pour gérer les processus le système d'exploitation utilise une table en mémoire centrale appelé **table des processus** ou **bloc de contrôle des processus** (PCB). Cette table contient plusieurs informations concernant chaque processus (conteurs ordinal, mots d'état, pointeur de pile, ressources utilisés ..etc)

PCB

Identité : - Nom du processus
Gestion du processus : - Compteur ordinal - mot d'état - Registres - Temps de traitement utilisé - processus fils - … etc
Gestion de la mémoire : - Limites mémoire - ..etc
Gestion des fichier : - Répertoire racine - Fichiers utilisés - ..etc

Introduction

- **Manipulation des processus**

Le système d'exploitation utilise plusieurs primitives (procédures) pour manipuler les processus :

Création : utilisé par un processus pour créer un autre processus, Le premier processus est appelé **processus père**, le deuxième processus est appelé p**rocessus fils**. Un processus père peut avoir plusieurs processus fils.

Activation : Elle permet de mettre un processus prêt dans l'état actif,

Suspension : Elle permet de suspendre un processus,

Destruction : Elle permet de terminer un processus et de libérer toutes les ressources qu'il utilise (mémoires, fichiers ..etc).

l'ordonnanceur des processus (scheduler)

- L'ordonnanceur est un module du système d'exploitation qui **partage le processeur** central (CPU) entre plusieurs processus en attente d'exécution (dans l'état prêt), suivant une **politique** définit a l'avance par les concepteurs du système d'exploitation.
- L'ordonnanceur donc est un programme qui **choisit le prochain processus a exécuter**.
- Le choix du processus a exécuter est appelé **ordonnancement** (**scheduling**).

Objectifs de l'ordonnanceur (scheduler)

- Les principaux objectifs de l'ordonnanceur sont :
- **L'équité** : l'ordonnanceur doit allouer le processeur central aux processus de même priorité de manière **juste** et **équitable**.
- **Utilisation maximal des ressources** :
  l'ordonnanceur doit assurer une utilisation maximal des ressources de la machine (E/S, mémoire, ..etc)

Objectifs de l'ordonnanceur (scheduler)

- Dans **les systèmes par lots :**
  ➢ L'ordonnanceur doit exécuter un maximum de travaux par heur,
  ➢ L'ordonnanceur doit réduire le délais entre la soumission du travail et la sortie des résultats,
  ➢ L'ordonnanceur doit utiliser au maximum le processeur central,
- Dans **les systèmes interactifs** (il y a interaction entre l'utilisateur et la machine) **:**
  ➢ L'ordonnanceur doit minimisé le temps de réponse, il doit donc répondre rapidement aux requêtes des utilisateurs,
- Dans **les systèmes temps réel** (il y a des contraintes temporelles comme dans centrales nucléaire, usines ..etc) :
  ➢ L'ordonnanceur doit respecter les délais.

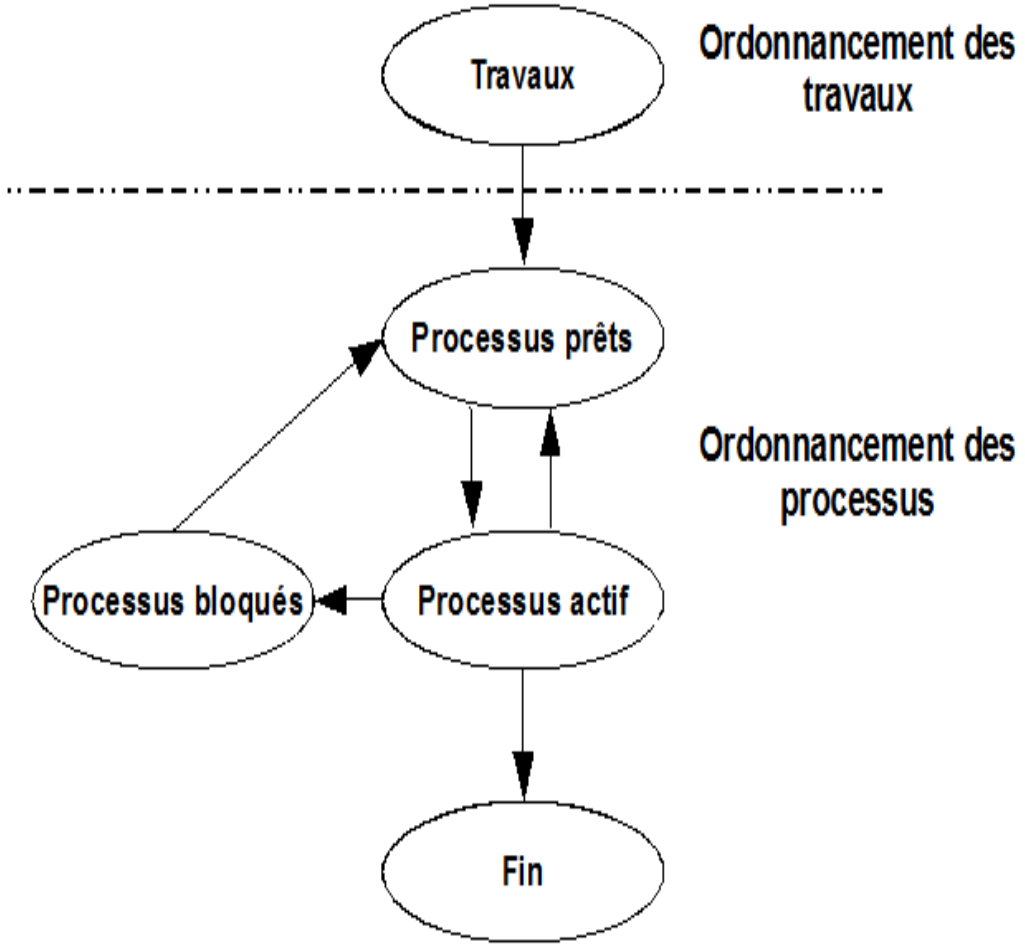
Les critères d'ordonnancement (scheduling)

- **Disponibilité des ressources: l**es programmes sont ordonnancés suivant les ressources qu'il demandes, et suivant les ressource disponible dans la machine.
- **Classe des programmes: l**es programmes sont ordonnancés suivant leurs type (interactif, temps réel, orienté calcul, orienté E/S ..etc).
- **Ordonnancement avec ou sans préemption:** Dans les politique d'ordonnancement **sans préemption** un programme s'exécute jusqu'à ce qu'il se bloque (pour attendre une E/S), ou il se termine. Par contre dans les politiques d'ordonnancement **avec préemption** l'ordonnanceur peut suspendre un programme en exécution et exécuter un autre programme.
- **Ordonnancement avec ou sans priorité:** Les programmes peuvent ou non avoir des priorités d'exécution. Par exemple un programme système peut être prioritaire qu'un programme utilisateur.

Niveaux d'ordonnancement

Il existe deux nivaux d'ordonnancement :

- L'**ordonnancement des travaux** permet de choisir le travail a admettre dans le système et créer le processus correspondant.
- L'**ordonnancement des processus** permet de choisit le processus a exécuter.



1. Politiques d'ordonnancement des travaux

1.1 Politique d'ordonnancement sans préemption

La politique du premier arrivé, premier servie (FCFC : First Come First Served, FIFO : First In First Out) : Dans cette politique le travail qui arrive le premier est servie le premier. Cette politique *désavantage les processus courts*.

La politique du travail le plus court d'abord (SJF : short job first): Dans cette politique le travail le plus court est servie le premier. Cette politiques *favorise les travaux cours* mais elle nécessite de *connaître a l'avance le temps d'exécution* des travaux.

L**a politique d'ordonnancement par priorités:** Dans cette politique l'ordonnanceur exécute le travail qui a la plus grade priorité. La priorité des programme dans ce cas est *fixé a l'avance*. l'ordonnancement par priorité peur être préemptif ou non préemptif.

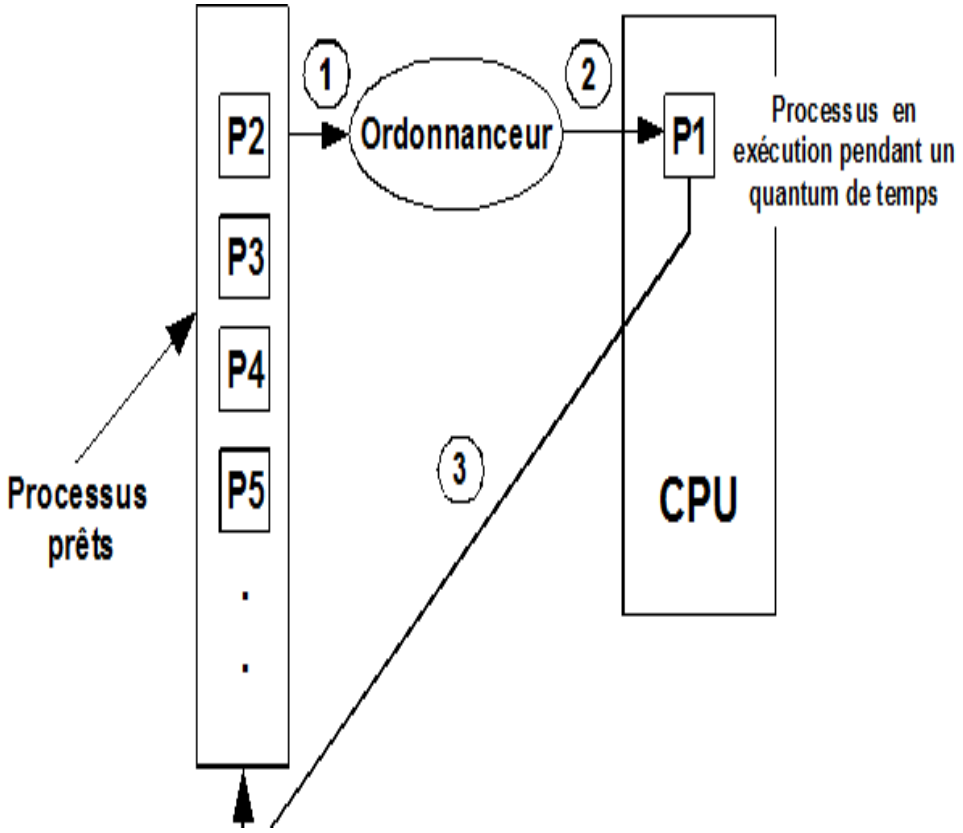
1. Politiques d'ordonnancement des travaux

1.2 Politique d'ordonnancement avec préemption
La politique du travail avec le plus court temps restant (SRTF : Shortest Remained Time First) : l'ordonnanceur choisit le travail dont le temps restant d'exécution est le plus court. C'est une version préemptive de SJF. Si par exemple un premier travail est en exécution et un nouveau travail arrive, si le temps d'exécution du deuxième travail est inférieur au temps restant au premier travail, le premier travail est arrêté et le deuxième travail est exécuté.

Cette politiques *favorise les travaux cours* mais elle nécessite de *connaître a l'avance le temps d'exécution* des travaux

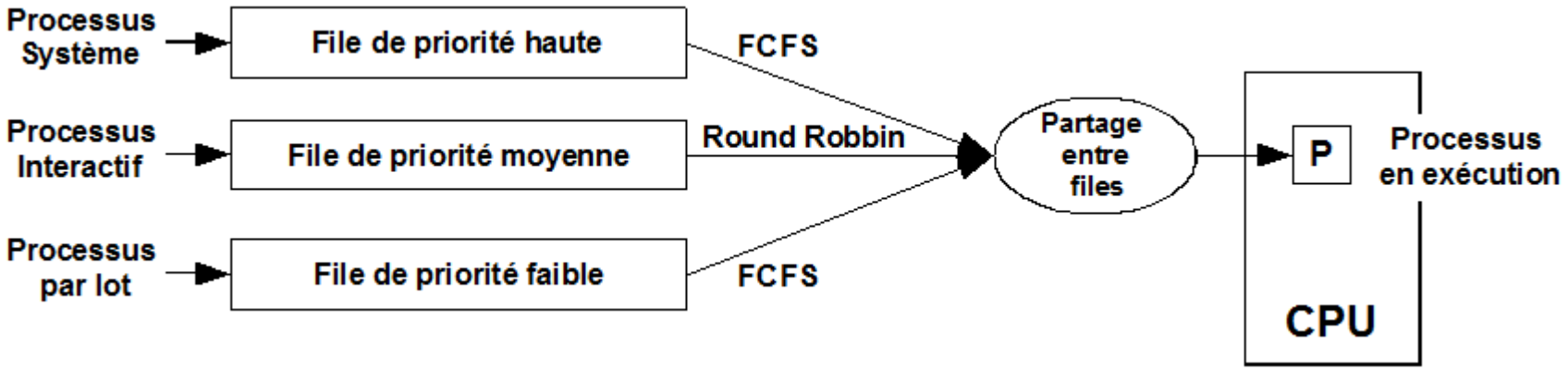
Politiques d'ordonnancement des processus

- **La politique du tourniquet (Round Robbin)** Elle est utilisé dans les systèmes a temps partagé, elle est similaire a la politique FCFS mais on rajoute la réquisition. Dans cette politique chaque processus prêt est exécuté pendant une tranche de temps appelé **quantum**, a la fin de ce temps l'ordonnanceur exécute un autre processus.
- Si le quantum est très grand cette politique ressemblera a la politique FCFS, par contre si le quantum est très petit chaque processus aura l'impression de posséder son propre processeur mais on aura une surcharge due a la **commutation de contexte** des processus.



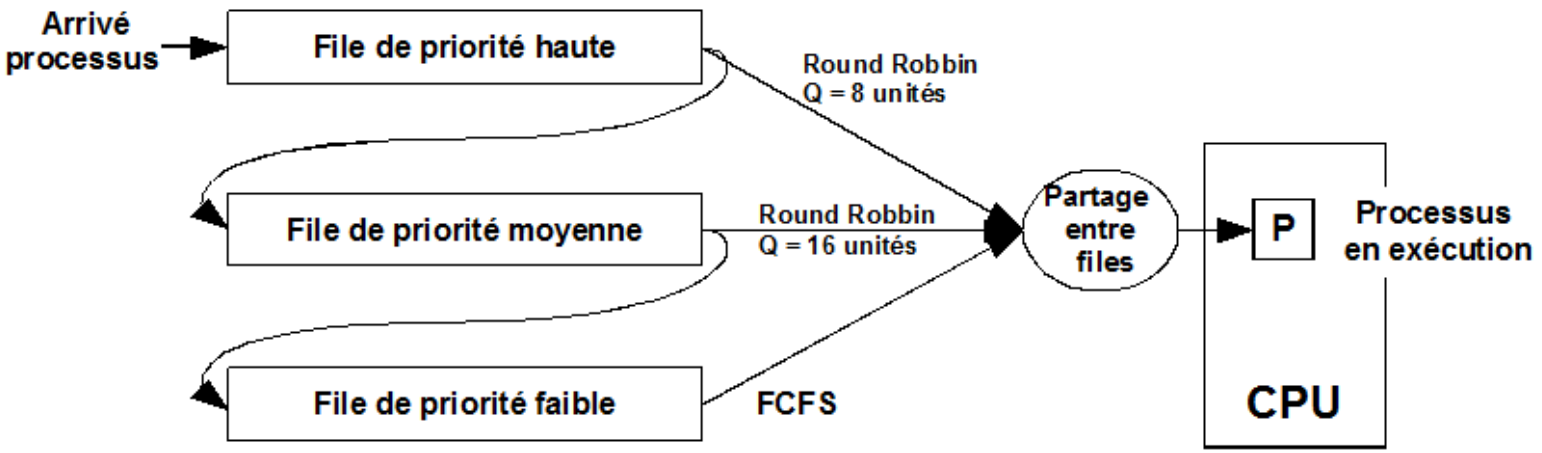
Politiques d'ordonnancement des processus

- **La politique à plusieurs niveaux de queues**
  Dans cette politique les processus sont regroupés en plusieurs fils selon leurs type (interactif, temps réel, ..etc) . Chaque file est ensuite géré par une politique d'ordonnancement qui s'adapte le mieux a cette file. Une autre politique est utilisé pour ordonnancer les files entre eux. Un exemple de cette politique est donné dans la figure suivante :



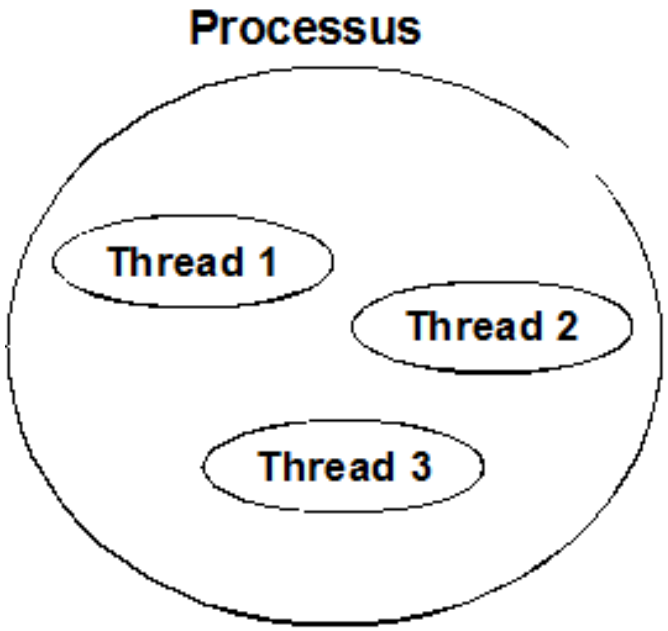
Politiques d'ordonnancement des processus

- **La politique à plusieurs niveaux de queues dépendantes**
  Dans cette politique un processus peut changer de file selon son comportement durant son exécution.



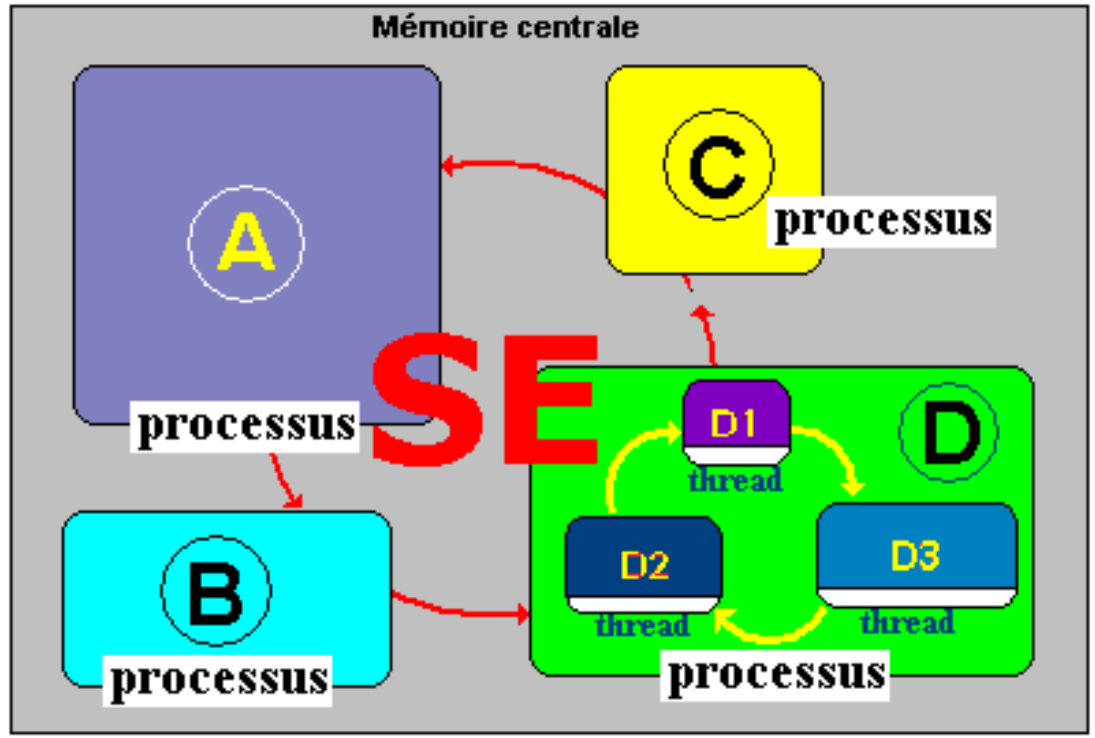
Les Threads

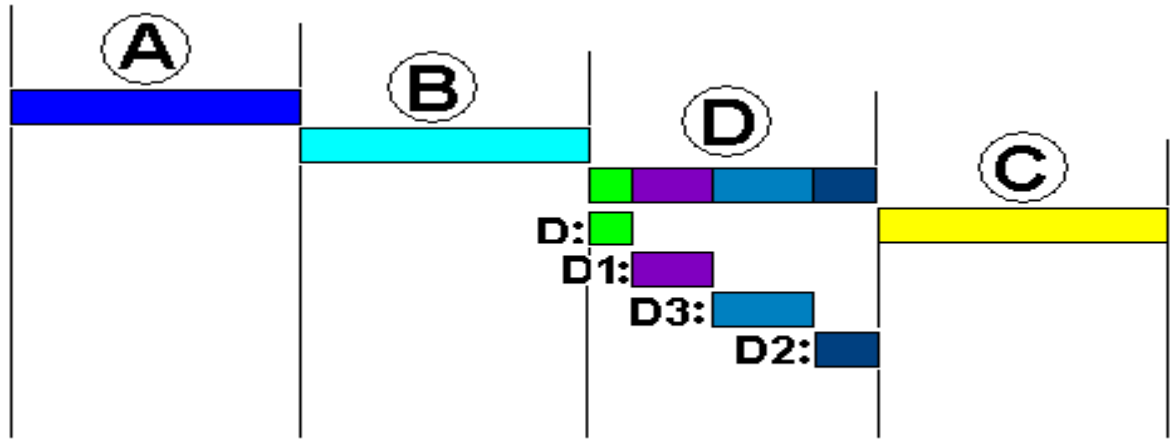
- plusieurs petits processus dans un même processus qui manipulent les données et calcules les résultats. Ces petits processus sont appelés **threads**. Par exemple un premier thread peut lire les données, en même temps un deuxième thread calcule les résultats et en même temps un troisième thread affiche ces résultats. Ces trois threads partagent le même espace d'adressage



Les Threads

- Les threads peuvent partager le même espace d'adressage et manipuler les mêmes données,
- La création et destruction des threads est très rapide par rapport a la création et a la destruction des processus,
- L'utilisation des threads permet a une application d'exécuter plusieurs tâches en même temps ce qui accélère l'exécution de l'application.
- L'utilisation des threads est très intéressante quand la machine est équipé de plusieurs processeurs chaque thread peut s'exécuter sur un processeur différent..





Tranches de temps allouées pendant l'exécution