

9.3

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. Généralement, l'ordonnanceur applique une politique de temps partagé. L'algorithme d'ordonnancement choisit un processus en attente d'exécution (en fonction de différents critères liés au système d'exploitation). Le processus élu s'exécute alors pendant un certain temps, appelé **quantum** d'ordonnancement. A la fin de ce quantum, l'ordonnanceur interrompt le processus et en choisit un nouveau.

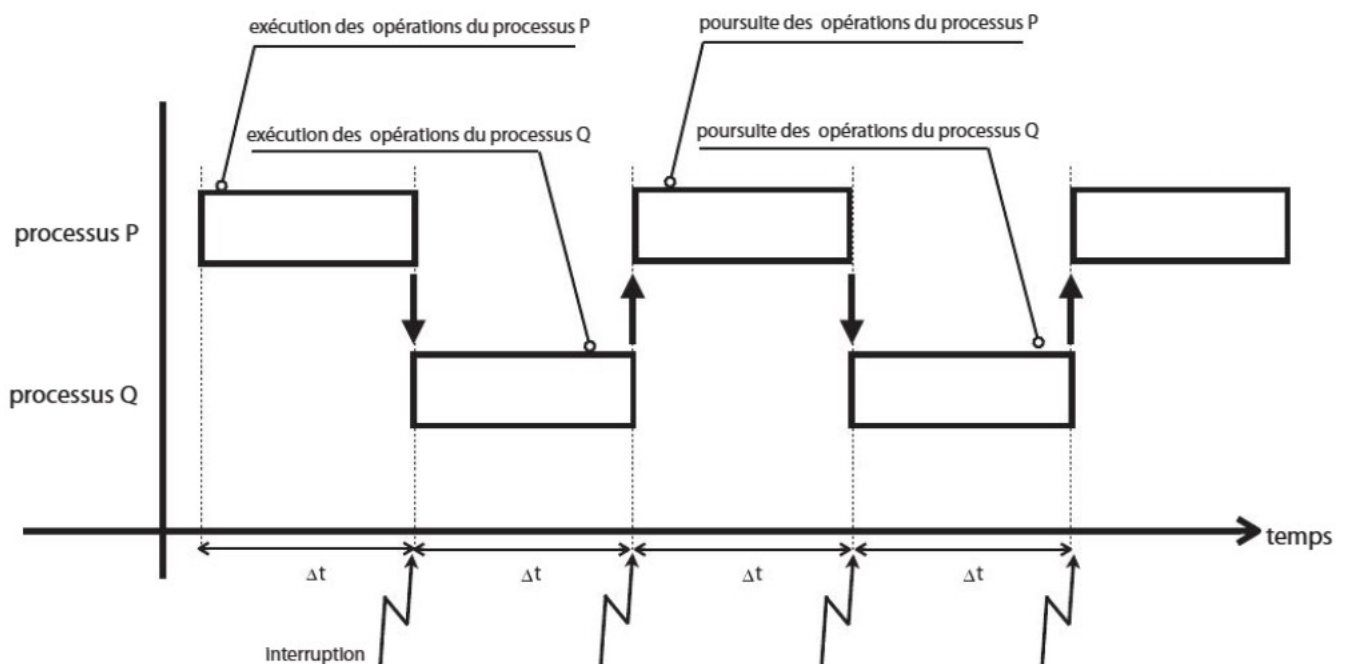


Illustration par un chronogramme de l'ordonnancement de 2 processus

9.3.1 Ressources

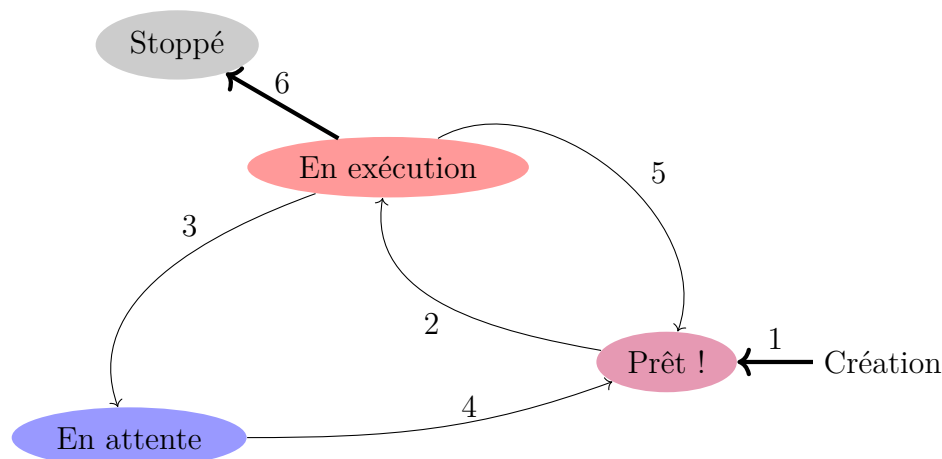
Durant son exécution, le processus peut avoir besoin de ressources matérielles (disque dur, carte réseau...) ou de ressources logicielles.

A un instant donné, une ressource peut être **libre** (aucun processus n'utilise cette ressource) ou **occupé** (un processus utilise la ressource).

9.3.2 État d'un processus

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 **Élection** 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 **préemption** : 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 :

- Les ordonnancements collaboratifs (non préemptifs) : Les tâches ne sont pas interrompibles :

- Gestion du premier entrée, premier sortie (**FIFO**).
 - **SRTF** (Shortest Remaining Time First) ou **SJF** (Shortest Job First) : 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.
 - 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".
- Les ordonnancements **préemptifs** : Les tâches ne sont interruptibles :
 - Le modèle **Round Robin** (**tourniquet**) : La ressource est affectée à chaque processus à tour de rôle. Cette méthode a plusieurs avantages : simplicité, rapidité de gestion, robustesse.

Exercice 9.3

Considérons trois processus notés P1, P2 et P3, dont les temps d'exécution et leurs arrivages respectifs sont donnés dans le tableau ci-dessous.

| Processus | P1 | P2 | P3 |
|--------------------|----|----|----|
| Durée (en quantum) | 8 | 5 | 9 |
| Arrivée | 8 | 3 | 0 |

Construire le schéma d'ordonnancement suivant le modèle FIFO, "premier arrivé, premier servi" ou "Le premier d'abord".

⚠ Cet ordonnancement est non préemption

Exercice 9.4

Considérons quatre processus notés P1, P2, P3 et P4, dont les temps d'exécution et leurs arrivages respectifs sont donnés dans le tableau ci-dessous.

| Processus | P1 | P2 | P3 | P4 |
|--------------------|----|----|----|----|
| Durée (en quantum) | 8 | 5 | 9 | 2 |
| Arrivée | 4 | 0 | 3 | 7 |

Construire le schéma d'ordonnancement suivant le modèle "Le plus court d'abord"

Exercice 9.5

Considérons 3 processus notés P1, P2 et P3, dont les temps d'exécution et leurs arrivages respectifs sont donnés dans le tableau ci-dessous.

| Processus | P1 | P2 | P3 |
|--------------------|----|----|----|
| Durée (en quantum) | 8 | 5 | 9 |
| Arrivée | 1 | 0 | 3 |

Construire le schéma d'ordonnancement suivant le modèle "du tourniquet". Parmi l'ensemble des processus prêts, l'ordonnanceur choisit ici celui qui n'a pas été exécuté depuis le plus longtemps. Si un processus vient d'arriver et n'a pas encore été exécuté, il est prioritaire.

Exercice 9.6

Considérons cinq processus notés P1,P2,P3,P4 et P5 dont les temps d'exécution et leurs arrivages respectifs sont donnés dans le tableau ci-dessous.

| Processus | P1 | P2 | P3 | P4 | P5 |
|----------------------|----|----|----|----|----|
| Durée (en quantum) | 3 | 6 | 4 | 2 | 1 |
| Arrivée | 0 | 1 | 4 | 6 | 7 |
| Temps de terminaison | | | | | |
| Temps de séjour | | | | | |
| Temps d'attente | | | | | |

Répondre aux questions suivantes :

1. Construire le schéma d'ordonnancement suivant le modèle "Le plus court en premier"
2. Compléter le tableau

Définition

Le **temps de terminaison** d'un processus est la durée écoulée entre le temps 0 et le temps où le processus est terminé.

Définition

Le **temps de séjour (ou temps d'exécution)** d'un processus est la différence du temps d'arrivée et du temps de terminaison.

Définition

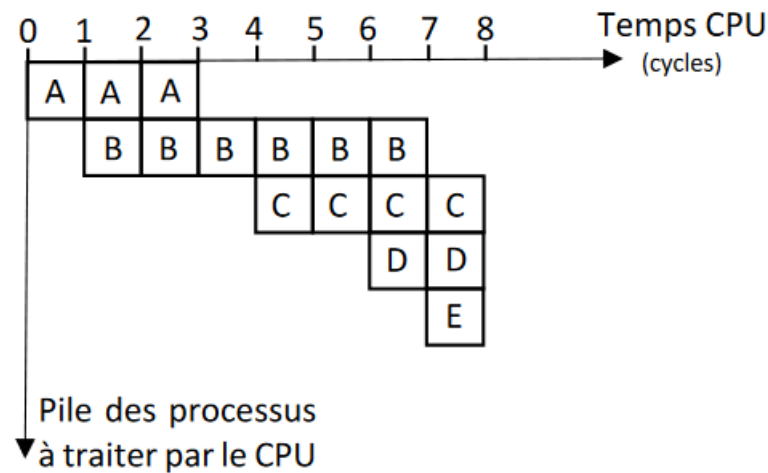
Le **temps d'attente** d'un processus est la différence entre le temps de séjour et la durée du processus.

Exercice 9.7

Considérons cinq processus notés 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'arrivée |
|-----------|-------------------|-----------------|
| 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 suivante afin de faciliter la compréhension de l'ordonnancement des processus :



Représenter pour chaque algorithme le schéma d'exécution. Calculer le temps moyen de séjour et le temps d'attente moyen.

1. Algorithme "Premier arrivé, premier servi"
2. Algorithme "Le plus court avant".
3. Algorithme du tourniquet. On considère ici que la priorité est donnée au processus qui n'a pas été exécuté depuis le plus longtemps. Un nouveau processus, non encore exécuté, est prioritaire.

Exercice 9.8

On considère un système monoprocesseur utilisant un algorithme d'ordonnancement préemptif de type **Tourniquet (Round-Robin)** avec un quantum de temps $q = 2$ unités de temps (u.t.).

Les processus prêts sont gérés dans une **file d'attente de type FIFO** (First In, First Out).

Les processus se partagent une ressource unique R utilisée en **exclusion mutuelle**.

Trois processus sont soumis au système avec les caractéristiques suivantes :

- **Processus A :**

- Arrive au temps $t = 0$.
- Nécessite **9 u.t.** d'exécution totale.
- Demande la ressource R après **3 u.t.** d'exécution et la conserve pendant **5 u.t.**

- **Processus B :**

- Arrive au temps $t = 2, 5$.
- Nécessite **6 u.t.** d'exécution totale.
- Demande la ressource R après **1 u.t.** d'exécution et la conserve pendant **4 u.t.**

- **Processus C :**

- Arrive au temps $t = 3$.
- Nécessite **4 u.t.** d'exécution totale.
- N'utilise pas la ressource R .

Pour traiter cet exercice, vous respecterez les règles suivantes :

1. **Arrivée en cours de quantum :** L'arrivée d'un nouveau processus dans la file des prêts ne provoque pas de préemption immédiate. Le processus en cours termine son quantum (sauf s'il se bloque).
2. **Blocage :** Si un processus demande la ressource R alors qu'elle est occupée, il passe immédiatement à l'état *Bloqué* et cède le processeur (même si son quantum n'est pas épuisé).

3. **Libération et Priorité** : Lorsqu'un processus libère R , le processus en attente est débloquent. En cas de concurrence pour l'entrée dans la file des prêts (fin de quantum simultanée à un déblocage), l'ordre de priorité est :

Processus débloquent > Nouveau processus > Processus sortant (fin de quantum)

1. Représentez le diagramme de Gantt de l'exécution des processus.

| t | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 |
|---|---|---|---|---|---|---|---|---|---|---|----|----|----|----|----|----|----|----|----|
| P | | | | | | | | | | | | | | | | | | | |

2. Indiquez, pour chaque changement de contexte, le contenu de la file d'attente des processus prêts.

Exercice 9.9

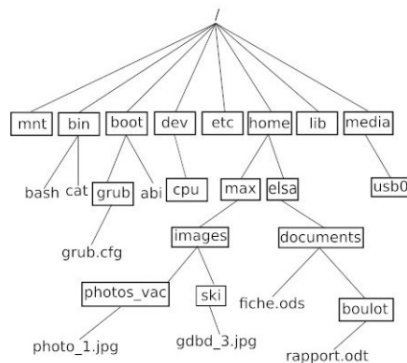
"Linux ou GNU/Linux est une famille de systèmes d'exploitation open source de type Unix fondée sur le noyau Linux, créé en 1991 par Linus Torvalds. De nombreuses distributions Linux ont depuis vu le jour et constituent un important vecteur de popularisation du mouvement du logiciel libre."

Source : Wikipédia, extrait de l'article consacré à GNU/Linux.

"Windows est au départ une interface graphique unifiée produite par Microsoft, qui est devenue ensuite une gamme de systèmes d'exploitation à part entière, principalement destinés aux ordinateurs compatibles PC. Windows est un système d'exploitation propriétaire."

Source : Wikipédia, extrait de l'article consacré à Windows.

- Expliquer succinctement les différences entre les logiciels libres et les logiciels propriétaires.
- Expliquer le rôle d'un système d'exploitation
- On donne ci-dessous une arborescence de fichiers sur un système GNU/Linux (les noms encadrés représentent des répertoires, les noms non encadrés représentent des fichiers, / correspond à la racine du système de fichiers) :



Arborescence de fichiers

Indiquer le chemin absolu du fichier `rapport.odt`.

- On suppose que le répertoire courant est le répertoire `elsa`. Indiquer le chemin relatif du fichier `photo_1.jpg`.
- L'utilisatrice Elsa ouvre une console (aussi parfois appelée terminal), le répertoire courant étant toujours le répertoire `elsa`. On fournit ci-dessous un extrait du manuel de la commande UNIX `cp` :

NOM

`cp` - copie un fichier

UTILISATION

`cp fichier_source fichier_destination`

Déterminer le contenu du répertoire `documents` et du répertoire `boulot` après avoir exécuté la commande suivante dans la console :

cp documents/fiche.ods documents/boulot

6. "Un système d'exploitation est multitâche (en anglais : multitasking) s'il permet d'exécuter, de façon apparemment simultanée, plusieurs programmes informatiques. GNU/Linux, comme tous les systèmes d'exploitation modernes, gère le multitâche." "Dans le cas de l'utilisation d'un monoprocesseur, la simultanéité apparente est le résultat de l'alternance rapide d'exécution des processus présents en mémoire."

Source : Wikipédia, extraits de l'article consacré au Multitâche.

Dans la suite de l'exercice, on s'intéresse aux processus. On considère qu'un monoprocesseur est utilisé. Un processus est soit élu, soit bloqué, soit prêt.

Recopier et compléter le schéma ci-dessous avec les termes suivants : élu, bloqué, prêt, élection, blocage, déblocage.

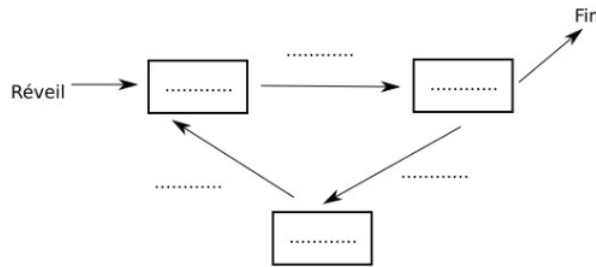


Schéma processus

7. Donner l'exemple d'une situation qui contraint un processus à passer de l'état élu à l'état bloqué.
8. Dans les systèmes d'exploitation, l'ordonnanceur est le composant du noyau du système d'exploitation choisissant l'ordre d'exécution des processus sur le processeur d'un ordinateur."

Source : Wikipédia, extrait de l'article consacré à l'ordonnancement.

L'ordonnanceur peut utiliser plusieurs types d'algorithmes pour gérer les processus. L'algorithme d'ordonnancement par "ordre de soumission" est un algorithme de type FIFO (First In First Out), il utilise donc une file.

Nommer une structure de données linéaire de type LIFO (Last In First Out).

9. À chaque processus, on associe un instant d'arrivée (instant où le processus demande l'accès au processeur pour la première fois) et une durée d'exécution (durée d'accès au processeur nécessaire pour que le processus s'exécute entièrement). Par exemple, l'exécution d'un processus P4 qui a un instant d'arrivée égal à 7 et une durée d'exécution égale à 2 peut être représentée par le schéma suivant :



Utilisation du processeur

L'ordonnanceur place les processus qui ont besoin d'un accès au processeur dans une file, en respectant leur ordre d'arrivée (le premier arrivé étant placé en tête de file). Dès qu'un processus a terminé son exécution, l'ordonnanceur donne l'accès au processus suivant dans la file. Le tableau suivant présente les instants d'arrivées et les durées d'exécution de cinq processus

| 5 processus | | |
|-------------|-------------------|-------------------|
| Processus | instant d'arrivée | durée d'exécution |
| P1 | 0 | 3 |
| P2 | 1 | 6 |
| P3 | 4 | 4 |
| P4 | 6 | 2 |
| P5 | 7 | 1 |

Utilisation du processeur

Créer le même type de schéma illustrant l'utilisation du processeur, avec les processus P1 à P5 en utilisant les informations présentes dans le tableau ci-dessus et l'algorithme d'ordonnancement "par ordre de soumission"

10. On utilise maintenant un autre algorithme d'ordonnancement : l'algorithme d'ordonnancement "par tourniquet". Dans cet algorithme, la durée d'exécution d'un processus ne peut pas dépasser une durée Q appelée quantum et fixée à l'avance. Si ce processus a besoin de plus de temps pour terminer son exécution, il doit retourner dans la file et attendre son tour pour poursuivre son exécution. Par exemple, si un processus P1 a une durée d'exécution de 3 et que la valeur de Q a été fixée à 2, P1 s'exécutera pendant deux unités de temps avant de retourner à la fin de la file pour attendre son tour ; une fois à nouveau élu, il pourra terminer de s'exécuter pendant sa troisième et dernière unité de temps d'exécution.

Recopier et compléter le schéma ci-dessous, en utilisant l'algorithme d'ordonnancement "par tourniquet" et les mêmes données que pour la question 9, en supposant que le quantum Q est fixé 2.



Utilisation du processeur

11. On considère deux processus P1 et P2, et deux ressources R1 et R2.
Décrire une situation qui conduit les deux processus P1 et P2 en situation d'interblocage.

Exercice 9.10

Considérons 4 processus notés A, B, C et D dont les temps d'exécution et leurs arrivages respectifs sont donnés :

- A, durée 5, temps d'arrivé 0
- B, durée 4, temps d'arrivé 0.5
- C, durée 1, temps d'arrivé 2
- D, durée 3, temps d'arrivé 5

Représenter pour l'algorithme du tourniquet, le schéma d'exécution. On considère ici qu'un nouveau processus, non encore exécuté, est placé à la fin de la file.

On se propose ici de présenter les résultats sous la forme suivante :

Quantum: 0

File d'attente:

| | | | | | | | | | | | | | | |
|----|----|--|--|--|--|--|--|--|--|--|--|--|--|--|
| P1 | P2 | | | | | | | | | | | | | |
|----|----|--|--|--|--|--|--|--|--|--|--|--|--|--|

Processus en mémoire P1:

| | | | | | | | | | | | | | | |
|---|--|--|--|--|--|--|--|--|--|--|--|--|--|--|
| 5 | | | | | | | | | | | | | | |
|---|--|--|--|--|--|--|--|--|--|--|--|--|--|--|

P2:

| | | | | | | | | | | | | | | |
|---|--|--|--|--|--|--|--|--|--|--|--|--|--|--|
| 4 | | | | | | | | | | | | | | |
|---|--|--|--|--|--|--|--|--|--|--|--|--|--|--|

Ordonnancement:

| | | | | | | | | | | | | | | |
|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|
| | | | | | | | | | | | | | | |
|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|

Solution :
ABABCABDABDAD