Chapitre IV : Ordonnancement des processus

Cours Système Exploitation II

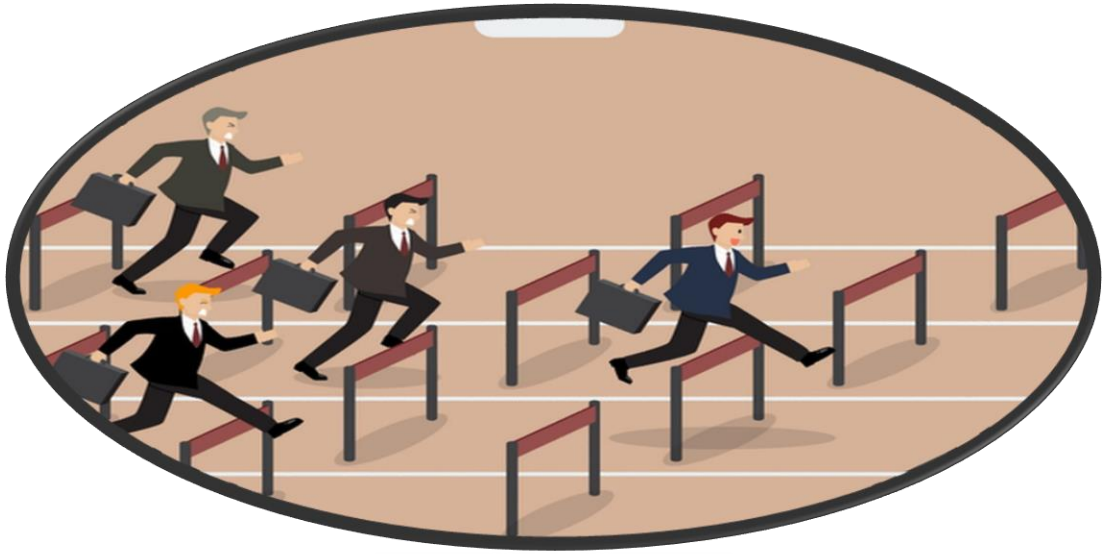
1 Linfo

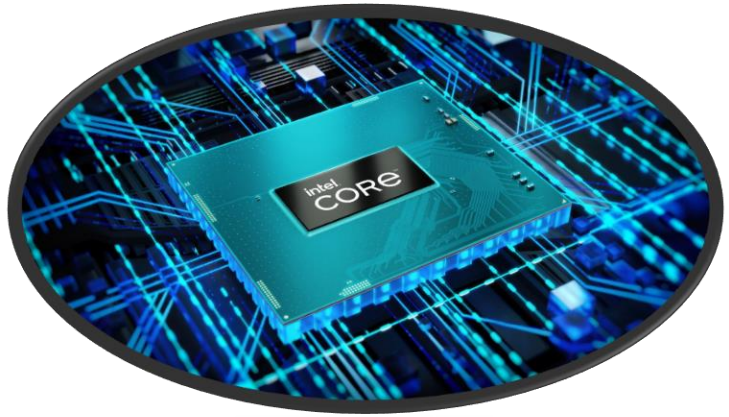
ISIMM

2022/2023

Motivations

Lorsqu'un ordinateur est multiprogrammé, il possède fréquemment plusieurs processus/threads en concurrence pour l'obtention de temps processeur.



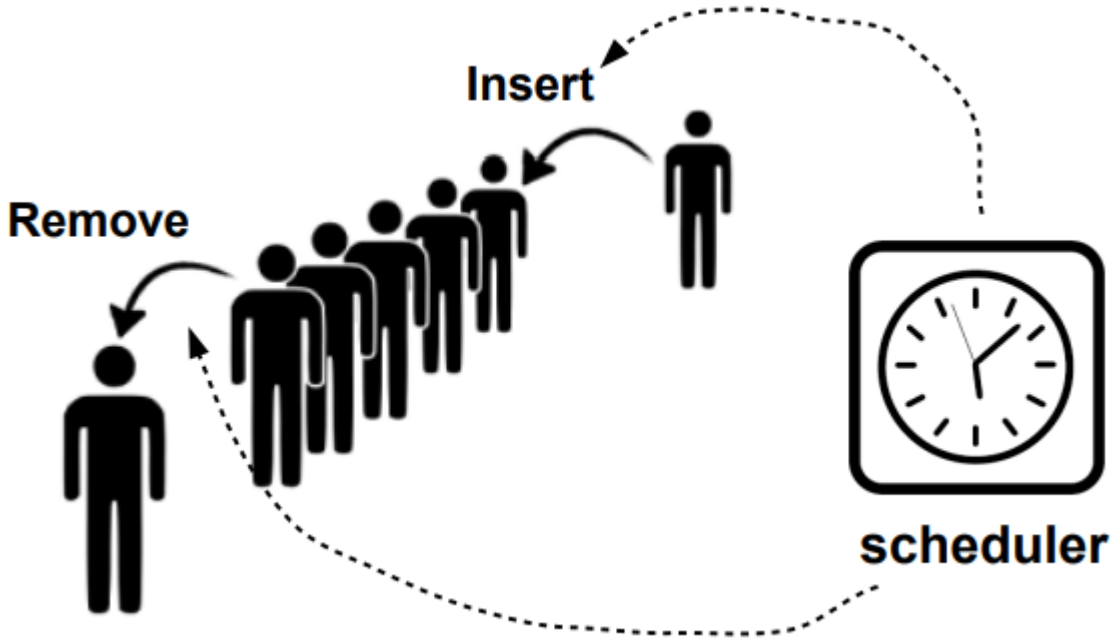


→ S'il n'y a qu'un seul processeur, un choix doit être fait quant au prochain processus à exécuter

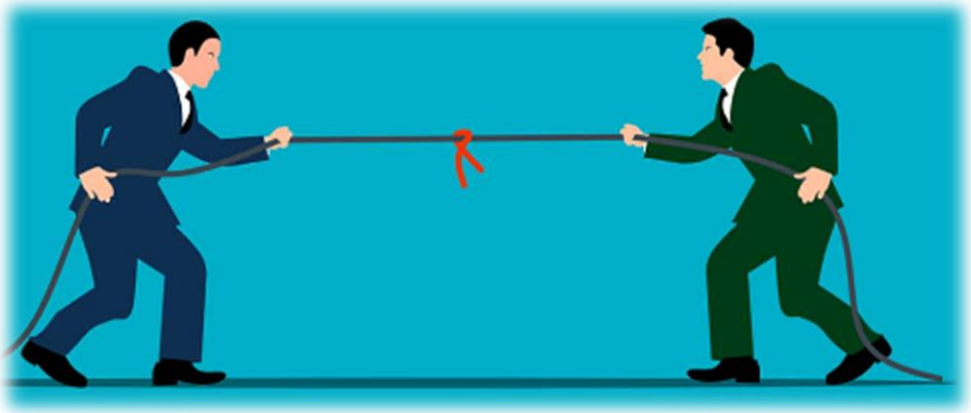
Définitions

La partie du système d'exploitation qui effectue ce choix se nomme l'ordonnanceur (scheduler) et l'algorithme qu'il emploie s'appel algorithme d'ordonnancement (scheduling algorithm).

Outre le fait de sélectionner le bon processus à exécuter, l'ordonnancement doit également se soucier de faire un usage efficace du processeur, car le passage d'un processus à l'autre sont coûteux en termes de temps de traitement



Défis

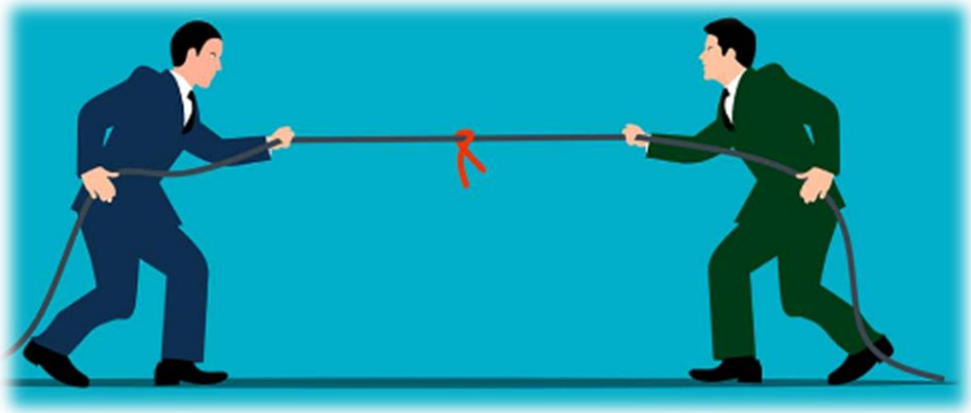


Temps de changement de contexte

Temps d’exécution

Efficacité : Utiliser au mieux le processeur

Objectifs de l'ordonnanceur



Les objectifs d'un ordonnanceur d'un système multi-utilisateur sont entre autres :

– S'assurer que chaque processus en attente d'exécution reçoive sa part de temps processeur.

– Minimiser le temps de réponse.

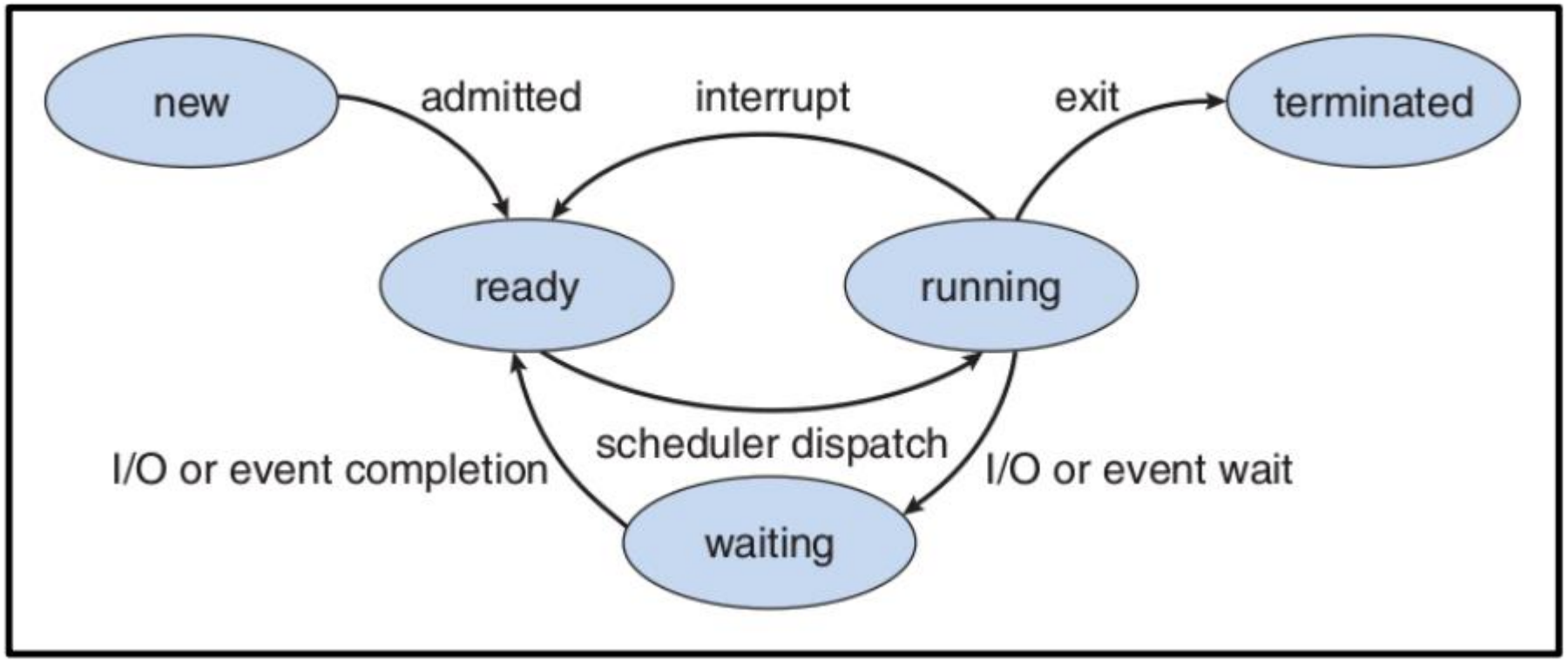
– Utiliser le processeur à 100%.

– Prendre en compte des priorités.

– Être prédictible.

États de processus et ordonnancement

Quand ordonnancer ?



États de processus et ordonnancement

Quand ordonnancer ?

- Lorsqu'un nouveau processus est créé :

→ il faut se décider s'il faut exécuter d'abord le processus parent ou le processus enfant.

- Lorsqu'un processus se termine

→ un autre processus doit être choisi parmi les processus prêts

États de processus et ordonnancement

Quand ordonnancer ?

- Lorsqu'un processus se bloque

→ un autre processus doit être sélectionné pour être exécuter

- Lorsqu'une interruption d'E/S se produit

→ il faut prendre une décision d'ordonnancement parmi les processus qui étaient bloqué en attente d'E/S

Qu’est ce qu’une interruption ?

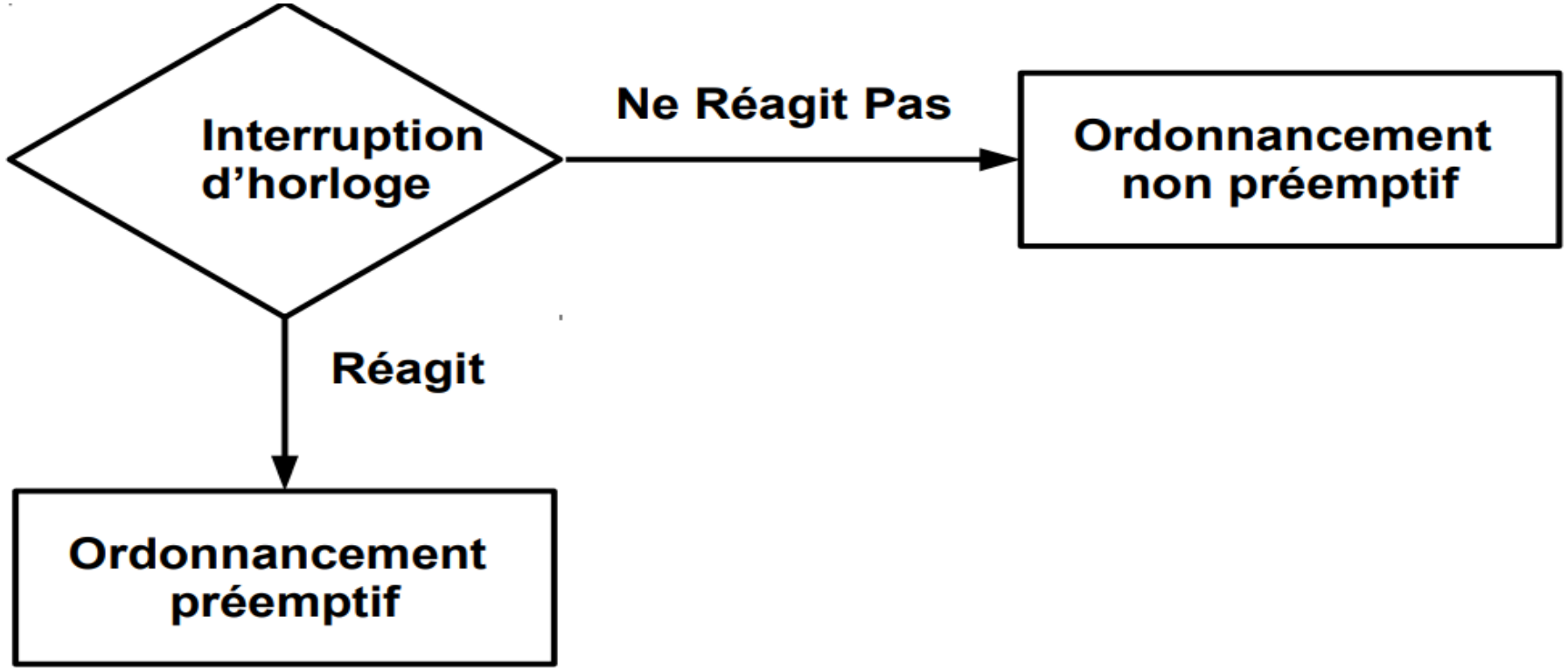
- **Interruption**
  - arrêt temporaire de l’exécution
- **Catégorie** : **Matérielle** vs. **Logicielle**
- **Interruption Logicielle : Trappe ou déroutement**
  - Si le CPU détecte une erreur dans le traitement d’une instruction (division par zéro par exemple)
  - Arrêt de l’exécution pour exécuter une routine particulière ISR (*Interrupt Service Routine*) par type d’erreur rencontrer (débordement de mémoire, division par zéro,...)

Interruption Matérielle IRQ : Interrupt Request

- **Générées par les périphériques (clavier, disque, USB,…)**
- **Peuvent être masquées (interdites ou autorisés)**
- **Interruption : le périphérique signale au CPU les événements par interruption**
  - Éviter au CPU **d'attendre un délai supplémentaire (boucler)** pour que les données soient émises par le périphérique (technique de polling ou questionnement)
- **Le CPU arrête momentanément l'exécution d'un processus pour exécuter la requête du périphérique.**
  - Déclenche l'ordonnancement entre les processus bloqué en l'attente de l'E/S en question

Classification des algorithmes d'ordonnancement

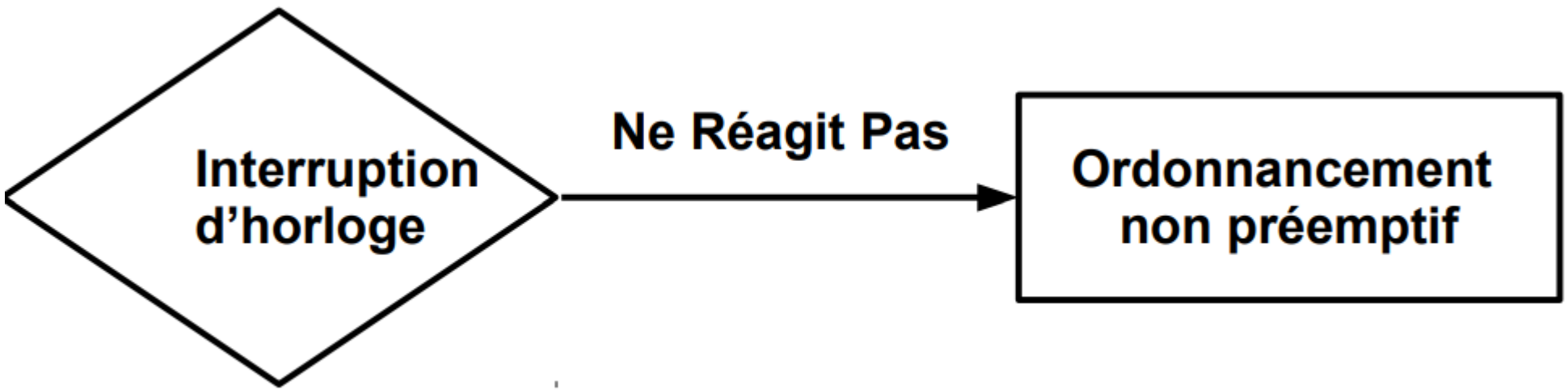
On utilise les interruptions d’horloge pour commuter les tâches dans les systèmes multitâches.



Algorithmes d'ordonnancement non préemptif

- **Sélectionne un processus, puis le laisse s'exécuter jusqu'à ce qu'il se bloque, ou qu'il libère volontairement le processeur**
  - Même s'il s'exécute pendant des heures, il ne sera pas suspendu de force
  - Aucune décision d'ordonnancement n'intervient pendant les interruptions d'horloge

Ordonnanceurs non préemptifs



Algorithmes d'ordonnancement préemptif

Sélectionne un processus et le laisse s’exécuter pendant un délai déterminé

- Si le processus est toujours en cours d’exécution à l’issue de ce délai, il est suspendu.
- L’ordonnanceur sélectionne un autre processus à exécuter.

Ordonnanceurs non préemptifs

- **Dans un système à ordonnancement non préemtif ou sans réquisition, le système d'exploitation choisit le prochain processus à exécuter:**
  - le Premier Arrivé est le Premier Servi PAPS (ou First-Come First-Served FCFS)
  - ou le plus court d'abord (Shortest Job First SJF)

- **Il lui alloue le processeur jusqu'à ce qu'il se termine ou qu'il se bloque (en attente d'un événement).**

  → Il n'y a pas de réquisition

Critères spécifiques d'ordonnancement

- **Utilisation UCT: pourcentage d'utilisation**
- **Débit = Throughput: nombre de processus qui complètent dans l'unité de temps**
- **Temps de rotation: turnaround = le temps pris par le processus depuis son arrivée jusqu'à sa terminaison**
- **Temps d'attente: attente dans la file prêt (somme de tout le temps passé en file prêt)**
- **Temps de réponse: le temps entre une demande et la réponse**

Critères spécifiques d'ordonnancement:
minimiser/maximiser

- **Utilisation UCT: pourcentage d'utilisation : à maximiser**
- **Débit = Throughput: nombre de processus qui complètent dans l'unité de temps : à maximiser**
- **Temps de rotation: turnaround = le temps pris par le processus depuis son arrivée jusqu'à sa terminaison : à minimiser**
- **Temps d'attente: attente dans la file prêt (somme de tout le temps passé en file prêt) : à minimiser**
- **Temps de réponse: le temps entre une demande et la réponse : à minimiser**

First-Come, First-Served

- Implémenter via une file d'attente FIFO → **code très facile à implémenter**
- Par contre, c'est une stratégie qui peut engendrer des **temps d'attentes moyens importants et très variables**
- Exemple : considérons les processus P1, P2, P3 avec les temps d'exécutions suivants. Ces processus arrivent tous au temps 0.

First-Come, First-Served

P1	24
P2	3
P3	3

First-Come, First-Served

- Si les processus arrivent dans l'ordre P1, P2, P3, on aura le diagramme de Gantt suivant

- Le temps d'attente de P1 est de 0ms, celui de P2 est de 24ms et enfin celui de P3 est 27ms

- Le temps d'attente moyen est de (0 +24 +27)/3 = 17ms

First-Come, First-Served

- 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

First-Come, First-Served

- Supposons que les processus arrivent dans l'odre P2, P3, P1, on aura la diagramme de Gantt suivant :

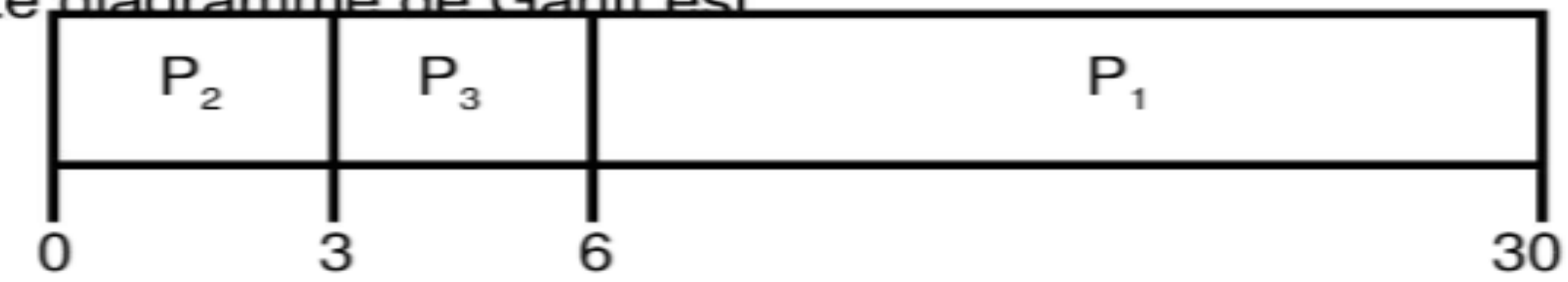
- Le temps moyen d'attente devient (0+3+6)/3 = 3ms

First-Come, First-Served

Si les mêmes processus arrivent à 0 mais dans l'ordre

P_2 , P_3 , P_1 .

Le diagramme de Gantt est:



- Temps d'attente pour $P_1 = 6$ $P_2 = 0$ $P_3 = 3$
- Temps moyen d'attente: (6 + 0 + 3)/3 = 3
- Temps de rotation moyen: (3+6+30)/3 = 13
- Beaucoup mieux!
- Donc pour cette technique, les temps peuvent varier grandement par rapport à l'ordre d'arrivée de différent processus