17.2 Ordonnancement

NSI terminale - JB Duthoit

Dans un système multitâche, plusieurs processus sont actifs simultanément, mais un processeur (simple coeur) ne peut exécuter qu'une instruction à la fois !

☞ Nous allons donc avoir l'impression que le processeur gère les processus de façon simultanée, mais il en est rien.

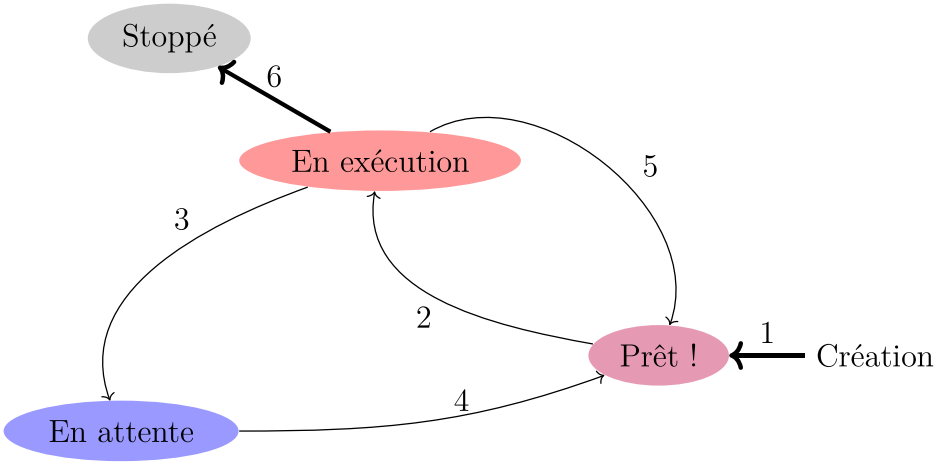
Il va donc falloir partager le temps de processeur disponible entre tous les processus : c'est le travail de l'ordonnanceur. Ce dernier a pour tâche de sélectionner le processus suivant à exécuter parmi ceux qui sont prêts.

Définition

L'**ordonnanceur** gère les processus, ayant pour objectif de partager le temps d'utilisation du processeur.

Afin de gérer cet ordonnancement, les processus se voient affectés d'états différents :

- prêt (ready) : le processus attend son tour pour prendre la main
- (running) : le processus a accès au processeur pour exécuter ses instructions
- (sleeping) : le processus attend qu'un événement se produise (saisie clavier, réception d'une donnée par le réseau ou le disque dur ...) en exécution .
- (stopped) : le processus a fini son travail ou a reçu un signal de terminaison (SIGTERM, SIGKILL, ...). Il libère les ressources qu'il occupe.



1 Le processus est créé ; il est "prêt"

2 **Eléction** L'ordonnanceur choisit ce processus. Il entre en exécution.

3 Le processus se met en attente d'un événement.

4 L'événement attendu se produit. Le processus est donc "prêt"

5 L'ordonnanceur décide de suspendre le processus afin de donner la main à un autre processus.

6 Le processus est terminé.

☞ Il est vraiment important de bien comprendre que c'est le système d'exploitation qui attribue aux processus les différents états "En exécution, En attente, prêts".
On dit que le système d'exploitation gère l'ordonnancement des processus (tel processus sera prioritaire sur tel autre...)

Afin d'élire le processus qui va repasser en mode exécution, l'ordonnanceur applique un algorithme prédéfini lors de la conception de l'OS.
Le choix de cet algorithme va impacter directement la réactivité du système et les usages qui pourront en être fait. C'est un élément critique du système d'exploitation.

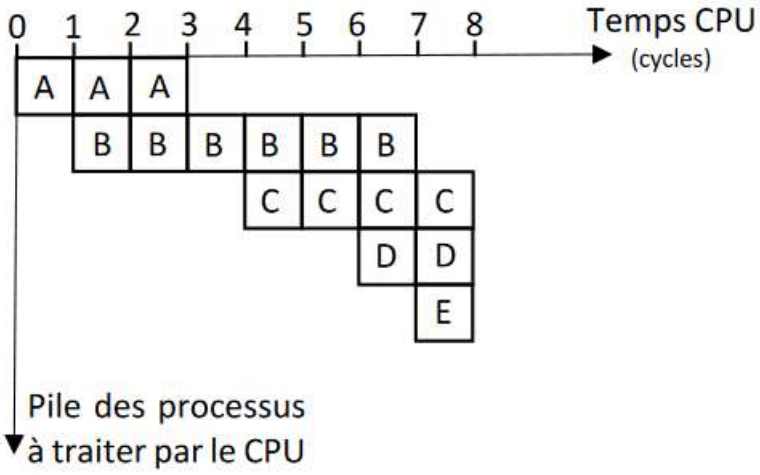
Parmi les algorithmes les plus répandus :

- Le **tourniquet** : La ressource est affectée à chaque processus à tour de rôle. Cette méthode à plusieurs avantages : simplicité, rapidité de gestion, robustesse.
- Mise en place d'un système de priorité : l'ordre de l'affectation de la ressource sera alors fonction de la priorité des tâches.
  ⚠ Attention au réglage du niveau de priorité, qui doit être "équitable" et "objectif".
- Gestion du premier entrée, premier sortie (FIFO). Comme par exemple la file d'impression des documents sur une imprimante.
- Le plus court d'abord. Assez efficace, mais la difficulté réside dans le fait d'estimer la durée du processus en amont, avant qu'il ne commence.

Exercice 17.168
Considérons cinq processusnotés A, B, C, D et E, dont les temps d'exécution et leurs arrivages respectifs sont donnés dans le tableau ci-contre.

Processus	Temps d'exécution	Temps d'arrivage
A	3	0
B	6	1
C	4	4
D	2	6
E	1	7

On peut également représenter le tableau précédent de la manière suivanteafin de faciliter la compréhension de l'ordonnancement des processus :



Représenter pour chaque algorithme le schéma d'exécution. Calculer le temps de séjours, le temps moyen de séjour, le temps d'attente (soustraction entre le temps de séjour et le temps d'exécution) et enfin le temps d'attente moyen

1. Algorithme "Premier arrivé, premier servi"
2. Algorithme "Le plus court avant".

**