Chapitre 2 : Ordonnancement de Processus

Dr Mandicou BA
mandicou.ba@esp.sn
http://www.mandicouba.net

Diplôme D'Ingénieur de Conception (DIC, 1^{e} année)
en Informatique / Télécommunications-Réseaux
Licence Professionnelle
en Génie Logiciel et Systèmes d'Information (GLSI)

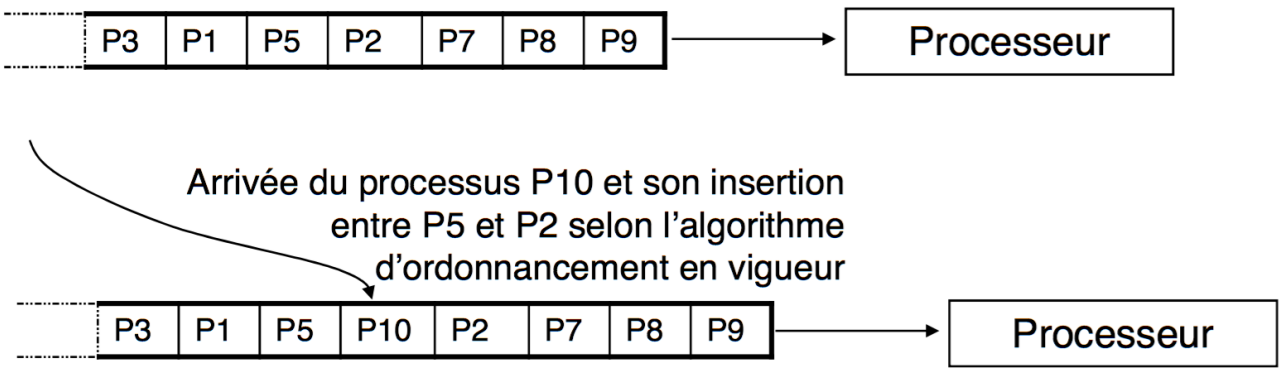
ECOLE SUPERIEURE POLYTECHNIQUE
www.esp.sn

Plan du Chapitre

Sommaire

Principe de l'ordonnancement

- Dans un système multi-tâches, plusieurs processus sont en cours simultanément
- Or, le processeur ne peut, à un moment donné, exécuter qu'une instruction (d'un programme, donc d'un processus) à la fois.
- Le processeur travaille donc en **temps partagé**.
- L' **ordonnanceur** (*scheduler* ou *dispatcher*) est le module du SE qui s'occupe de sélectionner le processus suivant à exécuter parmi ceux qui sont prêts.



Principe de l'ordonnancement

- Un processus passe entre les diverses files d'attente pendant sa durée de vie (file d'attente des processus prêt (attendant pour s'exécuter), fille d'attentes des périphériques, etc.).
- Le SE doit sélectionner les processus à partir de ces files d'attente d'une manière quelconque.
- Le processus de sélection est mené à bien par le scheduleur approprié.
- Classification des processus :
  - **Tributaire des E/S** : effectue la plupart du temps des E/S plutôt que des calculs
  - **Tributaire de la CPU** : génère peu fréquemment des requêtes d'E/S, i.e., passe plus de temps à effectuer des calculs

Principe de l'ordonnancement

- **L'ordonnancement à long terme** (ou scheduleur de travaux) :
  - Sélectionne le processus qui doit aller dans la fille de processus prêts
  - S'exécute moins fréquemment : peut être lent (seconds, minutes)
  - Contrôle le degré de multiprogrammation (le nombre de processus dans la mémoire)
  - Il est important que l'ordonnanceur à long terme réalise un bon mélange de processus tributaires de la CPU et tributaires des E/S

- **L'ordonnancement à court terme** (ou scheduleur de la CPU):
  - Choisit parmi les processus prêts celui à être exécuté (alloue la CPU à lui).
  - Appelé assez fréquemment : doit être très rapide

Principe de l'ordonnancement

- ☛ Possibilité d'introduire un **ordonnancement à moyen terme**
  - ➠ Idée clé : Il peut être avantageux de supprimer des processus de la mémoire et réduire ainsi le degré de multiprogrammation
  - ➠ Plus tard, un processus peut être réintroduit dans la mémoire et son exécution peut reprendre là ou elle s'était arrêtée
  - ➠ Ce schéma est appelé **swapping** (transfert des informations de la mémoire principale à la mémoire auxiliaire et vice-versa)

Classification des algorithmes d'ordonnancement

- Deux catégories d'algorithmes d'ordonnancement :
  1. **Non préemptif** :
     - Sélectionne un processus, puis le laisse s'exécuter jusqu'à ce qu'il bloque (soit sur une E/S, soit en attente d'un autre processus) où 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 de l'horloge.
  2. **Préemptif** :
     - Sélectionne un processus et le laisse s'exécute pendant un délai déterminé.
     - Si le processus est toujours en cours à l'issue de ce délais, il est suspendu et un autre processus est sélectionné.

Critères d'ordonnancement

- Équité : s'assurer que chaque processus reçoit sa part du temps CPU
- Efficacité : utilisation de la CPU : occupé la CPU à 100%
- Capacité de traitement (Throughput) : nombre de processus terminés par unité de temps.
- Temps d'attente : quantité de temps qu'un processus passe à attendre dans la file d'attente des processus prêts.
- Temps de restitution (Turnaround time) : temps nécessaire pour l'exécution d'un processus :
  - **Temps de restitution** = temps passé à attendre d'entrer dans la mémoire + temps passé à la file d'attente des processus prêts + temps passé é exécuter sur la CPU + temps passé à effectuer des E/S
- Temps de réponse : temps écoulé à partir du moment où on soumet une requête jusqu'à l'arrivée de la première réponse

Algorithmes d'ordonnancement

- Ordonnancement non préemptif (sans réquisition) : un processus élu reste élu tant qu'il n'est pas bloqué ou terminé
  1. FCFS (FIFO) : First-Come, First-Served
  2. SJF : Shortest Job First
  3. Priorités simples :
- Ordonnancement préemptif (avec réquisition) : possibilité d'interruption d'un processus élu. Ceci se produit dans le cas où la durée maximale d'exécution est atteinte, arrivée d'un processus plus prioritaire, etc.
  1. SRT : Shortest Remaining Time
  2. Round-Robbin
  3. Multi-niveaux

Sommaire

FCFS: First-Fome, First-Served

Scheduling du premier arrivé, premier servi

- L'implantation FCFS est facilement gérée avec une file d'attente FIFO
- Une fois que la CPU a été allouée à un processus, celui-ci garde la CPU jusqu'à ce qu'il la libère (fin ou E/S).
- Implémentation facile, pas de famine
- Incommode pour le temps partagé où il est important que chaque utilisateur obtienne la CPU à des intervalles réguliers.
- Temps moyen d'attente : généralement n'est pas minimal et peut varier substantiellement si les temps de cycles de la CPU du processus varient beaucoup
- Effet d'accumulation (Convoy effect): provoque une utilisation de la CPU et des périphériques plus lente que si on permettait au processus le plus court de passer le premier

SJF: Shortest Job First

Scheduling du travail plus court d'abord

- Associe à chaque processus la longueur de son prochain cycle de CPU
- Quand la CPU est disponible, elle est assignée au processus qui possède le prochain cycle le plus petit
- Si deux processus possèdent la même longueur, le FCFS est utilisé.
- SJF est optimal : il obtient le temps moyen d'attente minimal pour un ensemble de processus donné
- Il est particulièrement difficile de connaitre la longueur de la prochaine requête de la CPU

Ordonnancement avec priorités simples

- ☛ Une priorité est associée à chaque processus.
- ☛ CPU est allouée au processus de plus haute priorité
- ☛ Les processus ayant la même priorité sont choisis dans un ordre FCFS :

Processus	Temps de cycle	Priorité
P_1	10	3
P_2	1	1
P_3	2	3
P_4	1	4
P_5	5	2

- ➠ Problème: Famine (starvation) ou blocage indéfini: processus avec des basses priorités peuvent attendre indéfiniment (ne jamais être exécutés).
- ➠ Solution: Vieillissement (aging): technique qui consiste à augmenter la priorité des processus moins prioritaires

Sommaire

SRT : Shortest Remaining Time

Plus court temps restant

- La stratégie SRT (pour Shortest Remaining Time,) est la version avec réquisition de SJF : donc utilisable en temps partagé
- La priorité est donnée au processus dont le temps d'exécution restant est le plus faible en considérant à chaque instant les nouveaux arrivants
- Un processus actif peut donc être interrompu au profit d'un nouveau processus ayant un temps d'exécution estimé plus court que le temps nécessaire à l'achèvement du premier
- Le coût de SRT est supérieur à celui de SJF :
  - prise en compte du temps déjà alloué aux processus en cours
  - effectuer les commutations à chaque arrivée d'un travail court qui sera exécuté immédiatement avant de reprendre le processus interrompu
- Les travaux longs subissent une attente moyenne plus longue

SRT : Shortest Remaining Time

Plus court temps restant : optimisation 1/2

- En théorie SRT devrait offrir un temps d'attente minimum
- Du fait de son coût d'exploitation propre, il se peut que dans certaines situations, SJF soit plus performant
- Comment rendre SRT plus efficace : envisager plusieurs raffinements évitant la réquisition dans des cas limites
  - supposons que le processus en cours soit presque achevé et qu'un travail avec un temps d'exécution estimé faible arrive.
  - Doit-il y avoir réquisition?
  - Dans ce cas de figure garantir à un processus en cours dont le temps d'exécution restant est inférieur à un seuil qu'il soit achevé quelles que soient les arrivées

SRT : Shortest Remaining Time

Plus court temps restant : optimisation 2/2

- ☛ Comment rendre SRT plus efficace : envisager plusieurs raffinements évitant la réquisition dans des cas limites
  - ➡ le processus actif a un temps d'exécution restant légèrement au temps estimé d'un travail arrivant.
  - ➡ comme il y a réquisition! Donc ...?
  - ➡ si le coût de cette réquisition est supérieur à la différence entre les deux temps estimé, cette décision devient absurde!

Synthèse

- ☛ Bien évaluer avec beaucoup de précautions les couts engendrés par des mécanismes sophistiqués car ils peuvent dans bien des cas aller à l'encontre du but recherché : le gain de temps.

Round Robbin : Tourniquet

- L'ordonnancement de type tourniquet s'inspire de la technique FIFO :
  - association d'une tranche de temps autorisant la réquisition.
- Pour ce faire, les processus, au fur et à mesure qu'ils obtiennent le statut prêt, sont rangés dans une file d'attente
- A chacune de ses interventions, l'ordonnanceur alloue le CPU au processus en tête de file
- Si le temps d'exécution qui lui est ainsi imparti expire avant son achèvement, il est placé en queue de file et le contrôle est donné au processus suivant.

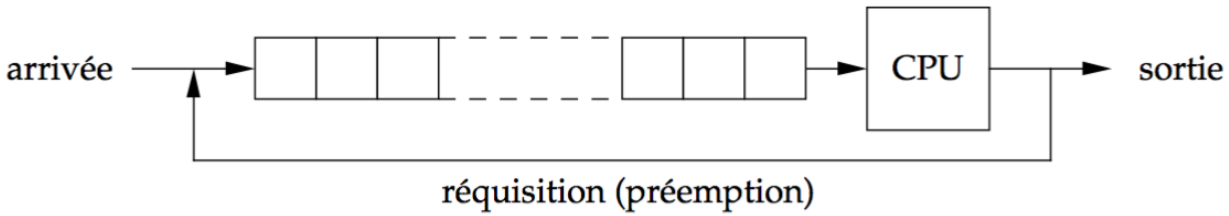


Figure: Ordonnancement par tourniquet

Round Robbin : Tourniquet

- Cette technique est satisfaisante dans les systèmes temps partagé où les utilisateurs interactifs doivent bénéficier de temps de réponse corrects.
- Le cout de la réquisition peut être maintenu faible si les mécanismes de commutation sont efficaces et la mémoire suffisante pour contenir plusieurs processus simultanément
- Réglage judicieux du quantum pour accroître le taux d'utilisation du processeur et donc diminuer les temps de réponse

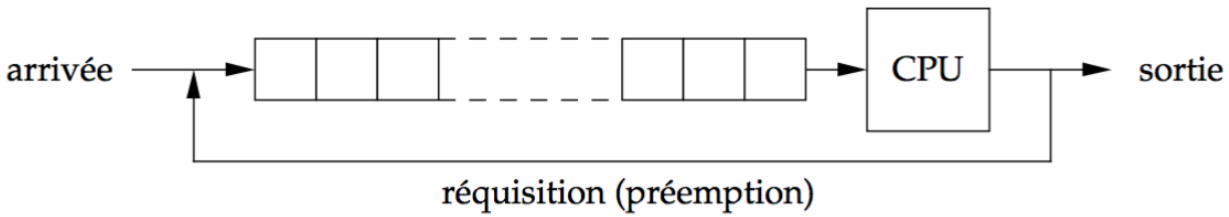


Figure: Ordonnancement par tourniquet

Vers un ordonnancement à multi-niveau

- Vu les problèmes dans SJF et SRT : la difficulté de connaitre à l'avance la quantité de temps CPU nécessaire à l'exécution d'un programme
- Un travail fonctionnant essentiellement en E/S n'utilisera en fait la CPU que de courts instants.
- A l'opposé, un travail réclamant le contrôler en permanence monopolisera la CPU durant des heures si l'on suppose un schéma sans réquisition.
- Or, il est préférable qu'un ordonnanceur :
  1. favorise les travaux courts ;
  2. favorise les travaux effectuant de nombreuses E/S (pour une bonne utilisation des unité externes) ;
  3. déterminer le plus rapidement possible la nature de chaque travail afin de le traiter en conséquence.
- Le Multi-niveau réponds à ces attentes

Le multi-niveau

Déroulement de l'ordonnancement du multi-niveau

1. Un nouveau processus est stocké en queue de la file de plus haut niveau
2. Il progresse dans cette file FIFO jusqu'à ce qu'il obtienne le contrôle
3. Si le processus s'achève ou libère la CPU pour une E/S ou une attente d'événement, il est placé en queue de la même file d'attente
4. Si le quantum expire avant, le processus est placé en queue de la file de niveau inférieur.
5. Il deviendra à nouveau actif lorsqu'il parviendra en tête de cette file et à condition que celles de niveau supérieur soit vides
6. Ainsi, à chaque fois que le processus épuisera sa tranche de temps il passera en queue de la file de niveau inférieur à celle où il se trouvait jusqu'à ce qu'il atteigne la file de plus bas niveau.

Le multi-niveau

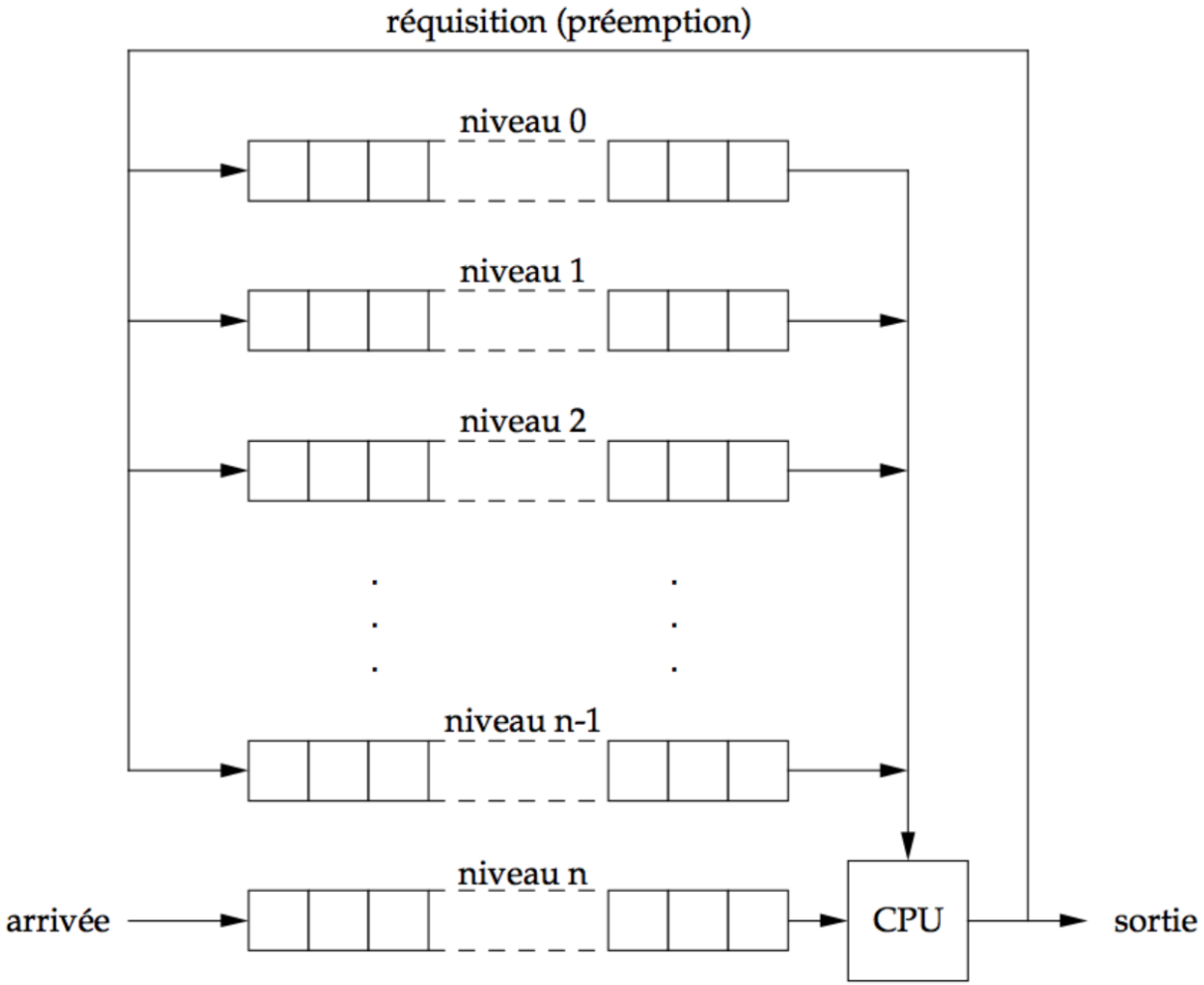


Figure: Schéma de principe du niquet multi-niveaux

Le multi-niveau

- La taille du quantum s'accroit au fur et à mesure que l'on descend dans les niveaux de file
- Progresse dans cette file FIFO jusqu'à ce qu'il obtienne le contrôle
- En conséquence, plus un travail est long, plus le temps CPU dont il bénéficie est grand.
- Par contre, le processeur lui sera alloué plus rarement puisque les processus des files supérieures ont une plus grande priorité
- Un processus en tête de quelque file que ce soit ne pourra devenir actif que si les files de niveau supérieur (si elles existent) sont vides
- Il y aura réquisition dès qu'un travail arrivera dans la file de plus haut niveau

Sommaire

☛ **Voir fiche TD**

Chapitre 2 : Ordonnancement de Processus

Dr Mandicou BA
mandicou.ba@esp.sn
http://www.mandicouba.net

Diplôme D'Ingénieur de Conception (DIC, 1^{e} année)
en Informatique / Télécommunications-Réseaux
Licence Professionnelle
en Génie Logiciel et Systèmes d'Information (GLSI)

ECOLE SUPERIEURE POLYTECHNIQUE
www.esp.sn