Chapitre 2

Gestion du Processeur

Université El-Chahid Hamma Lakhdar
Module : SE1
2ème Année LMD Informatique
Enseignant : BERDJOUH Chafik

Plan

- Introduction
- Définitions
  - Notion de Programme.
  - Notion de Processus.
  - Notion de Thread.
  - Notion de ressource
  - Notion de travail (Job)
- Différents états d'un processus.
- Hiérarchies de processus.
- Relations entre processus (compétition, coopération et synchronisation).
- Techniques d'ordonnancement de processus :
  - Critères (Equité, efficacité, temps de réponse, temps d'exécution, rendement)
- Algorithmes d'ordonnancement (parmi les plus utilisés) :
  - Algorithme du premier entré, premier servi ou FCFS (First Come First-Served).
  - Algorithme du travail le plus court d'abord ou SJF (Shortest Job First).
  - Algorithme du temps restant le plus court ou SRT (Shortest Remaining Time).
  - Tourniquet (Round Robin RR).
  - Algorithme avec priorité.

INTRODUCTION

Nous avons vu précédemment que, pour améliorer la rentabilité des machines, il est nécessaire de pouvoir exécuter des activités parallèles, comme exécuter un transfert d'E/S en même temps qu'un calcul interne au processeur. A cet effet, sont apparus les systèmes multitâches (exécutant plusieurs tâches à la fois), et les systèmes multiutilisateurs (plusieurs utilisateurs travaillent en même temps). Dans ce cas, le système doit gérer plusieurs programmes (processus), et les exécuter dans les meilleurs délais possibles en partageant le temps processeur.

De plus, il doit donner à chaque utilisateur l'impression de disposer du processeur à lui seul. Pour cela, un programme du SE s'occupe de gérer l'utilisation ou plutôt l'allocation du processeur aux différents programmes : c'est le Scheduler (ordonnanceur). La gestion du processeur central se base principalement sur l'organisation ou la stratégie qui permet l'allocation du processeur aux différents programmes qui existent dans la machine.

II.1 Définitions

Notion de Programme.

Un programme est une suite d'instructions permettant de réaliser un traitement. Il revêt un caractère statique.

II.1 Définitions

Notion de processus

- Un processus est un programme en état d'exécution dans la mémoire. Il décrit par son contexte.(objet dynamique : programme + contexte)
- Un processus est l'image en mémoire centrale d'un programme s'exécutant avec son contexte d'exécution.
- Contexte :
  - Un Contexte du processeur (CPU) : l'ensemble des registres (CO, SW, …., …)
  - Un Contexte de la Mémoire Centrale : segments de code, segments de données, …

☛ Ne pas confondre processus et programme

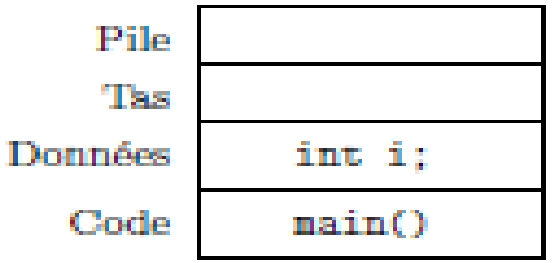
- Programme :

Code + données (passif)

```
int i;
int main() {
  printf("Salut\n");
}
```

- Processus :

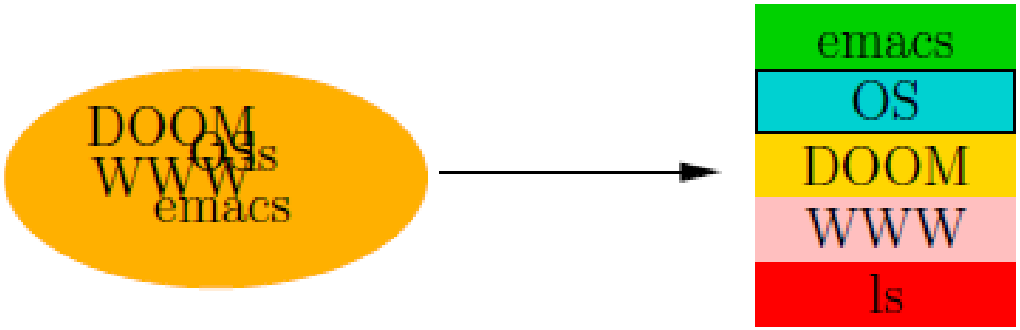
Programme en cours d'exécution



- Vous pouvez utiliser le même programme que moi, mais ça ne sera pas le même processus que moi
- Même différence qu'entre classe d'objet et instance d'objet

☛Utilité des processus : simplicité

- L'ordinateur a des activités différentes
- Comment les faire cohabiter simplement ?
- En plaçant chaque activité dans un processus isolé



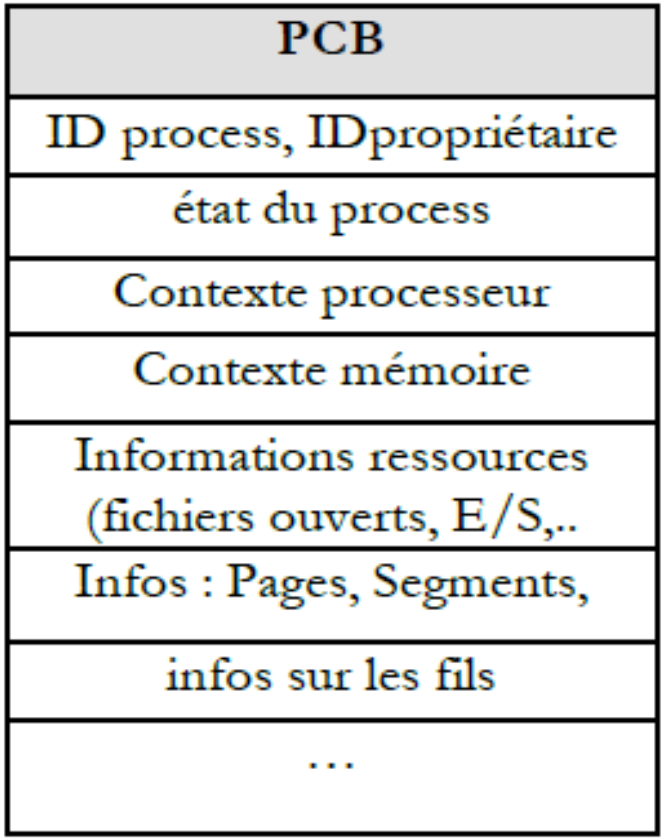
- La décomposition est une réponse classique à la complexité.

Informations sur un Processus

- Identificateur du processus (PID)
  - numéro unique
- Informations de généalogie
  - processus parent (PPID)
- Information de droits
  - utilisateur propriétaire du processus
- Information sur la mémoire utilisée
- Information sur le temps passé
- Information sur les fichiers ouverts (ressources)

Représentation interne des processus

- Au moment du chargement d'un programme exécutable, le SE lui crée une structure représentant le processus associé. Cette structure devrait contenir toutes les informations nécessaires permettant l'évolution dynamique du processus au sein du SE, entre autre son contexte, son état, …cette structure est appelée le Bloc de Contrôle du Process (PCB).
- Pour manipuler tous les processus, le SE détient d'une **table de processus** dont chaque entrée contient un pointeur vers un PCB d'un processus.



Bloc de contrôle du processus

II.1 Définitions

Notion de thread (fil d'exécution, tâche, sous-processus)

Lorsqu'un processus présente un code un petit peu long ou compliqué, il peut être décortiqué en un ensemble d'unités de traitements élémentaires ayant une cohérence logique dont la fonction est bien déterminée. Cette unité est appelée une **tâche.**

- Un thread est une subdivision d`un processus
- Les différents threads d'un processus partagent l'espace adressable et les ressources d'un processus
- Les threads sont des processus légers exécutés «à l'intérieur» d'un processus
- L'exécution des threads est concurrente
- il existe toujours au moins un thread : le thread principal
- La durée de vie d'un thread ne peut pas dépasser celle du processus qui l'a crée

Multi-threading

Une application multi-thread est un logiciel qui dès sa conception a été partagé en différentes sous-applications appelées threads.

A l'inverse de ce qui passe dans le cadre du multi-tâche où chaque processus dispose d'une partie distincte de la mémoire pour s'exécuter indépendamment des autres. Les threads issus de même application partagent un même espace.

Tel que : Word contient des sous applications (thread) : Un pour l'intercation avec l'utilisateur, un pour la sauvegarde périodique du document, un autre pour la vérification de grammaire … etc.

II.1 Définitions

Notion de Ressources

– Une ressource désigne toute entité dont a besoin un processus pour s'exécuter.

- Ressource matérielle (processeur, périphérique)
- Ressource logicielle (fichier, variable).

– Une ressource est caractérisée :

- par un état : libre / occupée
- par son nombre de points d'accès (nombre de processus pouvant l'utiliser en même temps)

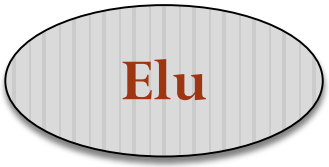
- Ressource critique – ressource ne pouvant être utilisée que par un seul processus à la fois . Ex. : Processeur, imprimante

- Un processus utilise une ressource comme suit :

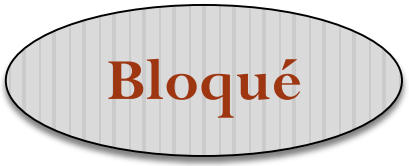
- Allocation de la ressource
- Utilisation
- Restitution de la ressource

II.2 Différents états d'un processus

- Au fur et a mesure qu'un processus exécute, il est caractérisé par un état
  - Lorsque le processus obtient le processeur et s'exécute, il est dans l'état **élu**. L'état **élu** est l'état d'exécution du processus



  - Lors de l'exécution, le processus peut demander à accéder à une ressource. Il quitte alors le processeur et passe dans l'état **bloqué**. L'état **bloqué** est l'état d'attente d'une ressource autre que le processeur.



II.2 Différents états d'un processus.

- Lorsque le processus est passé dans l'état bloqué, le processeur a été alloué à un autre processus. Le processeur n'est donc pas forcément libre. Le processus passe dans l'état **prêt**. L'état **prêt** est l'état d'attente du processeur.

II.2 Différents états d'un processus.

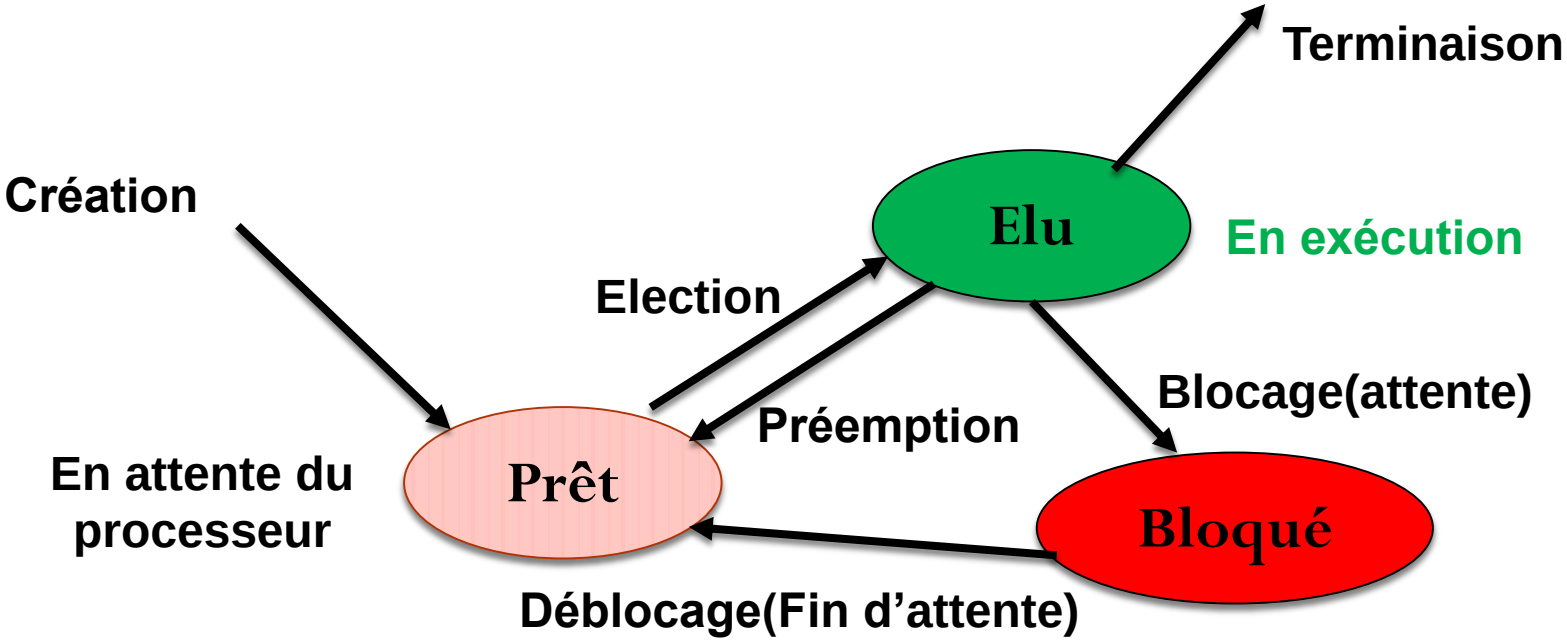
Diagramme d'états d'un processus

Transition Prêt →Élu : opération d'élection

Transition Élu→Bloqué : opération de blocage (attente E/S …)

Transition Bloqué→Prêt : opération de déblocage

Transition Elu→Prêt : réquisition du processeur, préemption (fin de quantum dans le mode temps partagé).



II.3 Hiérarchie de processus

- Les SE qui font appel au concept de processus doivent permettre de créer et détruire dynamiquement les processus
- Un processus peut créer un autre processus, et le processus enfant peut lui même créer plusieurs processus, formant un **hiérarchie de processus**.
- Un processus a un seul parent et peut avoir 0 ou plusieurs fils.
- Un processus et l'ensemble de ses descendants sont appelés un groupe de processus
- Linux/UNIX:
  - Si le processus A crée le processus B, A est le parent de B, B est le fils de A (A par défaut, exécute le même code que B) B peut à son tour créer des processus. Un processus avec tous ses descendants forment un groupe de processus représenté par un arbre de processus.
  - fork est le seul appel système de création de processus.

II.3 Hiérarchie de processus

- MS-DOS :

- Un appel système pour charger un fichier binaire en mémoire et l'exécuter en tant que processus fils
- Contrairement à Unix, MS-DOS suspend le père jusqu'à ce que le fils ai terminé son exécution

II.4 Relations entre processus

Synchronisation

- La synchronisation est un terme lié au temps, c'est une définition qui décrit les contraintes par rapport au temps selon lesquelles les instructions d'un processus doivent être exécutées par rapport aux autre processus.
- On dit qu'un processus est synchronisé avec d'autres processus s'il a la possibilité de:
  - ❖Se bloquer,
  - ❖Bloquer un autre processus,
  - ❖Réveiller un autre processus.

⇒ L'accès à des ressources partagées entre plusieurs processus nécessite des synchronisations explicites pour être correct.

II.4 Relations entre processus

Compétition

- Situation dans laquelle plusieurs processus doivent utiliser **simultanément** une ressource à **accès exclusif**
- La ressource ne pouvant être utilisée que par un seul processus à la fois
- Exemples
  - processeur (cas du pseudo-parallélisme)
  - imprimante
- Une solution possible (mais non la seule) : faire attendre les processus demandeurs jusqu'à ce que l'occupant actuel ait fini (premier arrivé, premier servi)

II.4 Relations entre processus

Coopération

- Situation dans laquelle plusieurs processus collaborent à une tâche commune
- Les processus doivent se synchroniser pour réaliser cette tâche
- Exemples
  - p1 produit un fichier, p2 imprime le fichier
  - p1 met à jour un fichier, p2 consulte le fichier

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

II.5 Techniques d'ordonnancement de processus

Objectif : Choisir le prochain processus à exécuter de façon optimale.

Un changement de processus suppose :

– enregistrer le contexte du processus actuel (registres, config mémoire, …)

– charger le contexte du prochain processus (registres, config mémoire, …)

II.5 Techniques d'ordonnancement de processus

- ## Allocation du CPU et changement de contexte

"Retirer" le processeur à un processus et l'allouer à un autre processus passe par une opération de sauvegarde du contexte de l'ancien processus et chargement du contexte du nouveau processus. Si plusieurs processus sont à l'état prêt, le système d'exploitation doit choisir un processus en particulier qui deviendra le processus actif. Ce choix est basé sur un algorithme d'ordonnancement et le module du système d'exploitation qui joue ce rôle s'appelle *l'ordonnanceur* (scheduler).

- ## Ordonnancement des processus

L'ordonnanceur (scheduler) est un module du système d'exploitation qui attribue le contrôle du CPU à tour de rôle aux différents processus en compétition suivant une politique définie à l'avance par les concepteurs du système.

Critères d'ordonnancement

La politique d'ordonnancement doit optimiser les performances du système à travers plusieurs critères :

- ✓Utilisation de l'UC (rendement) : pourcentage d'utilisation
- ✓Débit : nombre de processus qui terminent leur exécution par unité de temps.
- ✓Equité : capacité de l'ordonnanceur à allouer le CPU d'une façon équitable à tous les processus de même priorité (éviter la famine).
- ✓Temps de rotation : le temps depuis le lancement du processus jusqu'à sa terminaison (les attentes incluses).
- ✓Temps d'attente : temps d'un processus dans la file d'attente des processus prêts.
- ✓Temps de réponse (pour les systèmes interactifs) : temps moyen qui s'écoule entre le moment où un utilisateur soumet une requête et celui où il commence à recevoir les réponses (le temps entre une demande et la réponse).

Equations

- *Temps de rotation = Date de fin d'Exe. – Temps d'arrivée*
- *Temps d'attente = Temps de rotation – Durée d'Exe.*
- *Temps d'attente = Temps début exécution – Date d'arrivée*

- *Temps d'attente moyen :*

$$\text{TAM} = \sum_{\text{processus}} (\text{date début d'exécution} - \text{date d'arrivée}) \quad / \quad \textit{nombre de processus}$$

- *Temps de rotation moyen :*

$$\text{TRM} = \sum_{\text{processus}} (\text{date de fin d'exécution} - \text{date d'arrivée}) \quad / \quad \textit{nombre de processus}$$

II.6 Algorithmes d'ordonnancement

Deux catégories d'algorithmes

L'ordonnanceur du CPU permet de décider à quel processus (dans la file d'attente des processus prêts) attribuer le contrôle du CPU. Les stratégies d'ordonnancement peuvent être regrouper en deux catégories : sans réquisition du CPU (stratégie non préemptive) ou avec réquisition du CPU (stratégie préemptive).

a) Ordonnancement sans réquisition du CPU

Dans cette catégorie, un processus conserve le contrôle du CPU jusqu'à ce qu'il se bloque ou qu'il se termine. Cette approche correspond aux besoins des travaux par lots (systèmes batch).

Exemple : Systèmes à temps réel : non préemptif (traitement critique non interruptible).

- Il existe plusieurs algorithmes dans cette catégorie, on peut citer parmi eux :

❖ Premier Arrivé Premier Servi (FCFS : First come First Serve)

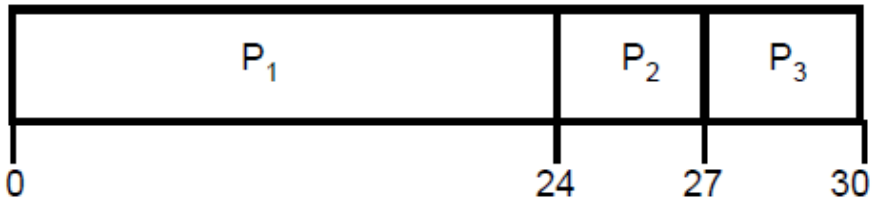
Les tâches sont ordonnancées dans l'ordre où elles sont reçues. Le processus qui sollicite le CPU le premier sera servi le premier. On utilise une structure de file.

- **Exemple :**

Processus	Temps d'exécution
P1	24
P2	3
P3	3

Si les processus arrivent au temps 0 dans l'ordre: P1 , P2 , P3

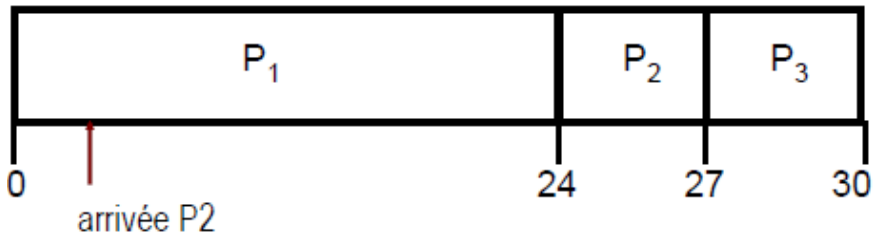
Le diagramme Gantt (exécution) est:



- Temps d'attente pour P1= 0; P2= 24; P3= 27
- Temps attente moyen: (0 + 24 + 27)/3 = 17
- Utilisation UCT = 100%
- Débit = 3/30 = 0,1
- 3 processus complétés en 30 unités de temps
- Temps de rotation moyen: (24+27+30)/3 = 27

❖ Tenir compte du temps d'arrivée!

- Dans le cas où les processus arrivent à moment différents, il faut soustraire les temps d'arrivée
- **Exercice**: répéter les calculs si:
  - P1 arrive à temps 0 et dure 24
  - P2 arrive à temps 2 et dure 3
  - P3 arrive à temps 5 et dure 3
- Donc P1 attend 0 comme avant
- Mais P2 attend 24-2, etc.



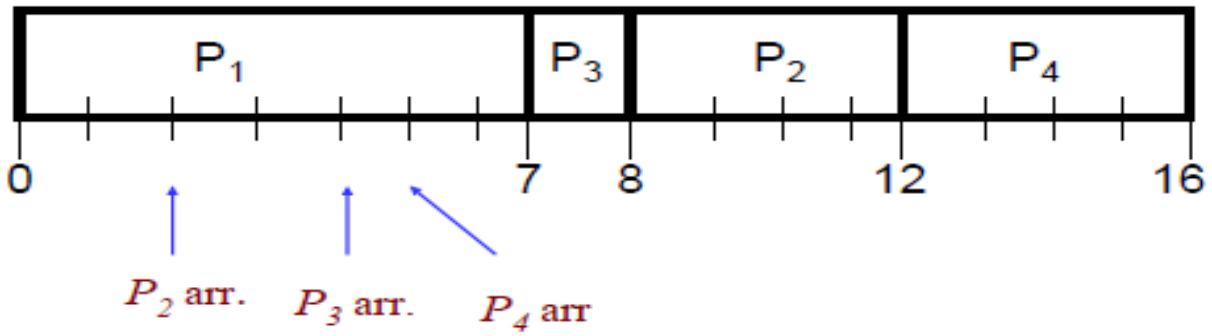
❖Le Plus Court Job d'abord (SJF: Shortest Job First)

Cet algorithme nécessite la connaissance du temps d'exécution estimé de chaque processus. Le CPU est attribué au processus dont le temps d'exécution estimé est minimal.

Exemple :

Processus	Temps d'arrivée	Temps d'exécution
P1	0	7
P2	2	4
P3	4	1
P4	5	4

- Diagramme d'exécution :



- **Temps d'attente moyen** = (0+(8-2)+(7-4)+(12-5))/4
  = (0 + 6 + 3 + 7)/4 = 4
- **Temps de rotation moyen** = 7+(12-2)+(8-4)+(16-5) = 8

Difficultés majeures avec les algorithmes discutés

- Premier arrivé, premier servi, FCFS:
  - Temps moyen d'attente non-optimal
  - Mauvaise utilisation des ressources s'il y a apport continu de processus aux cycles longs (v. effet d'accumulation)
- Plus court servi, SJF:
  - Difficulté de prévoir la durée du prochain cycle
  - *Famine* possible des processus 'longs' s'il y a apport continu de processus aux cycles courts

✓ Donc besoin d'une méthode systématiquement préemptive

b) Ordonnancement avec réquisition du CPU

Dans cette catégorie, l'ordonnanceur peut retirer le CPU à un processus avant que ce dernier ne se bloque ou se termine afin de l'attribuer à un autre processus.

Exemple :

- Systèmes à temps partagé : préemptif.

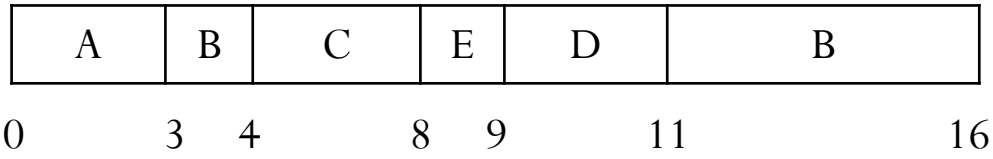
❖Politique du Job ayant le Plus Court Temps Restant (SRTF, Shortest Remaining Time First)

Lorsqu'un processus est en cours d'exécution, et qu'un nouveau processus ayant un temps d'exécution plus court que celui qui reste pour terminer l'exécution du processus en cours, ce processus est arrêté (préempté), et le nouveau processus est exécuté. Cette méthode équivaut à la méthode SJF mais préemptive.

Exemple : Le temps restant le plus court (SRT : Shortest Remaining Time)

Processus	Durée d'Exe.	Temps d'arrivée	Date début d'Exe.	Préempté A	Repris A	Date fin d'Exe.	Temps de rotation	Temps d'attente
A	3	0	0			3	3	0
B	6	1	3	4	11	16	15	9
C	4	4	4			8	4	0
D	2	6	9			11	5	3
E	1	7	8			9	2	1
Temps de rotation moyen			(3+15+4+5+2)/5 = 29/5 = 5,8 UT					
Temps d'attente moyen			(0+9+0+3+1)/5 = 13/5 = 2,6 UT					
Nbre de changements de contexte			5					

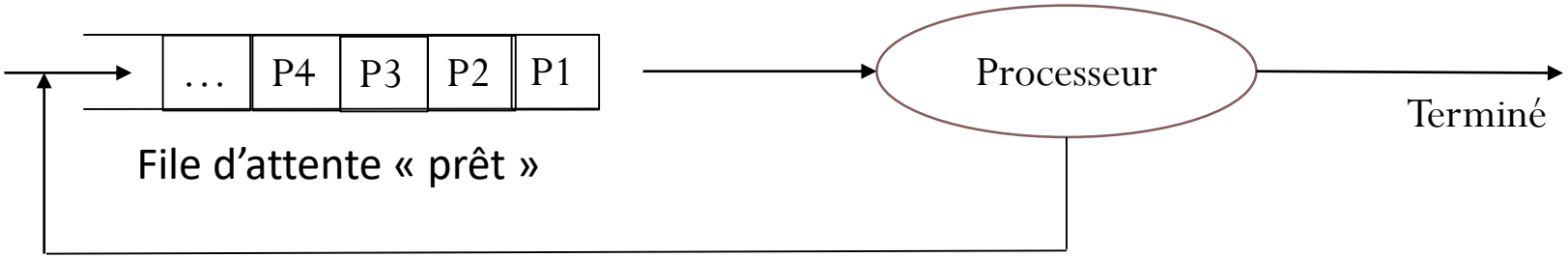
Le diagramme d'exécution :



➢Les travaux longs peuvent être victimes de famine.

❖Algorithme à Tourniquet (RR : Round Robin)

Le contrôle du CPU est attribué à chaque processus pendant une tranche de temps Q, appelée quantum, à tour de rôle. Lorsqu'un processus s'exécute pendant un quantum, le contexte de ce dernier est sauvegardé et le CPU est attribué au processus suivant dans la liste des processus prêts (Figure).



Si le quantum n'est pas épuisé alors retour à la file d'attente

Problème : choix du quantum de temps :

–Trop court=> beaucoup de changements => perte importante de temps en changements

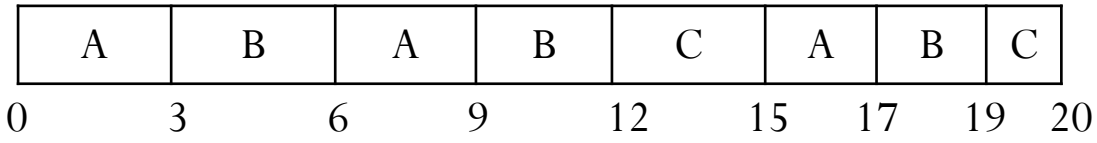
–Trop long=> temps de réponse long (pas bon pour interactif ou multimédia)

Exemple : Tourniquet (RR)

On suppose que le Quantum = 3 et que la durée de commutation = 0.

Processus	Durée d'Exe.	Temps d'arrivée	Date début d'Exe.	Préempté A	Repris A	Date fin d'Exe.	Temps de rotation	Temps d'attente
A	8	0	0	3	6	17	17	9
				9	15			
B	8	2	3	6	9	19	16	8
				12	17			
C	4	7	12	15	19	20	13	9
Temps de rotation moyen			(17+16+13)/3 = 46/3 = 15,33 UT					
Temps d'attente moyen			(9+8+9)/3 = 26/3 = 8,66 UT					

Le diagramme d'exécution :



❖Algorithme avec priorité

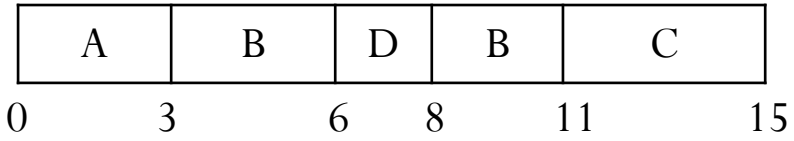
Le système associe à chaque processus une priorité : nombre mesurant l'importance du processus dans le système. Le CPU est attribué au processus de plus haute priorité. En cas d'égalité, l'algorithme FCFS est utilisé. L'ordonnancement des priorités peut être avec ou sans réquisition.

Exemple :

On suppose dans cet exemple que la valeur la plus grande correspond à la priorité la plus élevée.

Processus	Durée d'Exe.	Temps d'arrivée	Prio.	Date début d'Exe.	Préempté A	Repris A	Date fin d'Exe.	Temps de rotation	Temps d'attente
A	3	0	3	0			3	3	0
B	6	3	1	3	6	8	11	8	2
C	4	5	2	11			15	10	6
D	2	6	0	6			8	2	0
Temps de rotation moyen			(3+8+10+2)/4 = 23/4 = 5,75 UT						
Temps d'attente moyen			(0+2+6+0)/4 = 8/4 = 2 UT						

Le diagramme d'exécution : AAABBBDDBBBCCCC



Problème possible avec les priorités

- Famine: les processus moins prioritaires n'arrivent jamais à exécuter
- Solution: vieillissement:

– modifier la priorité d 'un processus en fonction de son âge et de son historique d'exécution

Conclusion

Ce chapitre était consacré à la gestion des processeur. Après la sélection du processus par l'ordonnanceur, ce dernier se charge en mémoire centrale pour s'exécuter. La mémoire centrale représente le lieu d'exécution des processus. Le prochain chapitre traitera en détail la gestion de la mémoire centrale.