TD2 : Ordonnancement de processus

QCM

1- Un processus est :

- ◻ Un programme exécutable
- ◻ Un logiciel
- ◻ une instance d'exécution d'un programme

2- Une tâche est :

- ◻ Un programme exécutable
- ◻ Un processus
- ◻ Une exécution de programme
- ◻ Un mouvement de données avec les périphériques

3- Le multitâche :

- ◻ nécessite, pour un système d'exploitation, d'avoir en mémoire centrale plusieurs tâches simultanément
- ◻ Permet de commencer l'exécution d'un second programme alors qu'un premier est déjà en exécution, chacun s'exécutant à tour de rôle
- ◻ Ne permet pas le multi-utilisateur

4- Un processus prêt

- ◻ est exécuté
- ◻ a été exécuté par le processeur
- ◻ attend d'obtenir le processeur

5- Un processus bloqué :

- ◻ ne peut plus s'exécuter
- ◻ attend l'achèvement d'un accès à la mémoire.
- ◻ attend d'obtenir le processeur.

6- L'ordonnanceur :

- ◻ donne des instructions pour réparer des processus cassés.
- ◻ transforme un programme en processus.
- ◻ planifie l'exécution d'un processus.

Interblocage

Objectif : Déceler une situation d'interblocage

Sept processus P_i sont dans la situation suivante par rapport aux ressources R_i :

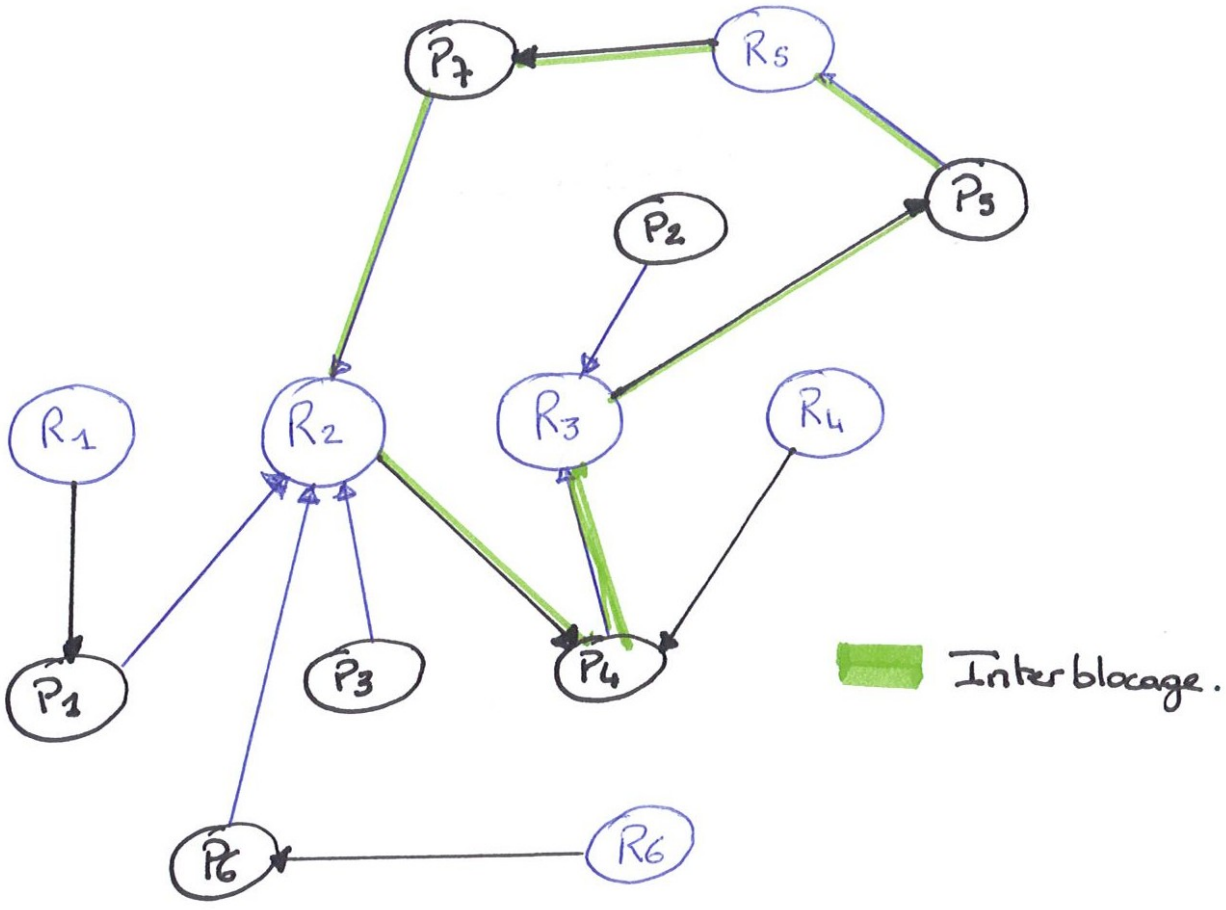
- ➔ P_1 a obtenu R_1 et demande R_2 ;
- ➔ P_2 demande R_3 et n'a obtenu aucune ressource tout comme P_3 demande R_2 ;
- ➔ P_4 a obtenu R_2 et R_4 et demande R_3 ;
- ➔ P_5 a obtenu R_3 et demande R_5 ;
- ➔ P_6 a obtenu R_6 et demande R_2 ;
- ➔ P_7 a obtenu R_5 et demande R_2.

On voudrait savoir s'il y a interblocage.

1. **Construire** un graphe orienté où les sommets sont les processus et ressources et où :
   - ➔ la présence de l'arc $R_i \rightarrow P_j$ signifie que le processus a obtenu la ressource R_i ;
   - ➔ la présence de l'arc $P_j \rightarrow R_i$ signifie que le processus P_j demande la ressource R_i.

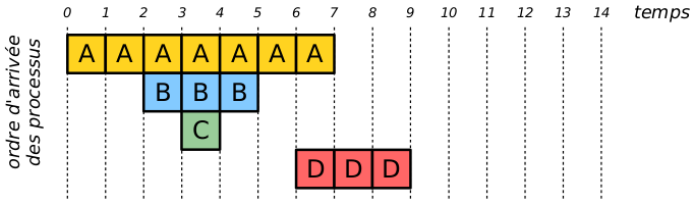
Il y a interblocage lorsque des cycles sont présents dans le graphe.

2. **Chercher** ce cycles afin de déterminer les éventuelles situations d'interblocages.



Règles basiques d'ordonnancement

Un système d'exploitation multitâche est capable de simuler l'exécution simultanée de plusieurs processus indépendamment du nombre de cœurs disponibles dans le microprocesseur.
Les OS actuels sont multitâches préemptifs (Unix (1969), Windows NT 3.1 (1993), AmigaOS (1985), Windows 95 (1995) et Mac OS X (2001)). Ils cherchent à octroyer un certain temps d'exécution au processus avant de reprendre la main de force en sauvegardant l'état du processus, au moyen d'une interruption programmable.
Pour illustrer les différents algorithmes d'ordonnancement de tâches nous considérerons la situation simplifiée de quatre processus A, B, C et D de même priorité dont le moment d'arrivée est donné par l'échelle de temps et de durée indiquée par le découpage en cycles ou quantum de temps pleins (nombre entier de carrés) :



Règles d'ordonnancement etéudiées

➔ La file (FIFO) : le premier processus arrivé est le premier à être exécuté et ce n'est qu'à sa fin que le processus suivant peut commencer. L'algorithme conduit alors à l'ordonnancement suivant :

➔ Le plus rapide à finir en premier (Shortest Remaining Time First) : on choisit d'exécuter en priorité le processus dont le temps d'exécution restant est le plus petit. Comme il est impossible de prévoir à l'avance ce temps d'exécution, on se base sur une estimation du temps restant par le calcul d'une moyenne des temps d'exécution réalisés précédemment. L'algorithme conduit alors à l'ordonnancement suivant :

Travail à faire

Un système d'exploitation reçoit cinq processus notés A, B, C, D et E, dont les temps d'exécution et les instants d'arrivage respectifs sont :

Processus	Temps d'exécution	Temps d'arrivée
A	3	0
B	5	1
C	2	4
D	5	6
E	1	7

1. Le schéma suivant représente la durée et l'ordre d'arrivée des tâches à traiter par le système d'exploitation. **Compléter** ce schéma en tenant compte des informations du tableau précédent.

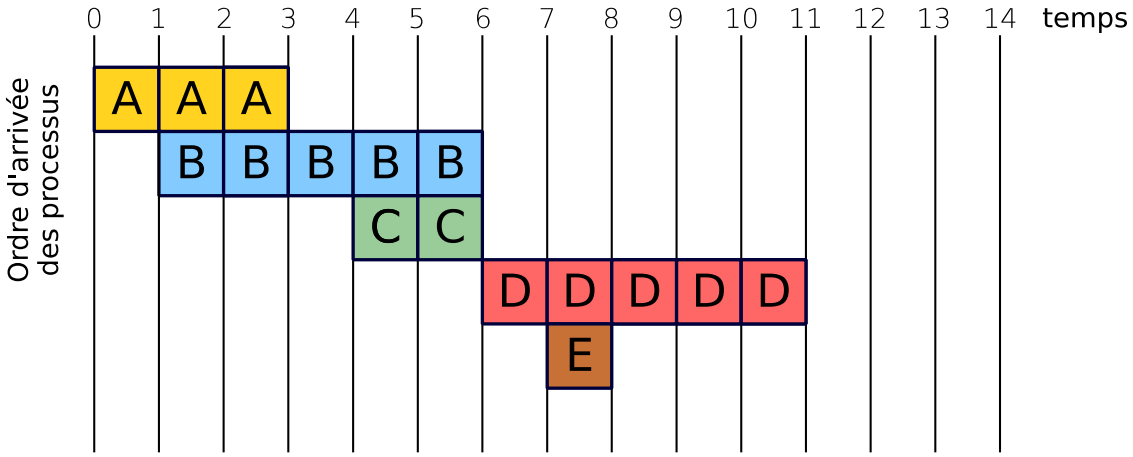


Figure 1: Schéma représentatif des ordres et durées des tâches

2. **Compléter** sur le schéma d'exécution suivant, l'ordre de traitement des tâches dans le cas d'un ordonnancement de type premier arrivé, premier exécuté.

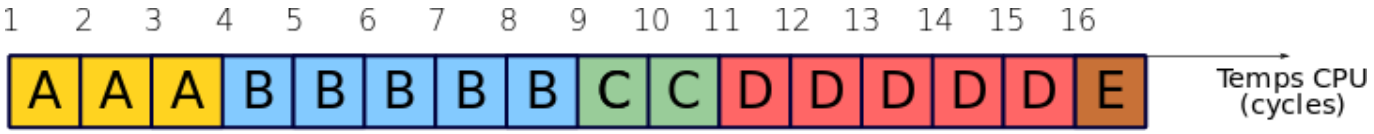


Figure 2: Schéma d'exécution dans le cas d'un ordonnancement "**premier arrivé, premier exécuté**"

Le temps de séjour pour chaque processus est obtenu en soustrayant le temps d'entrée du processus au temps de terminaison. Le temps d'attente est calculé en soustrayant le temps d'exécution au temps de séjour.

3. **Déterminer** les temps de séjour et d'attente pour ces 5 processus.

Processus	**Temps de séjour**	**Temps d'attente**
A	$t_s = 4 - 0 = 4$	$t_a = 4 - 3 = 1$
B	$t_s = 9 - 1 = 8$	$t_a = 8 - 5 = 3$
C	$t_s = 11 - 4 = 7$	$t_a = 7 - 2 = 5$
D	$t_s = 16 - 6 = 10$	$t_a = 10 - 5 = 5$
E	$t_s = 17 - 7 = 10$	$t_a = 10 - 1 = 9$

4. **Compléter** sur le schéma d'exécution suivant l'ordre de traitement des tâches dans le cas d'un ordonnancement de type le plus rapide à exécuter en premier.

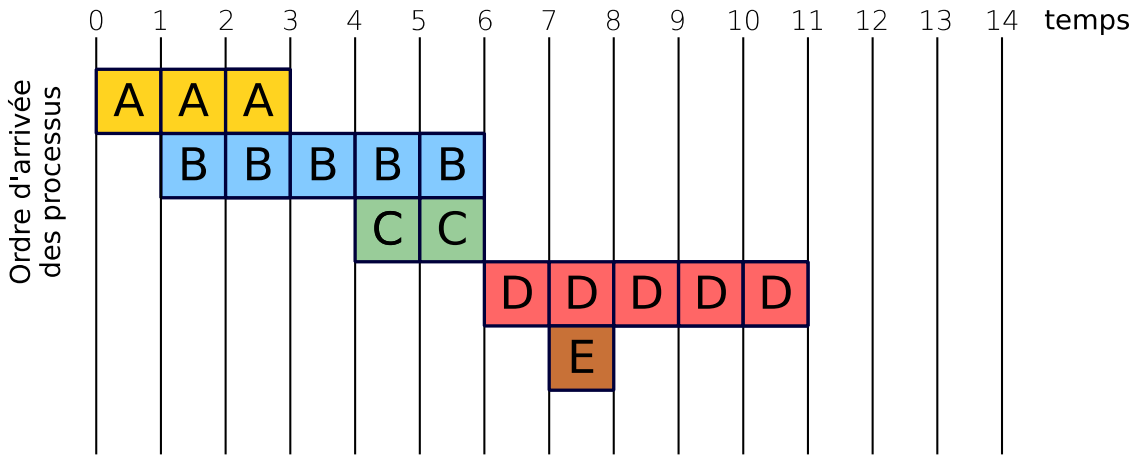


Figure 3: Schéma représentatif des ordres et durées des tâches

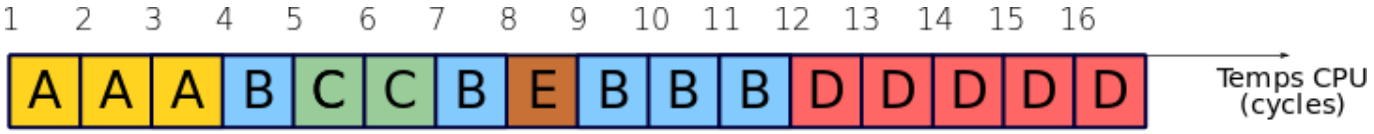


Figure 4: Schéma d'exécution dans le cas d'un ordonnancement "**plus rapide à exécuter en premier**"

5. **Calculer** les temps de séjour et d'attente pour ces 5 processus.

Processus	Temps de séjour	Temps d'attente
A	$t_s = 4 - 0 = 4$	$t_a = 4 - 3 = 1$
B	$t_s = 12 - 1 = 11$	$t_a = 11 - 5 = 6$
C	$t_s = 7 - 4 = 3$	$t_a = 3 - 2 = 1$
D	$t_s = 17 - 6 = 11$	$t_a = 11 - 5 = 6$
E	$t_s = 9 - 7 = 2$	$t_a = 2 - 1 = 1$

6. **Comparer** le temps d'attente moyen de ces deux méthodes d'ordonnancement et **indiquer** le meilleur ordonnancement dans ce cas.

	Ordonnancement **premier arrivé, premier exécuté**	Ordonnancement **plus rapide à exécuter en premier**
Temps d'attente moyen	$t_{moy} = 24 / 5 = 4,8$	$t_{moy} = 15 / 5 = 3$

Ordonnancement 'plus rapide à exécuter en premier' est le plus performant car il permet de réduire de façon importante (-50%) le temps d'attente