

Université de Ziguinchor

Cours de Système d'Exploitation

Licence 3 Informatique

Chapitre 3: Les stratégies d'ordonnancement des processus

Youssou FAYE
Département d'Informatique

Année 2018-2019

Sommaire

- ① Les critères d'ordonnancement
- ② Les types d'ordonnancement
- ③ Les algorithmes d'ordonnancement

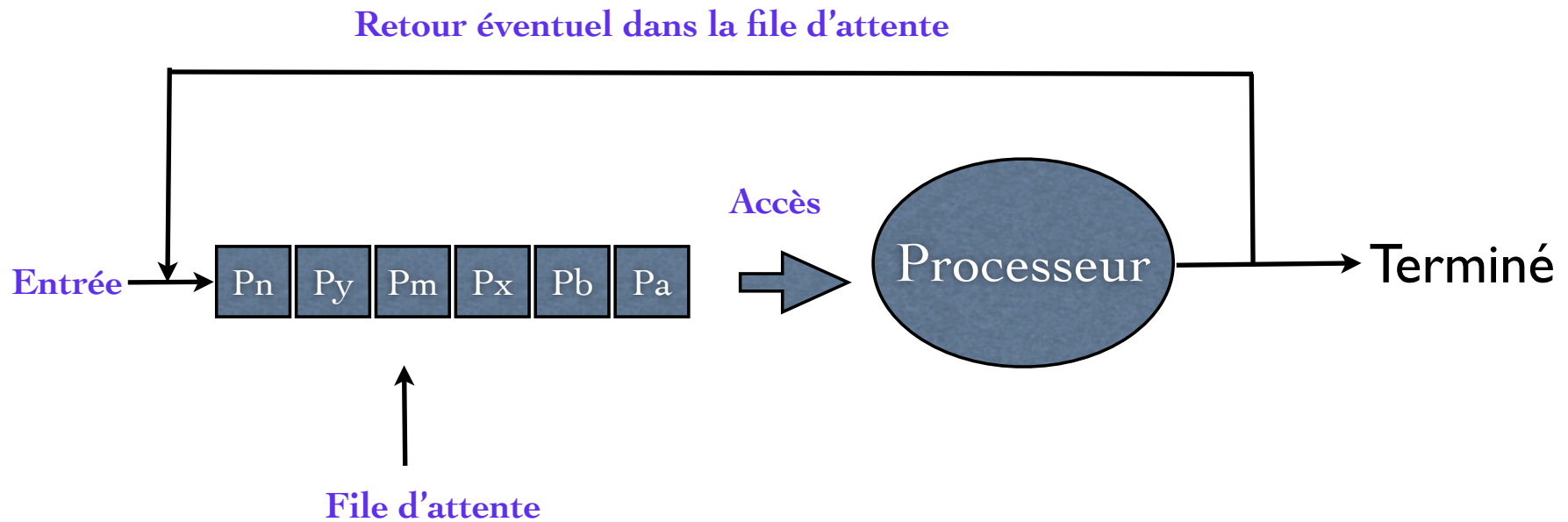
Introduction (1)

- ▶ Le SE gère plusieurs processus
- ▶ L'efficacité théorique serait maximale si le nombre de processeurs était comparable à celui des processus
- ▶ Souvent une machine possède un seul processeur
- ▶ En multiprogrammation, on a plusieurs processus en cours d'exécution
- ▶ Problème: plusieurs processus se partagent un seul processeur
- ▶ Solution: définir une politique d'accès au processeur

Introduction (2)

- ▶ L'ordonnancement définit des critères selon lesquels les processus ont accès au processeur.
- ▶ Exemples:
 - ▶ Un carrefour routier
 - ▶ Guichets d'une administration
- ▶ Critère d'ordonnancement
 - ▶ Optimiser l'utilisation de l'UCT
 - ▶ Besoins et priorités des applications

Introduction (3)



► Schéma d'ordonnancement

Optimisation des ressources

- ▶ Maximiser le pourcentage d'utilisation du processeur
- ▶ Maximiser le débit (nombre de processus exécutés/unités de temps)
- ▶ Minimiser le temps de rotation: temps de terminaison - temps d'arrivée (attente incluses)
- ▶ Minimiser le temps d'attente (temps passé dans la file d'attente)
- ▶ Minimiser le temps de réponse (pour les système interactif): temps entre une demande et première réponse

Priorité des utilisateurs

- ▶ **Ordre d'arrivée:** le premier arrivé est le premier servi (caisses au RestoU, au supermarché, à la poste etc.)
- ▶ **Urgence :** le premier servi est celui dont le besoin d'accès rapide à la ressource est le plus grand (pompier)
- ▶ **Importance :** le premier servi est celui dont l'accès à la ressource est le plus important (personne âgée dans les transports en commun)

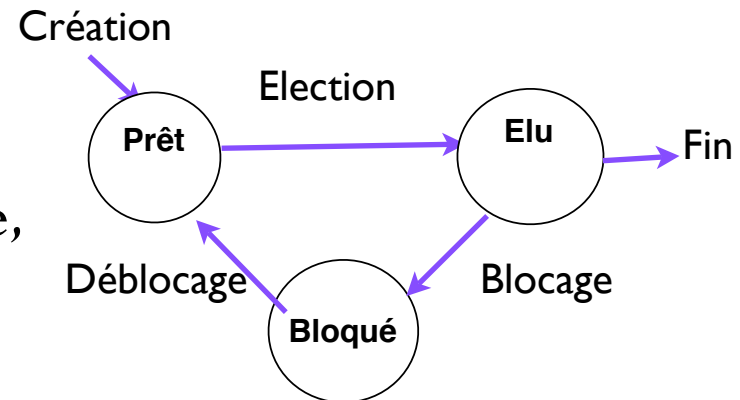
Pénalisation

- ▶ La pénalité d'un processus est liée à son temps d'attente, c'est le nombre d'unités de temps durant lesquelles le processus est présent dans la file d'attente (sans être exécuté)
- ▶ Mesurer de la pénalité (retard T) en fonction du temps d'attente et du temps d'exécution.
- ▶ $T = \frac{e}{e + a}$
 - ▶ e = temps d'exécution processus
 - ▶ e+a= temps d'exécution+ temps d'attente
- ▶ Dans le cas idéal $T=1=e/e$, donc $a=0$

Types d'ordonnancement

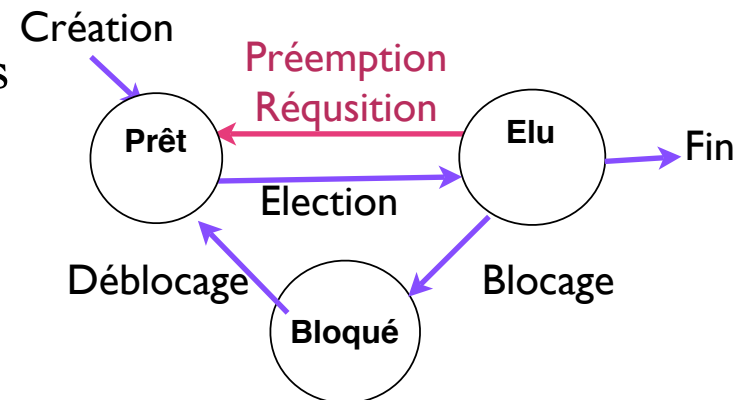
► Ordonnancement non préemptif

(sans réquisition): le processus à l'état élu est traité jusqu'à sa fin quelque soit sa durée. L'importance, l'urgence etc. ne sont pas prise en compte



► Ordonnancement préemptif (avec réquisition)

(avec réquisition): en fonction des critères d'ordonnancement, le SE remet le processus en file d'attente avant la fin de son exécution



FIFO (First In, First Out)

- ▶ Le traitement est séquentiel
- ▶ Premier arrivé en file d'attente, premier servi
- ▶ Les processus courts sont pénalisés
 - ▶ Exemple: photocopieuse utilisée par
 - ▶ Une personne qui photocopie un livre
 - ▶ Une personne qui photocopie une page

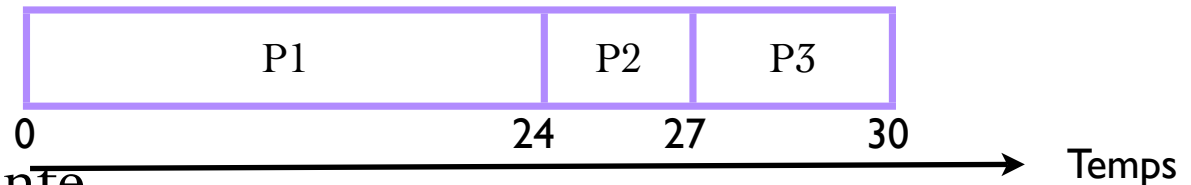
FIFO (First In, First Out)

▶ Exemple 1

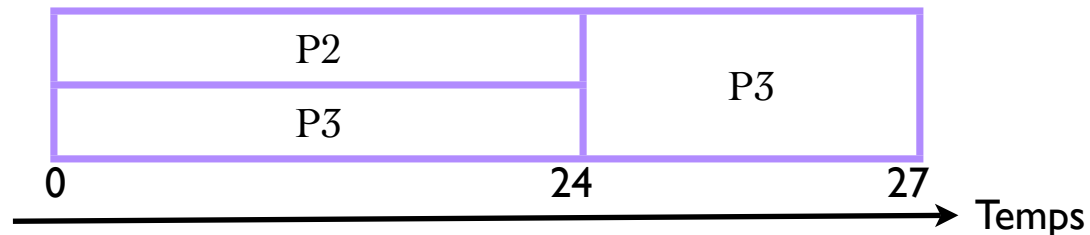
Processus	P1	P2	P3
Durée d'exécution	24	3	3

▶ Si les processus arrivent au temps 0 dans l'ordre: P₁, P₂, P₃

▶ Le schéma d'exécution



▶ Le schéma d'attente



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

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

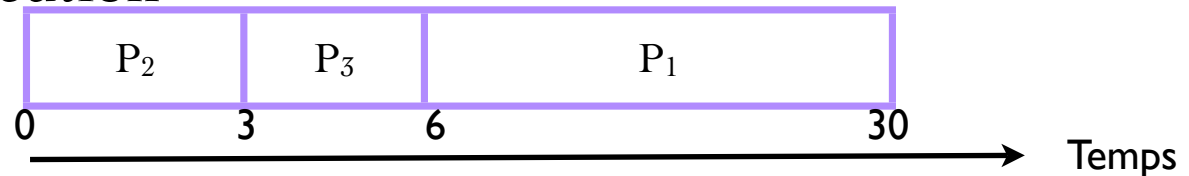
FIFO (First In, First Out)

Exemple 2

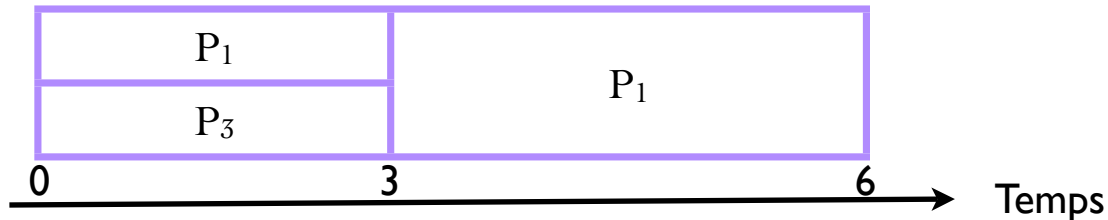
Processus	P1	P2	P3
Durée d'exécution	24	3	3

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

Le schéma d'exécution



Le schéma d'attente



Temps d'attente pour P₁ = 6 P₂ = 0 P₃ = 3

Temps moyen d'attente: $(6 + 0 + 3)/3 = 3$

Temps d'attente est
 fonction de l'ordre
 d'arrivée

SJF (Shortest Job First)

- ▶ **Shortest JOB First:** Plus court travail
- ▶ Il s'agit d'une évolution de la stratégie précédente
- ▶ La file d'attente est ordonnée non plus de façon chronologique mais en fonction du temps d'exécution nécessaire (on fait passer en tête les travaux courts)
- ▶ Un processus qui arrive est classé dans la file d'attente en fonction de sa durée
- ▶ Les travaux longs sont pénalisés, au pire ils risquent de ne jamais être exécutés tant que de petits travaux sont soumis au système
- ▶ En cas d'égalité de temps d'exécution, on applique FIFO

SRTF (Shortest Remaining-Time First)

- ▶ Favorise le temps restant le plus court
- ▶ Si un processus qui dure moins que le restant du processus courant se présente plus tard, l'UCT est enlevée au processus courant et donnée à ce nouveau processus
- ▶ Favorise les travaux courts

Algorithmes préemptifs: Tourniquet ou Round Robin(1)

- ▶ Le SE attribue à chaque processus un temps (quantum de temps) pendant lequel il est autorisé à s'exécuter
- ▶ l'UCT est enlevée au processus courant P_x si le quantum s'achève avant la fin du processus et est donnée au premier processus P_y de la file d'attente
- ▶ Le processus P_x se trouve ainsi en queue de liste
- ▶ La gestion de la file d'attente est faite selon FIFO :
- ▶ Les travaux assez courts sont vite servis
- ▶ Le tourniquet garanti que les travaux longs sortiront du système au bout d'un temps fini

Algorithmes préemptifs: Tourniquet ou Round Robin (2)

- ▶ L'efficacité du système dépend de la valeur du quantum
 - ▶ Si le changement de contexte s'opère en c ms
 - ▶ Et le quantum fixé à q ms
 - ▶ La surcharge introduite par le système est: $\frac{c}{c + q}$
 - ▶ Un quantum trop petit provoque trop de changement de contexte et augmente la surcharge, ralentit la machine
 - ▶ Trop long, la réactivité du système diminue, les petits travaux sont pénalisés

Priorité

- ▶ Affectation d'une priorité à chaque processus (p.ex. un nombre entier)
- ▶ souvent les petits chiffres dénotent des hautes priorités
- ▶ Exemple: 0 la plus haute
- ▶ L'UCT est donnée au processus prêt avec la plus haute priorité
avec ou **sans** préemption
- ▶ A priorité égale, on applique FIFO

Priorité

- ▶ Problèmes possibles:
 - ▶ Famine: les processus moins prioritaires n'arrivent jamais à exécuter
- ▶ Solution d'ajustement dynamique de la priorité
 - ▶ Modifier la priorité d'un processus en fonction de son âge et de son historique d'exécution
 - ▶ Augmenter la priorité d'un processus pénalisé
 - ▶ Diminuer la priorité d'un processus en fonction du temps passé dans le CPU