Systèmes d’exploitation

Plan

- Introduction aux systèmes d'exploitation
- Fonctions de base
  - Gestion des fichiers
  - **Gestion des processus**
  - Gestion de la mémoire
- Exemples d'OS
  - MS-DOS
  - Linux

Gestion des processus

- Concepts de base
- Etats des processus
- Ordonnancement des processus

Gestion des processus

- **Concepts de base**
- Etats des processus
- Ordonnancement des processus

Concepts de base

- Un **processus** est une entité dynamique qui matérialise un programme en cours d'exécution avec ses propres ressources physiques (mémoire, processeur,…) et logiques (données, variables,...).

- Le système d'exploitation manipule deux types de processus :
  - **Processus système** : processus lancé par le système
  - **Processus utilisateur** : processus lancé par l'utilisateur

Concepts de base

Processus : « préparer gâteau au yaourt »

Programme

Concepts de base

Processus : « préparer gâteau au yaourt »

Données

Concepts de base

Processus : « préparer gâteau au yaourt »

Processeur

Le processus est l'activité qui consiste à préparer le gâteau

Concepts de base

Les fonctionnalités du système d'exploitation en matière de **gestion des processus** sont :

- La création, suppression et interruption de processus
- L'ordonnancement des processus afin de décider d'un ordre d'exécution équitable entre les utilisateurs.
- La synchronisation entre les processus ainsi que la communication.
- La gestion des conflits d'accès aux ressources partagées.
- La protection des processus d'un utilisateur contre les actions d'un autre utilisateur.

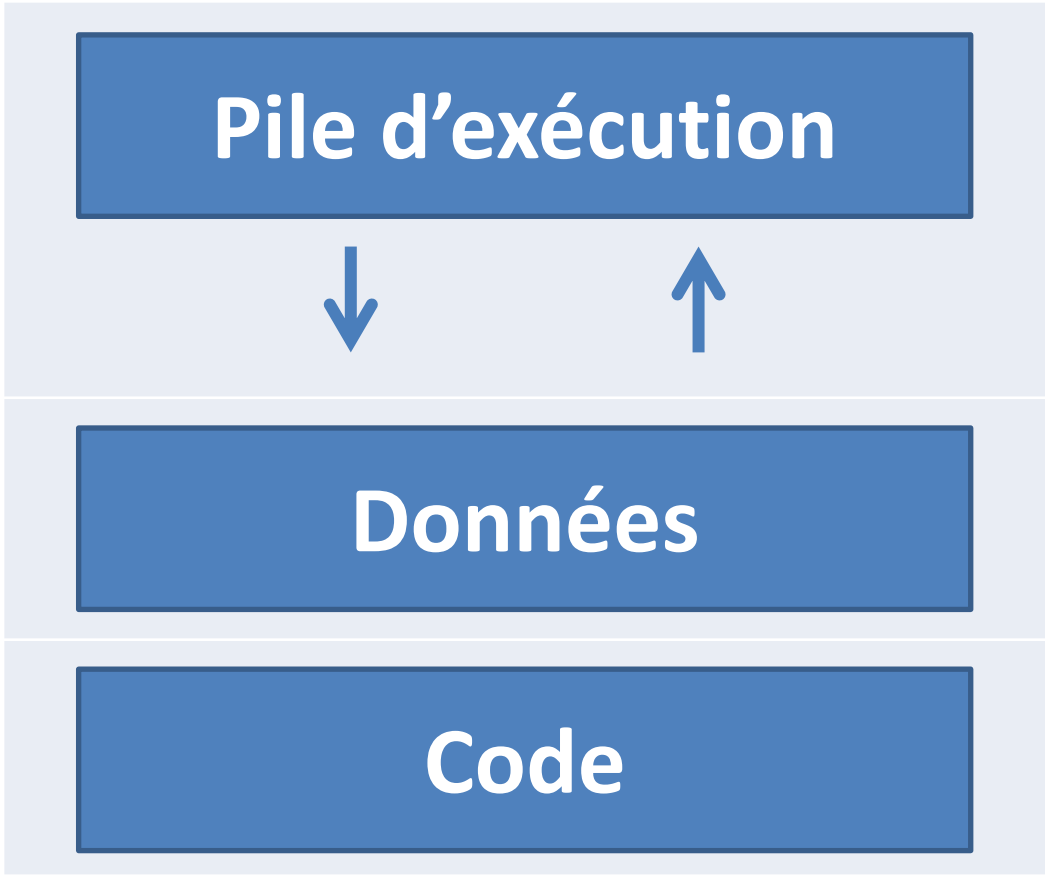
Concepts de base

Le système d'exploitation alloue à chaque processus une zone dans la mémoire centrale appelée **espace d'adressage** du processus. L'espace d'adressage d'un processus comporte :

- Une **zone de données** qui contient les variables globales ou statiques du programme,
- Une **zone de code** qui correspond aux instructions du programme à exécuter,
- Une **pile d'exécution** contenant les appels de fonctions, avec leurs paramètres et leurs variables locales.

Concepts de base

Adresse haute



Adresse basse

Concepts de base

Le **PCB** ou **bloc de contrôle d'un processus** est le contexte d'un processus. Il est créé au même moment que le processus et il est mis à jour en grande partie lors de l'interruption du processus, afin de pouvoir reprendre l'exécution du processus ultérieurement.

Concepts de base

Le PCB contient les informations suivantes :

- Le compteur ordinal : adresse de la prochaine instruction à exécuter par le processeur
- Les contenus des registres généraux : ils contiennent les résultats calculés par le processus
- Les registres qui décrivent l'espace qu'il occupe en mémoire centrale
- Le registre variable d'état qui indique l'état du processus
- D'autres informations telles que la valeur de l'horloge, la priorité du processus, etc.

Concepts de base

Changement de contexte

Quand le CPU passe de l'exécution d'un processus 0 à l'exécution d'un processus 1, l'OS procède comme suit :

1. Mise à jour et sauvegarde du PCB du processus 0,
2. Reprise du PCB du processus 1
3. Remise des registres CPU dans la même situation qui est décrite dans le PCB de processus 1

Concepts de base

Changement de contexte (exemple)

Exemple : La recette du gâteau au yaourt
Evénement : « Piqûre d'abeille »
Interruption du travail : « Enregistre l'endroit de la recette »
Programme : « Livre de première urgence »
Processus : « soin à apporter, calmer son fils »

Gestion des processus

- Concepts de base
- **Etats des processus**
- Ordonnancement des processus

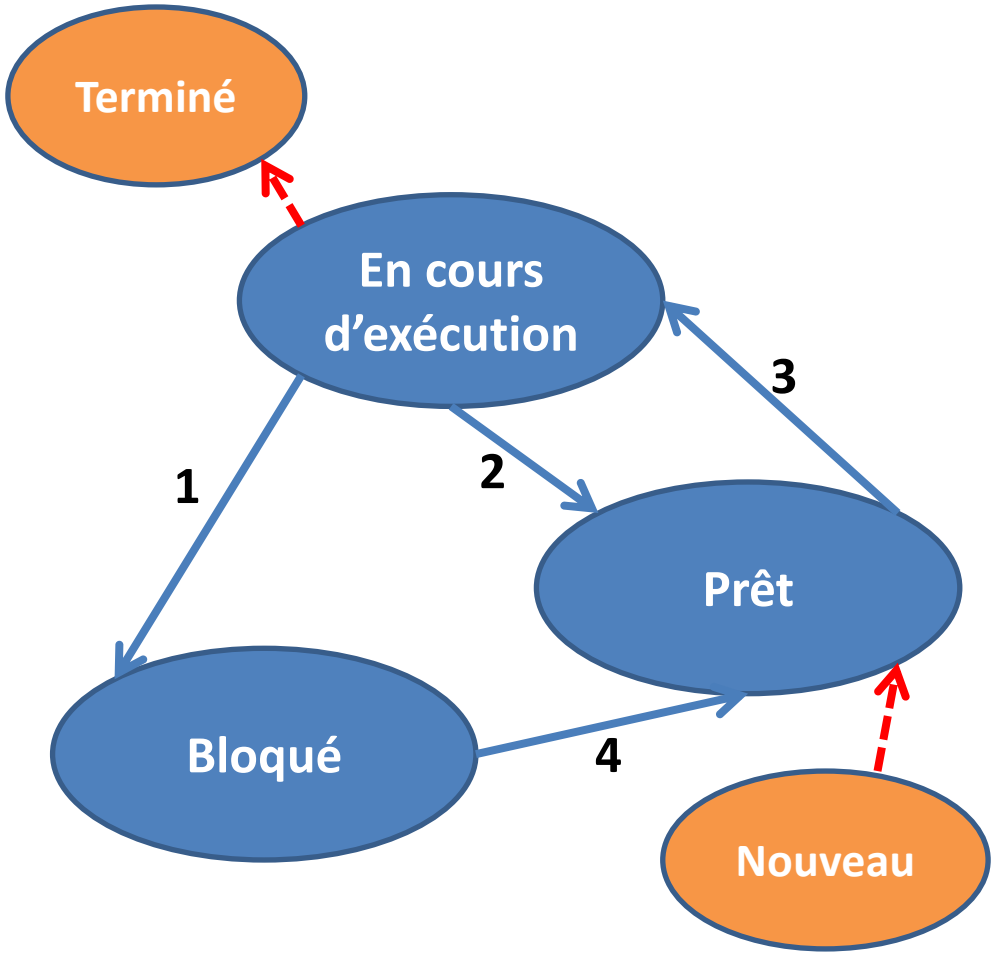
États des processus

- Dans les systèmes mono-tâche, un programme ne quitte pas l'unité centrale avant de terminer son exécution. Pendant cette période, il dispose de toutes les ressources de la machine.

États des processus

- Par contre, dans les systèmes multi-tâches, plusieurs programmes cohabitent en mémoire et le CPU exécute en alternance le code des différents programmes. Dans ces systèmes, un processus peut se trouver dans l'un des états suivants :
  - **En cours d'exécution** : si le processus est en cours d'exécution
  - **Bloqué** : attente d'E/S ou bien d'une ressource pour pouvoir continuer
  - **Prêt** : si le processus dispose de toutes les ressources nécessaires à son exécution à l'exception du processeur.

États des processus



1. Le processus est bloqué en attente de données
2. L'ordonnanceur choisi un autre processus (quantum terminé)
3. L'ordonnanceur choisi ce processus
4. La donnée est disponible

Hiérarchie des processus

- Le système d'exploitation fournit des appels système pour la gestion des processus: création, destruction…
- Un processus peut créer un ou plusieurs processus formant une hiérarchie de processus
- Un processus n'a qu'un seul parent mais peut avoir zéro ou plusieurs enfants.

Création d'un processus

Évènements:

- Initialisation du système (démarrage OS)
- Exécution d'un appel système de création d'un processus par un autre processus en cours d'exécution
- Requête utilisateur sollicitant la création d'un nouveau processus (fork () sous Unix)

Fin d'un processus

Un processus peut se terminer suite à l'un des 4 événements suivants :

1. Sortie normale, lorsque le processus a terminé sa tâche.
2. Sortie suite à une erreur (e.g. division par 0, inexistence d'un fichier passé en paramètre).
3. Tué par un autre processus (sous Unix par l'appel système kill).

Toutes les ressources du processus sont libérées par le SE.

Gestion des processus

- Concepts de base
- Etats des processus
- **Ordonnancement des processus**

Ordonnancement

- Un processeur n'est capable de traiter qu'un seul processus à la fois.
- La fonction d'ordonnancement des processus est assurée par le système d’exploitation : l’ordonnanceur.
- L’ordonnanceur gère le partage du processeur entre les différents processus en attente pour s'exécuter, c'est-à-dire entre les différents processus qui sont dans l'état prêt.

Ordonnancement

- Les objectifs d'un Ordonnanceur sont :
  - Maximiser l'utilisation du processeur
  - Présenter un temps de réponse acceptable
  - Respecter l'équité entre les processus
- L'ordonnancement des processus doit respecter certaines contraintes :
  - Assurer que tout processus fini par obtenir le processeur
  - Ne pas donner la main à un processus bloqué
  - Assurer la réactivité du système
  - Limiter le coût du changement de contexte

Ordonnancement

On peut distinguer deux types d'ordonnancement :

- Coopératif (sans réquisition) : l'ordonnanceur n'intervient que lorsque le processus en cours se termine ou se bloque.
- Préemptif (avec réquisition) : l'ordonnanceur ne peut laisser un processus monopoliser les ressources du système et réquisitionne régulièrement le processeur pour en répartir la disponibilité entre les processus qui simultanément sont prêts à être exécutés.

Algorithmes sans réquisition

Premier arrivé premier servi (FIFO)

Les jobs attendent dans une file. Le premier arrivé est admis immédiatement et s'exécute tant qu'il n'est pas bloqué ou terminé. Lorsqu'il se bloque, le processus suivant commence à s'exécuter et le processus bloqué va se mettre au bout de la file d'attente.

Avantage : algorithme simple
Inconvénient : les processus de faible temps d'exécution peuvent être pénalisés parce qu'un processus de longue durée les précède dans la file.

Algorithmes sans réquisition

Enoncé : en appliquant l'algorithme FIFO, compléter le tableau suivant

PID	Date arrivée	Durée	Début exécution	Fin exécution	Durée de traitement
01	0	5			
02	2	1			
03	3	3			

Temps moyen de traitement = ?

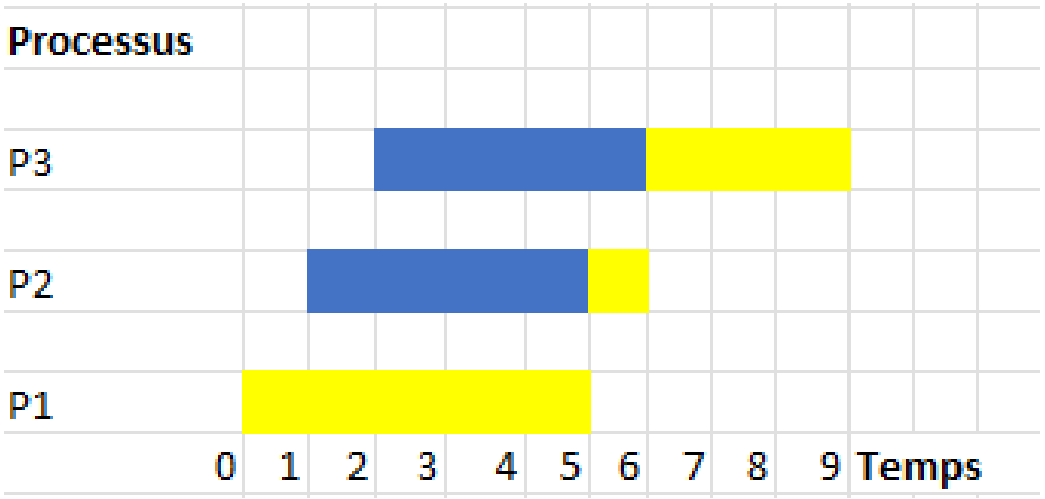
Algorithmes sans réquisition

Réponse :

PID	Date arrivée	Durée	Début exécution	Fin exécution	Durée de traitement
01	0	5	0	5	5 – 0 = 5
02	2	1	5	6	6 – 2 = 4
03	3	3	6	9	9 – 3 = 6

Temps moyen de traitement = 5 + 4 + 6 /3 = 5

Algorithmes sans réquisition



Algorithmes sans réquisition

Le plus court d'abord (SJF)

Sera élu, le processus dont on suppose que le traitement sera le plus court. Dans le cas où plusieurs processus possèdent la même durée, l'algorithme FIFO sera appliqué sur ces processus.

Avantage : optimise le temps moyen d'attente des processus
Inconvénient : Si des processus courts arrivent sans cesse, les processus plus longs n'auront jamais le temps de s'exécuter (famine).

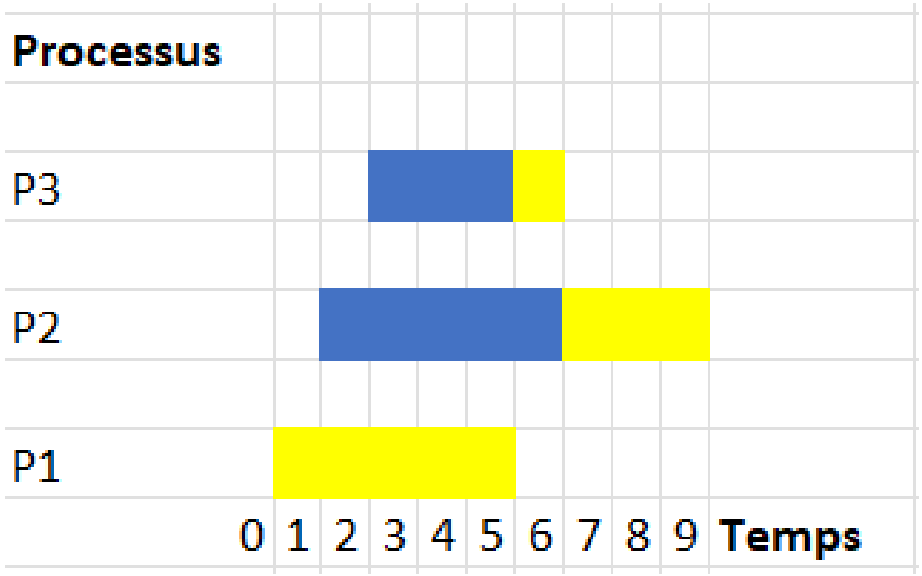
Algorithmes sans réquisition

Enoncé : en appliquant l'algorithme SJF, compléter le tableau suivant

PID	Date arrivée	Durée	Début exécution	Fin exécution	Durée de traitement
01	0	5			
02	2	3			
03	3	1			

Temps moyen de traitement = ?

Algorithmes sans réquisition



Algorithmes sans réquisition

Réponse :

PID	Date arrivée	Durée	Début exécution	Fin exécution	Durée de traitement
01	0	5	0	5	5 – 0 = 5
02	2	3	6	9	9 – 2 = 7
03	3	1	5	6	6 – 3 = 3

Temps moyen de traitement = 5 + 7 + 3 /3 = 5

Algorithmes avec réquisition

Le tourniquet (FIFO avec réquisition)

Chaque processus reçoit tour à tour un intervalle de temps appelé **quantum** pour s'exécuter. Au terme de ce quantum ou, si le processus s'achève ou se bloque avant cet instant, l'ordonnanceur attribue directement le processeur au processus suivant.

Avantage : L'algorithme est simple et équitable.
Contraintes : Un quantum trop court provoque trop de changement de contexte et abaisse l'efficacité du processeur. Un quantum trop long augmente le temps d'attente.

Algorithmes avec réquisition

Enoncé : à t=0, huit processus demandent le processeur dans l'ordre suivant P1, P2, P3, P4, P5, P6, P7 et enfin P8 avec :

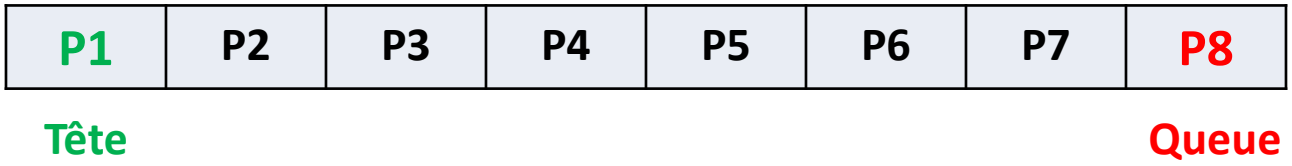
Processus	P1	P2	P3	P4	P5	P6	P7	P8
Durée	3 u	4 u	2 u	3 u	3 u	5 u	4 u	2 u

Donner le diagramme de gantt correspondant à l'exécution de ces processus en appliquant l'algorithme du tourniquet avec un quantum = 2 u.

N.B : le diagramme de gantt représente sur l'axe de temps les processus en cours d'exécution par le processeur.

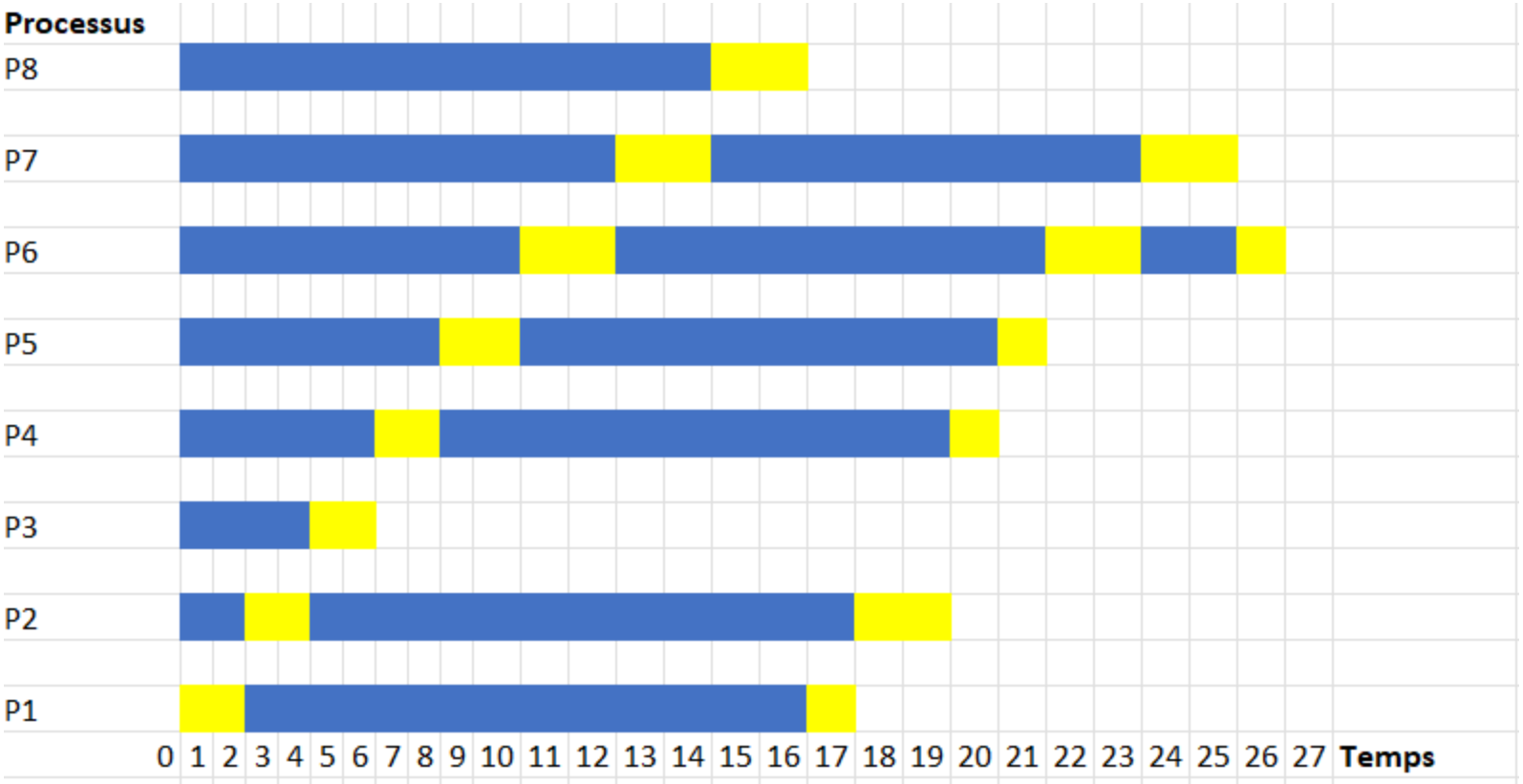
Algorithmes avec réquisition

Réponse : à t=0, la file d'attente des processus **prêts** se présente comme suit



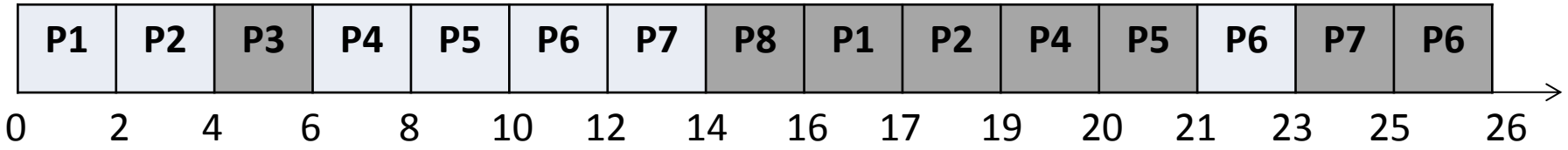
L'ordonnanceur choisit toujours le processus qui se trouve à la **tête**. Le processus choisi prend le processeur au maximum pendant 2u. Au bout du quantum, si le processus a terminé son exécution, il quitte la file, sinon il est placé à la **queue**.

Algorithmes avec réquisition



Algorithmes avec réquisition

Digramme de gantt



Temps moyen de traitement =

$$[17 + 19 + 6 + 20 + 21 + 26 + 25 + 16] / 8 = 18,75$$

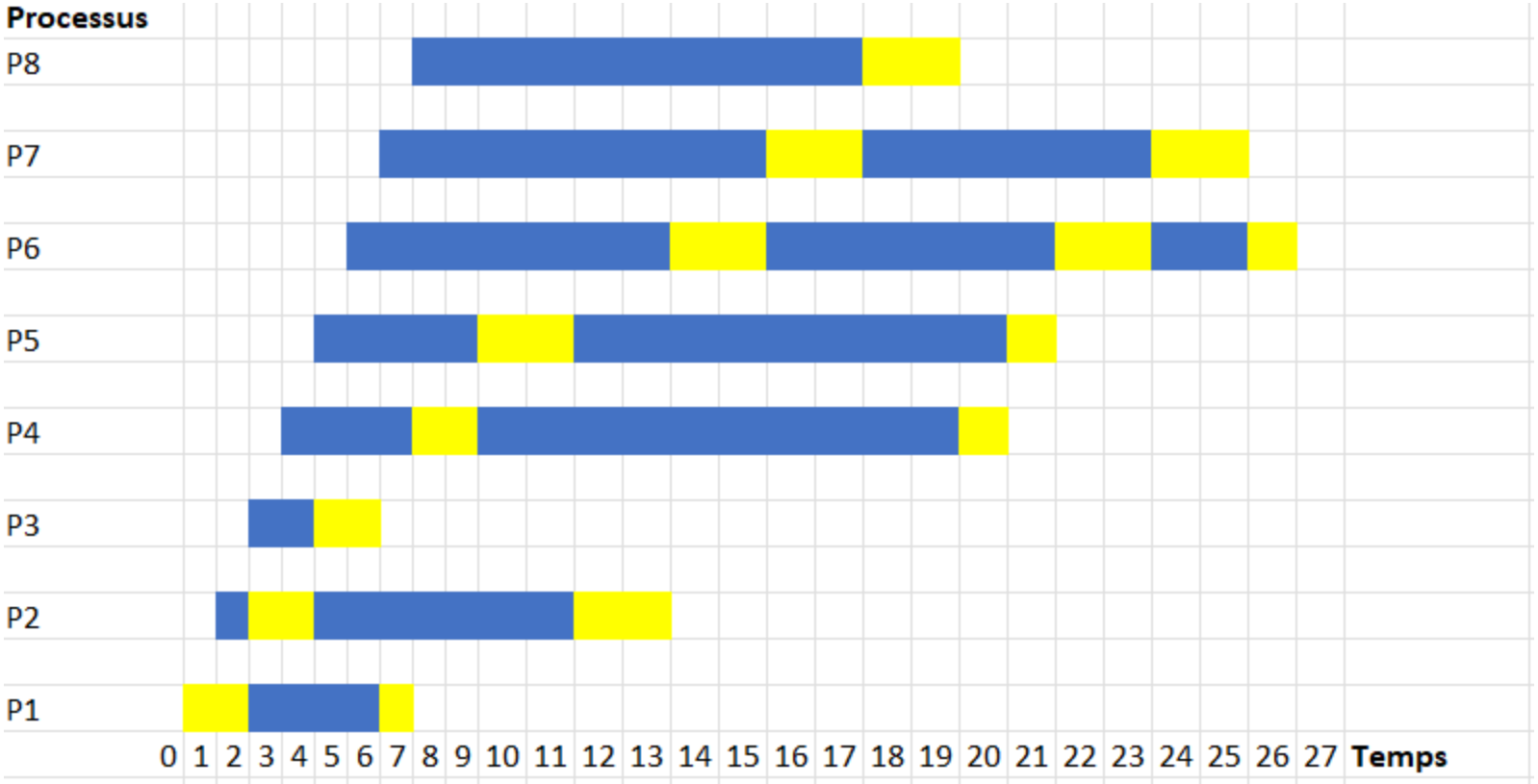
Algorithmes avec réquisition

Enoncé : on reprend le même exercice précédent, sauf que les processus demandent le CPU avec un décalage de 1 u

Processus	P1	P2	P3	P4	P5	P6	P7	P8
Durée	3 u	4 u	2 u	3 u	3 u	5 u	4 u	2 u
Date d'arrivée	0	1 u	2 u	3 u	4 u	5 u	6 u	7 u

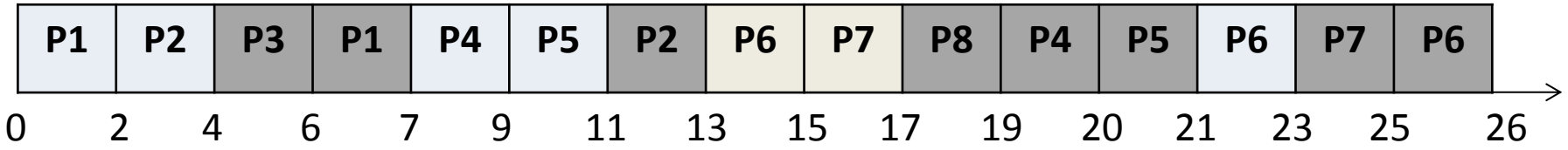
Donner le diagramme de gantt correspondant à l'exécution de ces processus en appliquant l'algorithme du tourniquet avec un quantum = 2 u.

Algorithmes avec réquisition



Algorithmes avec réquisition

Digramme de gantt



Temps moyen de traitement =

[(7-0) + (13-1) + (6-2) + (20-3) + (21-4) + (26-5) + (25-6) + (19-7)] /8 = 13,625

Algorithmes avec réquisition

Le temps restant le plus court (SRT)

C'est la version préemptive de l'algorithme SJF. Au début de chaque quantum, l'ordonnanceur attribue le CPU au processus qui a le plus petit temps restant.

Inconvénients : les processus longs peuvent être victimes de famine.

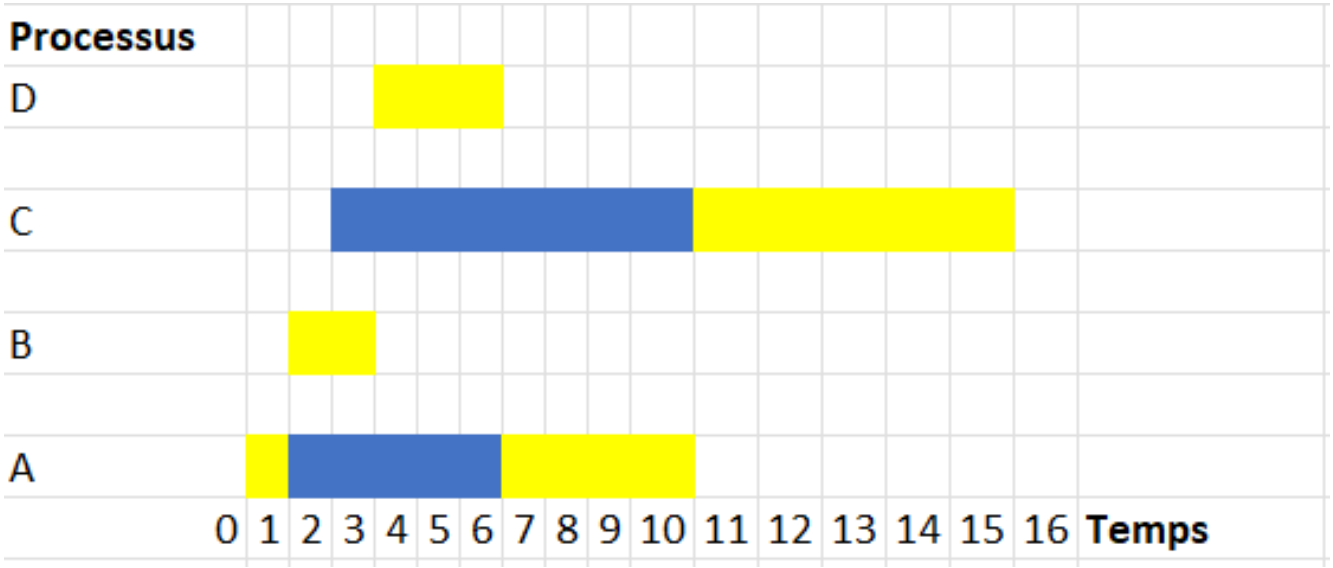
Algorithmes avec réquisition

Enoncé : en appliquant l'algorithme SRT, avec quantum = 1u, compléter le tableau suivant

PID	Date arrivée	Durée	Début exécution	Fin exécution	Durée de traitement
A	0	5			
B	1	2			
C	2	5			
D	3	3			

Temps moyen de traitement = ?

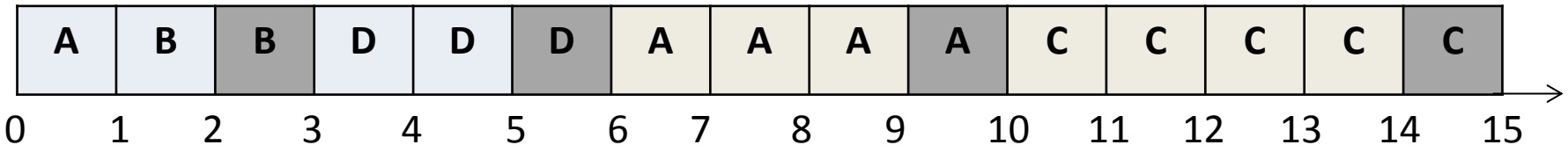
Algorithmes avec réquisition



Algorithmes avec réquisition

Réponse :

PID	Date arrivée	Durée	Début exécution	Fin exécution	Durée de traitement
A	0	5	0	10	10 – 0 = 10
B	1	2	1	3	3 – 1 = 2
C	2	5	10	15	15 – 2 = 13
D	3	3	3	6	6 – 3 = 3



Temps moyen de traitement = 10 + 2 + 13 + 3 /4 = 7

Algorithmes avec ou sans réquisition

Ordonnancement avec priorité

- Une priorité est associée à chaque processus.
- CPU est alloué au processus de plus haute priorité
- Les processus ayant la même priorité sont ordonnancés dans un ordre FIFO.

Inconvénient : **Famine** ou blocage indéfini des processus avec des priorités basses.
Solutions: augmenter la priorité d'un processus avec le temps d'attente. Le processus devient ainsi le plus prioritaire au bout d'un certain temps (Vieillissement).

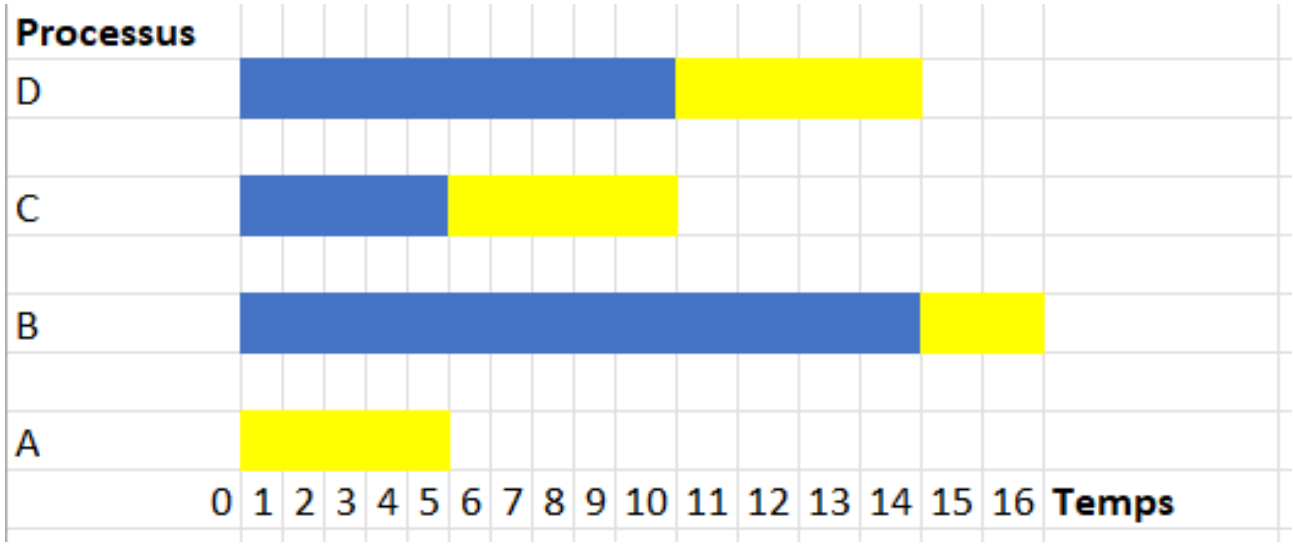
Algorithmes avec ou sans réquisition

Enoncé : à t=0, les 4 processus A, B, C et D demandent le processeur. En appliquant un ordonnancement avec priorité, **sans réquisition**, compléter le tableau suivant. Un indice de priorité faible correspond à une priorité haute.

PID	Priorité	Durée	Début exécution	Fin exécution
A	1	5		
B	4	2		
C	2	5		
D	3	4		

Temps moyen de traitement = ?

Algorithmes avec ou sans réquisition



Algorithmes avec ou sans réquisition

Réponse :

PID	Priorité	Durée	Début exécution	Fin exécution
A	1	5	0	5
B	4	2	14	16
C	2	5	5	10
D	3	4	10	14

Temps moyen de traitement = 5 + 16 + 10 + 14 /4 = 11,25

Références

[1] A. Tanenbaum, Systèmes d'exploitation, 3ème édition, 2008.
[2]http://www.univ-orleans.fr/lifo/Members/Mirian.Halfeld/Cours/SEBlois/SE2007-Processus.pdf
[3]http://langevin.univ-tln.fr/cours/UPS/extra/chapitre3_sgf.pdf
[4] http://www.dphu.org/uploads/attachements/books/books_1843_0.pdf
[5] https://fr.scribd.com/document/193669411/Memoire
[6]https://www.lrde.epita.fr/~didier/lectures/os_06_memory.pdf