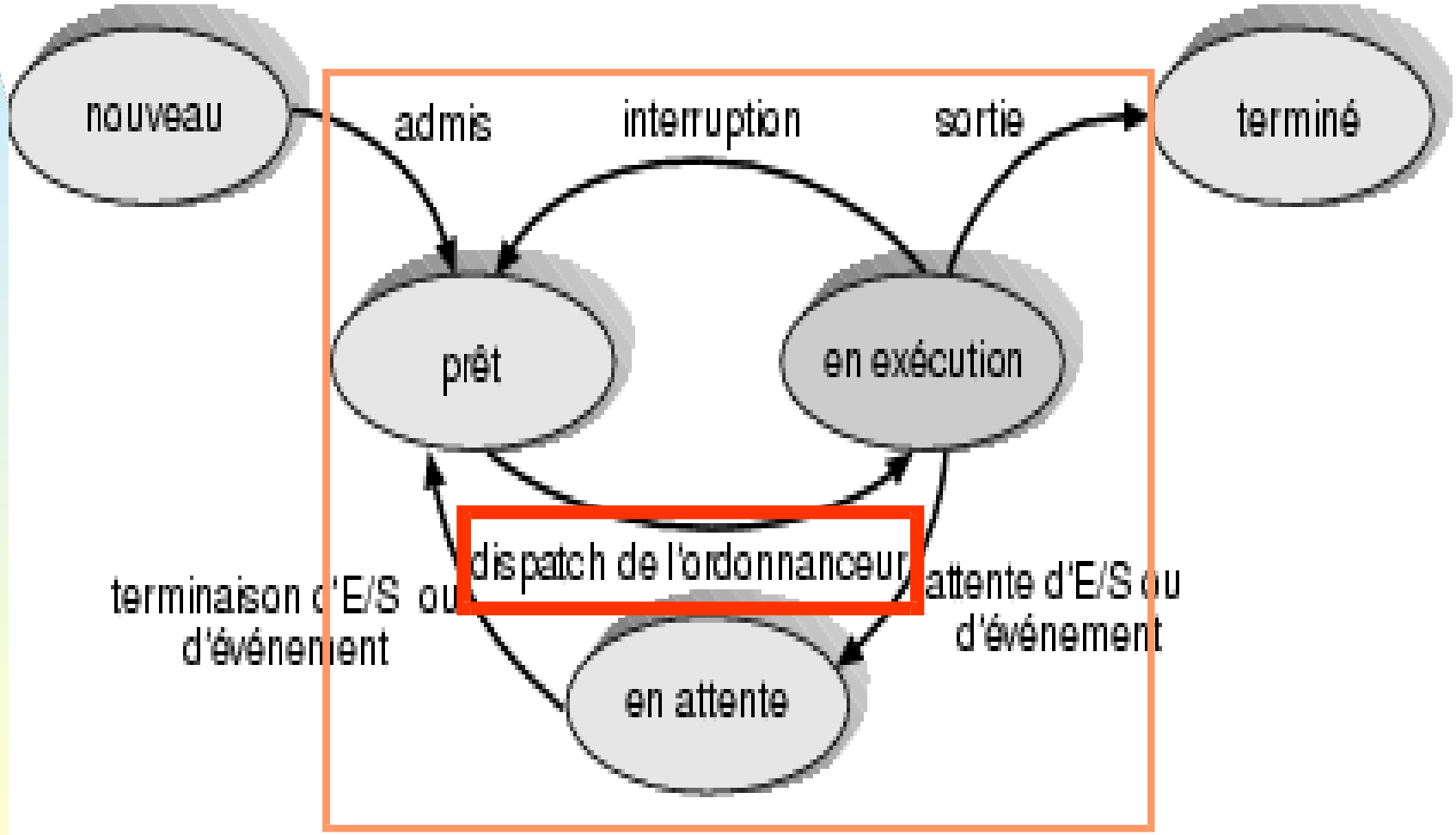
chapitre 4 - Ordonnancement Processus

Aperçu du module

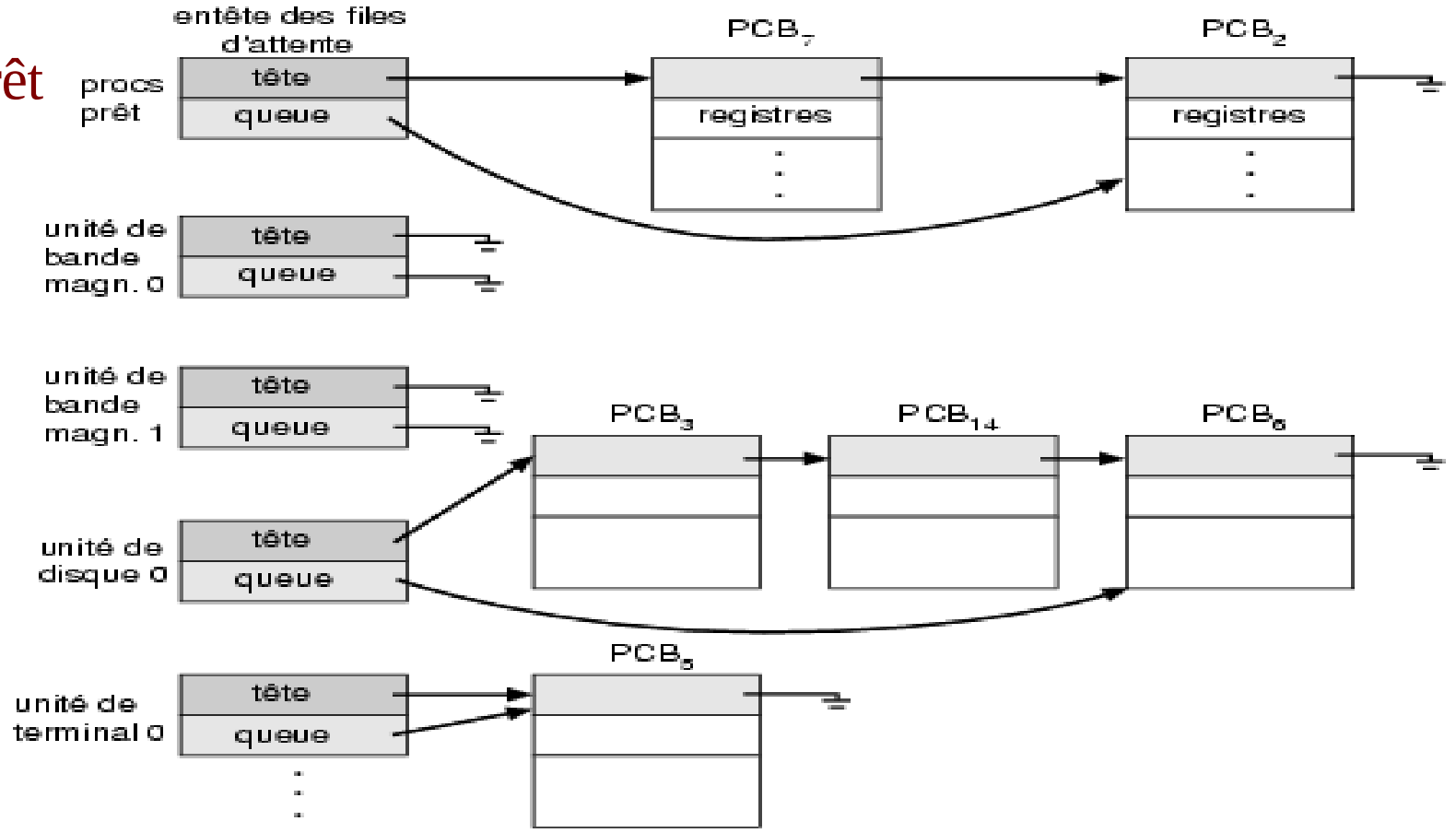
- **Concepts de base**
- **Critères d'ordonnancement**
- **Algorithmes d'ordonnancement**
- **Ordonnancement de multiprocesseurs**
- **Évaluation d'algorithmes**

Diagramme de transition d`états d`un processus



Files d'attente de processus pour ordonnancement

file prêt

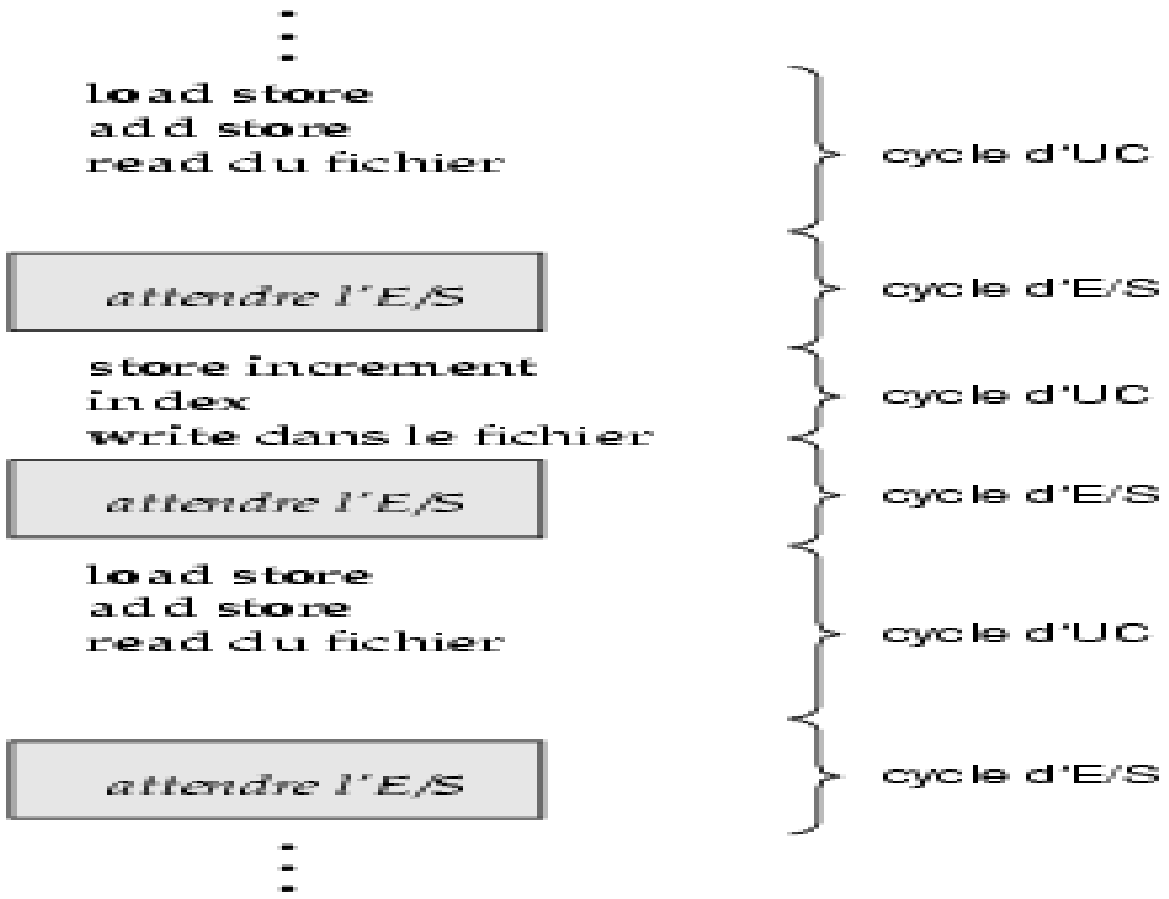


Nous ferons l'hypothèse que le premier processus dans une file est celui qui utilise la ressource: ici, proc7 exécute

Concepts de base

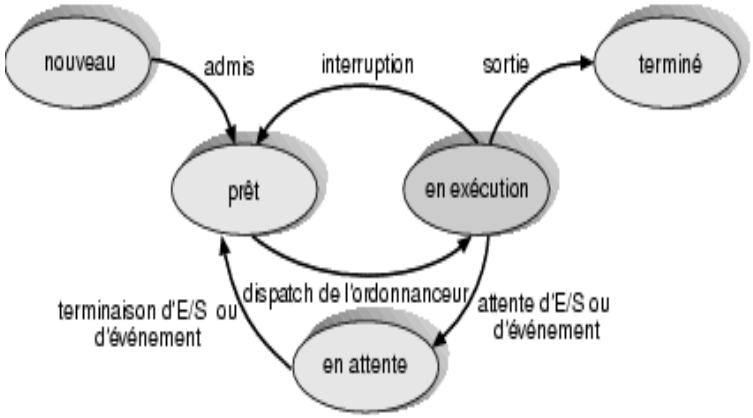
- **La multiprogrammation est conçue pour obtenir une utilisation maximale des ressources, surtout de l'UCT**
- **L`ordonnanceur UCT est la partie du SE qui décide quel processus dans la file ready/prêt obtient l 'UCT quand elle devient libre**
  - Objectif: utilisation optimale de l 'UCT
- **l 'UCT est la ressource la plus importante dans un ordinateur, donc nous parlons surtout d'elle**
  - Cependant, les principes que nous verrons s 'appliquent aussi à l'ordonnancement des autres ressources (unités E/S, etc).
- **Doit comprendre le comportement des processus**
  - Pour faire de bonne décision d'ordonnancement

Les cycles d'un processus



- **Cycles/activités (bursts) de l'UCT et E/S: l'exécution d'un processus consiste de séquences d'exécution sur l'UCT et d'attentes E/S**

Quand invoquer l'ordonnanceur



- **L 'ordonnanceur doit prendre sa décision chaque fois que le processus exécutant est interrompu, c-à-d:**
  - un processus est créé (nouveau) ou se termine ou
  - un processus exécutant devient bloqué en attente
  - un processus change d'exécutant/running à prêt/ready
  - un processus change de attente à prêt/read
  - en conclusion, tout événement dans un système cause une interruption de l'UCT et l'intervention de l'ordonnanceur, qui devra prendre une décision concernant quel processus ou fil aura l'UCT par la suite
- **Préférentiel: on a préférence dans les derniers deux cas si on enlève l'UCT à un processus qui l'avait et peut continuer à s'en servir**
- **Dans les 1ers deux cas, il n'y a pas de préférence**
- **Plusieurs problèmes à résoudre dans le cas préférentiel**

Dispatcheur

- **Le module passe le contrôle de l'UCT au processus choisi par l'ordonnanceur à court terme; cela implique:**
  - changer de contexte
  - changer au mode usager
  - réamorcer le processus choisi
- **Temps de réponse de requête de processus (dispatcher latency)**
  - Le temps nécessaire au dispatcher d'arrêter un processus et de démarrer un autre
  - Souvent négligeable, il faut supposer qu'il soit petit par rapport à la longueur d'un cycle

Critères d'ordonnancement

- **Il y aura normalement plusieurs processus dans la file prêt**
- **Quand l'UCT devient disponible, lequel choisir?**
- **L'idée générale est d'effectuer un choix pour optimiser l'utilisation de la machine**
- **Mais cette dernière peut être jugée selon différents critères…**

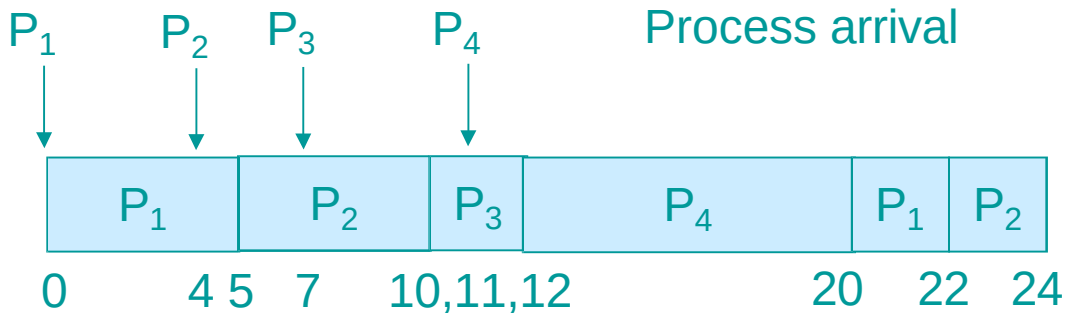
Critères d'ordonnancement

- **Raison principale pour l'ordonnancement**
  - Pourcentage d'utilisation: garder l'UCT et les modules E/S occupés aussi longtemps que possible
- **Systèmes à temps partagés?**
  - Temps de réponse (pour les systèmes interactifs): le temps entre une demande et la réponse
- **Serveurs?**
  - Débit (Throughput): nombre de processus qui achèvent leur exécution par unité de temps
- **Systèmes de traitement par lots (batch)?**
  - Temps de rotation (turnaround): le temps entre la soumission et la complétion d'un processus.
- **Systèmes chargés?**
  - Temps d'attente: le temps que passe un processus dans la file prêt

Critères d'ordonnancement: maximiser/minimiser

- **À maximiser**
  - Utilisation de l'UCT
  - Débit
- **À minimiser**
  - Temps de réponse
  - Temps de rotation
  - Temps d'attente

Exemple de mesure des critères d'ordonnancement



- **Utilisation de l'UCT:**
  - 100%
- **Temps de réponse (P_3, P_2):**
  - P_3: 3 = (10 – 7)
  - P_2: 1 = (5 – 4)
- **Débit :**
  - 4/24
- **Temps de rotation (P_3, P_2):**
  - P_3: 5 = (12 – 7)
  - P_2: 20 = (24 – 4)
- **Temps d'attente (P_2):**
  - P_2: 13 = (5 – 4) + (22 – 10)

Premier arrivé, premier servi (FCFS)

- Notez, aucune préemption

Exemple: Processus	Temps de cycle	Temps d'arrivée
P1	24	0 (premier)
P2	3	0 (second)
P3	3	0 (troisième)

Les processus arrivent au temps 0 dans l'ordre: P1 , P2 , P3

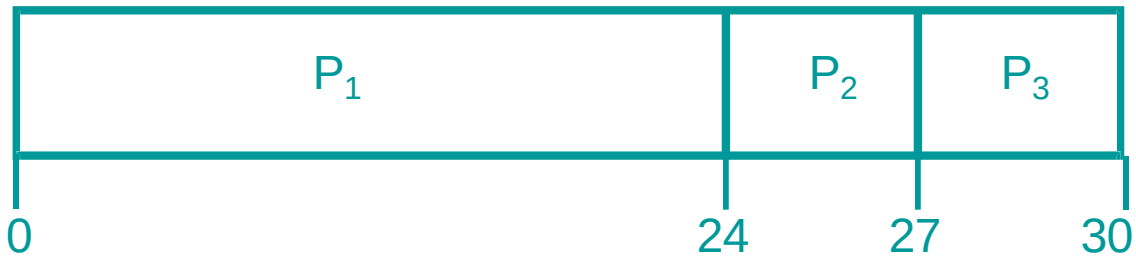
Le diagramme Gantt est:

Temps d'attente pour P1= 0; P2= 24; P3= 27

Temps attente moyen: (0 + 24 + 27)/3 = 17

Premier arrivé, premier servi

- **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**



Ordonnancement FCFS (suite)

Si les mêmes processus arrivaient 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**
- **Beaucoup mieux!**
- **Donc pour cette technique, le temps d'attente moyen peut varier grandement**
- ***Exercice: calculer aussi le temps moyen de rotation, débit, etc.***

Tenir compte du temps d'arrivée!

- **Dans le cas où les processus arrivent à des moments différents, il faut soustraire les temps d'arrivées**
- **Exercice: répéter les calculs si:**
  - P2 arrive à temps 0
  - P1 arrive à temps 2
  - P3 arrive à temps 5

Plus Court Job d'abord = Shortest Job First (SJF)

- **Le processus le plus court part le premier**
- **Optimal en principe du point de vue du temps d'attente moyen**
  - (voir le dernier exemple)
- **Mais comment savons-nous**

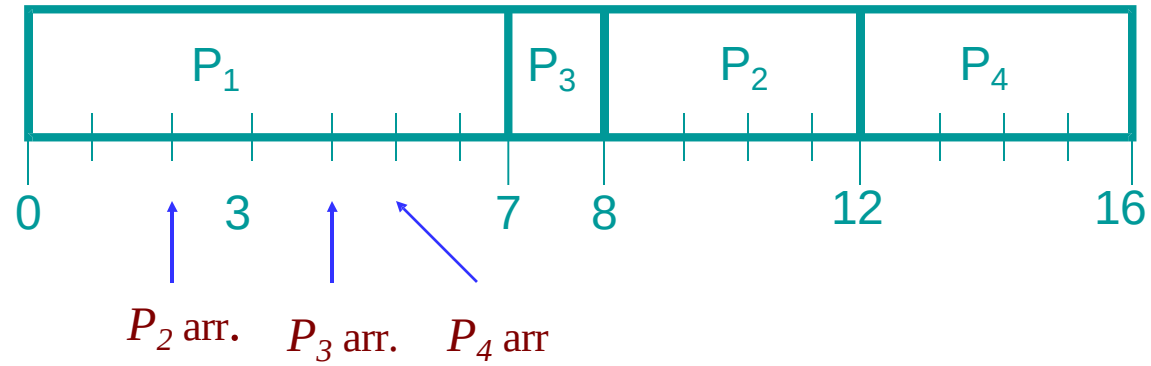
SJF avec préemption ou non

- **Avec préemption: si un processus qui dure moins que le *restant* du processus courant se présente plus tard, l'UCT est donnée à ce nouveau processus**
  - SRTF: shortest remaining-time first
- **Sans préemption: on permet au processus courant de terminer son cycle**
  - Observation: SRTF est plus logique car de toute façon le processus exécutant sera interrompu par l'arrivée du nouveau processus
    - Il est retourné à l'état prêt

Exemple de SJF sans préemption

Processus	Arrivée	Cycle
P_1	0	7
P_2	2	4
P_3	4	1
P_4	5	4

- SJF (sans préemption)

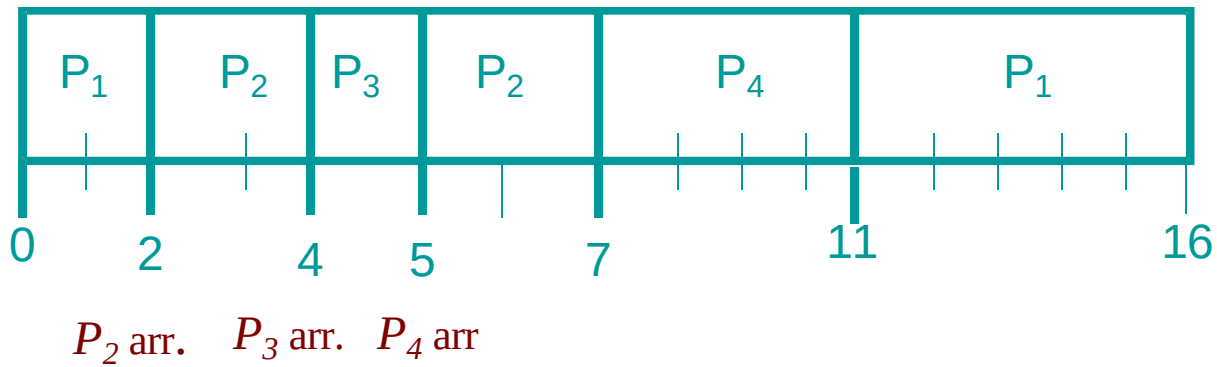


- Temps d'attente moyen = (0 + 6 + 3 + 7)/4 = 4

Exemple de SJF avec préemption

Processus	Arrivée	Cycle
P_1	0	7
P_2	2	4
P_3	4	1
P_4	5	4

- SJF (préemptive)



- Temps moyen d`attente = (9 + 1 + 0 +2)/4 = 3
  - P1 attend de 2 à 11, P2 de 4 à 5, P4 de 5 à 7

Comment déterminer la longueur des cycles à l'avance?

- **Quelques méthodes proposent de déterminer le comportement futur d'un processus sur la base de son passé**
  - ex. moyenne exponentielle

Le plus court d'abord SJF: critique

- **Difficulté d'estimer la longueur à l'avance**
- **Les processus longs souffriront de *famine* lorsqu'il y a un apport constant de processus courts**
- **La préemption est nécessaire pour environnements à temps partagé**
  - Un processus long peut monopoliser l'UCT s'il est le 1er à entrer dans le système et il ne fait pas d'E/S
- **Il y a assignation implicite de priorités: préférences aux travaux plus courts**

Priorités

- **Affectation d'une priorité à chaque processus (par ex. un nombre entier)**
  - souvent les petits chiffres dénotent des hautes priorités (dans UNIX)
    - 0 la plus haute
  - Windows fait l'inverse – donne une plus haute priorité aux plus grands chiffres
- **L'UCT est donnée au processus prêt avec la plus haute priorité**
  - avec ou sans préemption
  - il y a une file *prêt* pour chaque priorité
- **Priorités sont explicites**
  - Pour raisons politiques ou techniques
- **Priorités implicites**
  - Voir SJF - critiques

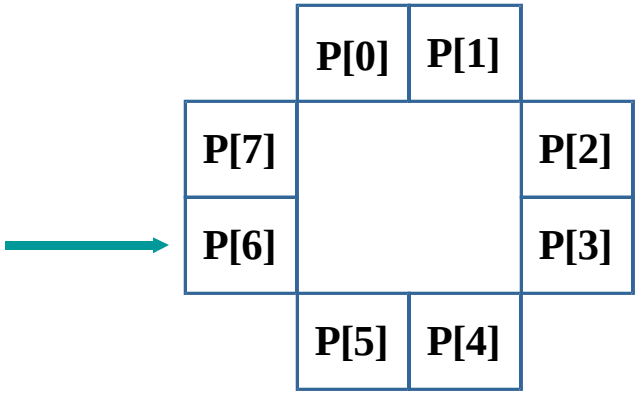
Problème possible avec les priorités

- **Famine: les processus moins prioritaires n'arrivent jamais à exécuter**
- **Solution: vieillissement:**
  - modifier la priorité d'un processus en fonction de son âge et de son historique d'exécution
  - le processus change de file d`attente
- **Généralement, la modification dynamique des priorités est une politique souvent utilisée (files à rétroaction ou retour)**
- **Que faire avec les processus de même priorités?**
  - FCFS
  - Ajoutons la préemption -> le Tourniquet

Tourniquet = Round-Robin (RR)

Le plus utilisé en pratique

- **Chaque processus est alloué un intervalle de temps de l'UCT (ex. 10 à 100 millisecs.) pour s'exécuter**
  - (terminologie du livre: *tranche de temps*)
- **Après que ce temps s'est écoulé, le processus est interrompu, mis à la fin de la queue *prêt* et l 'UCT est donnée au processus en tête de la queue**
- **Méthode préemptive**



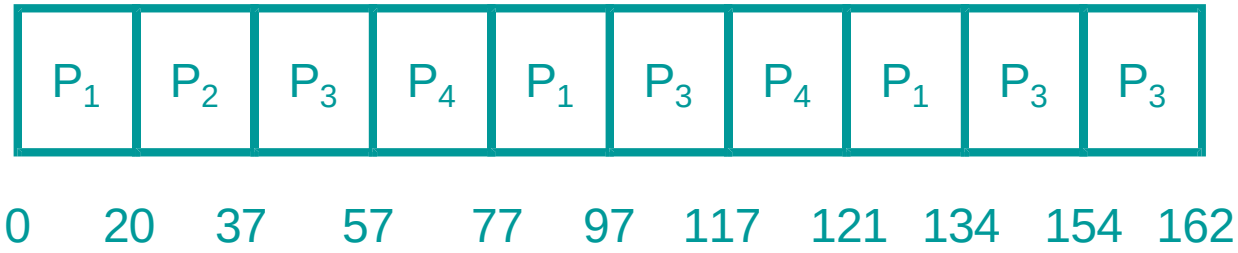
La file prêt est un cercle (RR)

Performance du tourniquet

- **S'il y a n processus dans la file *prêt* et la tranche de temps est q, alors chaque processus reçoit 1/n du temps d'UCT en unités de durée maximale q**
- **Si q est grand ⇒ FCFS**
- **Si q est petit... à voir**

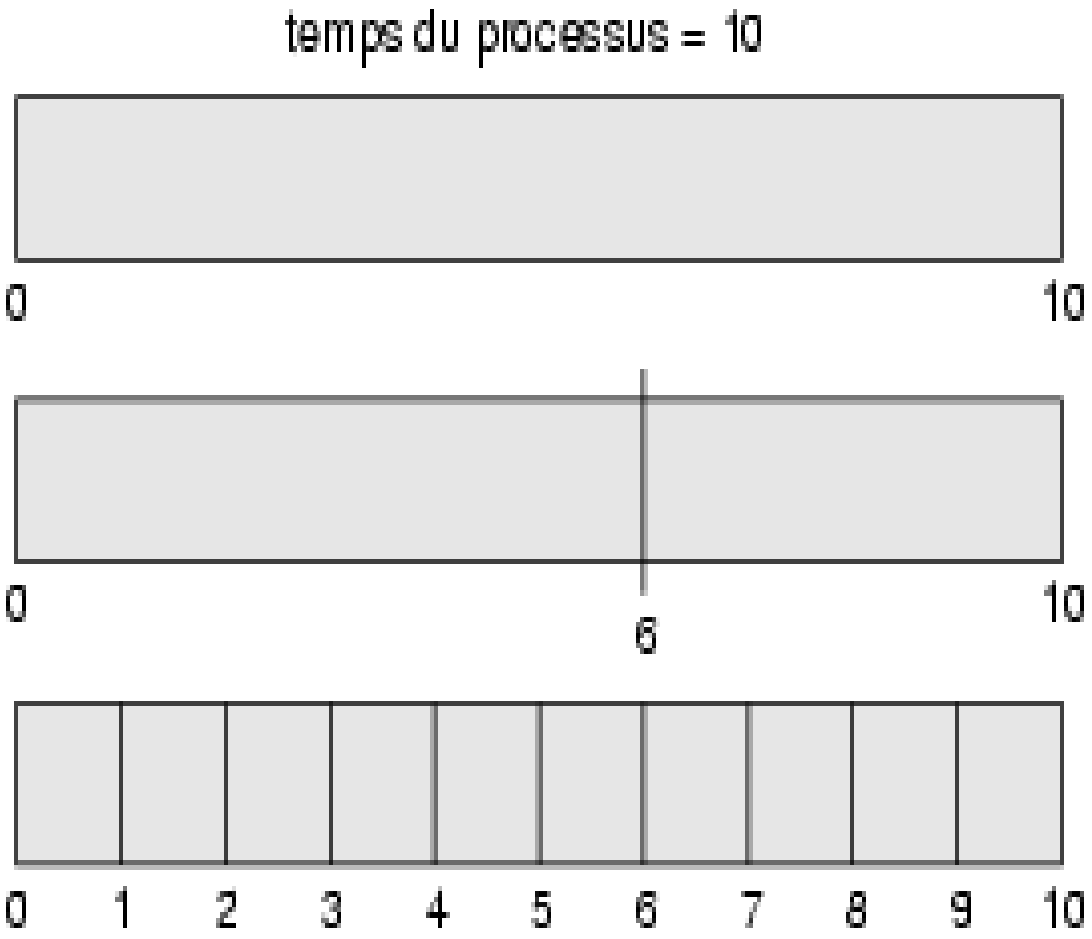
Exemple: Tourniquet Quantum = 20

Processus	Cycle
P_1	53
P_2	17
P_3	68
P_4	24



- **Normalement,**
  - temps de rotation (turnaround) plus élevé que SJF
  - mais temps d'attente moyen meilleur

Un petit intervalle augmente les commutations de contexte (temps de SE)



tranches de temps	changement de contexte
12	0
6	1
1	9

Résumé des algorithmes d'ordonnancement

- **Premier arrivé, premier servi (FIFO)**
  - simple, court temps de système (over Head), de faibles qualités
- **Plus court d'abords (SJF)**
  - Doit savoir les temps de traitements (pas pratique)
  - Doit prédire en utilisant la moyenne exponentielle du passé
- **Ordonnancement avec priorité**
  - C'est une classe d'algorithmes
- **Tourniquet**
  - FIFO avec préemption

Survol des sujets avancés de l'ordonnancement

- **L'ordonnancement avec plusieurs UCTs identiques**
- **Modèle d'évaluation**

Ordonnancement avec plusieurs UCTs identiques: *homogénéité*

- **Une seule liste *prêt* pour toutes les UCTs (division du travail = load sharing)**
  - ☞ une liste séparée pour chaque UCT ne permettrait pas cela
  - méthodes symétriques: chaque UCT peut exécuter l ’ordonnancement et la répartition
  - méthodes asymétriques: ces fonctions sont réservées à une seule UCT

Points importants dans ce chapitre

- **Queues d'attente pour l'UCT**
- **Critères d'ordonnancement**
- **Algorithmes d'ordonnancement**
  - FCFS: simple, non optimal
  - SJF: optimal, implémentation difficile
    - Procédé de moyenne exponentielle
  - Priorités
  - Tourniquet: sélection de la tranche de temps
- **Évaluation des méthodes, théorie des files,**
  - formule de Little