Chapitre3 : Gestion du processeur

3.1- Introduction :

Dans un système multi-utilisateurs à temps partagé, plusieurs processus peuvent être présents en mémoire centrale en attente d'exécution. Si plusieurs processus sont prêts, le système d'exploitation doit gérer l'allocation du processeur aux différents processus à exécuter. C'est l'ordonnanceur qui s'acquitte de cette tâche.

3.2- Définition de l'ordonnanceur (Scheduleur) et dispatcheur:

Chaque fois que le processeur devient inactif, le système d'exploitation doit sélectionner un processus de la file d'attente des processus prêts, et lui passe le contrôle. D'une manière concrète, cette tache est prise en charge par deux routines système en l'occurrence le Dispatcheur et le scheduleur.

3.2.1- Dispatcheur : Le dispatcheur est la routine système, qui s'occupe de l'affectation du processeur à un processus sélectionné par le scheduleur du processeur. Les taches confient à cette routine sont :

- ✓Commutation de contexte, c'est-à-dire sauvegarde du contexte du processus qui doit relâcher le processeur et restauration du contexte du processus qui aura le prochain cycle processeur.
- ✓Basculement vers le mode utilisateur, le dispatcheur s'exécute en mode maître et le processus à lancer est souvent un processus utilisateur qui doit être exécuté en mode esclave.
- ✓Branchement au bon emplacement dans le processus utilisateur pour le faire démarrer.

3.2.2- Scheduleur : Le scheduleur du processeur est également une autre routine système qui s'occupe de la sélection du processus qui aura le processeur, à partir de la file d'attente des processus prêts. Concrètement, cette routine n'est rien d'autre que l'implémentation d'un algorithme d'ordonnancement de processus.

3.3- Objectifs du scheduleur :

Les objectifs d'un scheduleur sont, entre autres :

- ✓S'assurer que chaque processus en attente d'exécution reçoit sa part de temps processeur.
- ✓Minimiser le temps de réponse.
- ✓Utiliser le processeur à 100%.
- ✓Utilisation équilibrée des ressources.
- ✓Prendre en compte des priorités.
- ✓Être prédictibles.

Ces objectifs sont parfois complémentaires, parfois contradictoires : augmenter la performance par rapport à l'un d'entre eux peut diminuer la performance considérée à partir d'autre. Il est impossible de créer un algorithme qui optimise tous les critères de façon simultanée.

3.4- Politiques d'ordonnancement :

Nous distinguons plusieurs algorithmes d'ordonnancement, les plus répondus sont :

3.4.1- Ordonnancement du premier arrivé premier servi : FIFO

Dans cet algorithme, les processus sont rangés dans la file d'attente des processus prêts selon leur ordre d'arrivé. Les règles régissant cet ordonnancement sont :

- ✓Quand un processus est prêt à s'exécuter, il est mis en queue de la file d'attente des processus prêts. .
- ✓Quand le processeur devient libre, il est alloué au processus se trouvant en tête de file d'attente des processus prêts.
- ✓Le processus élu relâche le processeur s'il se termine ou s'il demande une E/S (pas de réquisition).

Les principales caractéristiques de cet algorithme sont :

- ✓Ce scheduleur est sans réquisition.
- ✓Ce scheduleur est non adapté à un système temps partagé car dans un système partagé, chaque utilisateur obtienne le processeur à des intervalles réguliers.

3.4.2- Ordonnancement du plus court d'abord : SJF (Shortest Job First)

Dans cet algorithme, les processus sont rangés dans la file d'attente des processus prêts selon un ordre croissant de leur temps d'exécution. Les règles régissant cet ordonnancement sont :

- ✓Quand un processus est prêt à s'exécuter, il est inséré dans la file d'attente des processus prêts à sa position approprié.
- ✓Quand le processeur devient libre, il est assigné au processus se trouvant en tête de la file des processus prêts (ce processus possède le plus petit cycle processeur). Si deux processus ont la même longueur de cycle, on applique dans ce cas l'algorithme FIFO.
- ✓Le processus élu relâche le processeur s'il se termine ou s'il demande une E/S (pas de réquisition).

Les principales caractéristiques de cet algorithme sont :

- ✓Ce scheduleur est sans réquisition.
- ✓Ce scheduleur nécessite de connaître le temps d'exécution de chaque processus.
- ✓risque de famine (les processus de longue durée peuvent n'avoir jamais accès au processeur si des processus de courte durée arrivent en permanence).

3.4.3- Ordonnancement du plus court temps restant d'abord : SRTN (Shortest Remaining Time Next)

Dans cet algorithme, les processus sont rangés dans la file d'attente des processus prêts selon un ordre croissant de leur temps d'exécution restant. Les règles régissant cet ordonnancement sont :

1. Quand un processus est prêt à s'exécuter, il est inséré dans la file d'attente des processus prêts à sa position approprié.
2. Quand le processeur devient libre, il est assigné au processus se trouvant en tête de la file des processus prêts (ce processus possède le plus petit cycle processeur). Si deux processus ont la même longueur de cycle, on applique dans ce cas l'algorithme FIFO.
3. Le processus élu perd le processeur quand un processus ayant un temps d'exécution inférieur au temps d'exécution restant du processus élu. Le processus sera mis dans la file d'attente des processus prêts et le processeur sera alloué au processus qui vient d'entrer.

Les principales caractéristiques de cet algorithme sont :

- ✓Ce scheduleur est avec réquisition.
- ✓Cc scheduleur nécessite de connaître le temps d'exécution restant de chaque processus.

3.4.4- ordonnancement circulaire : le tourniquet (Round Robin)

Dans cet algorithme les processus sont rangés dans une file d'attente des processus prêts. Le processeur est alloué successivement a" ux différents processus pour une tranche de temps fixe Q appelée quantum.

Cet ordonnancement est régit par les règles suivantes :

I. Un processus qui rentre dans l'état prêt est mis en queue de la file d'attente des processus prêts.

- ✓Si un processus élu se termine ou se bloque (pour des raisons «de synchronisation par exemple) avant de consommer son quantum de temps, le- processeur est immédiatement alloué au prochain processus se trouvant en tète de la file d'attente des processus prêts.
- ✓Si le processus élu continue de s'exécuter au bout de son quantum, dans ce cas le processus sera interrompu est mis en queue de la file d'attente des processus prêts et le processeur est réquisitionné pour être réalloué au prochain processus en tête de cet même file d'attente.

Les principales caractéristiques de cet algorithme sont :

- ✓Ce scheduleur est avec réquisition.
- ✓Ce scheduleur est bien adapté aux systèmes temps partagé.
- ✓L'efficacité de ce scheduleur dépend principalement de la valeur du quantum Q car :

✓Le choix d'un Q assez petit augmente le nombre de commutation.

✓Le choix d'un Q assez grand augmente le temps de réponse du système.

2.4.5- Ordonnancement circulaire à plusieurs niveaux :

Dans cet algorithme on dispose de n files d'attentes (F0,...,F_n.i) pour les processus prêts, à chaque file est associée un quantum de temps dont la valeur croit avec i

Cet ordonnancement est régit par les règles suivantes :

- ✓Tout nouveau processus qui entre dans le système est mis dans la file d'attente F0.
- ✓Tout processus de la file d'attente F1 ne sera servi que si toutes les files Fi tel que j<i sont vides.
- ✓Si un processus de la file F1 a consommé son quantum ci; sans être terminé, il passe dans la file d'attente suivante F,+1 sauf celui de la file Fn.i qui doit retourner dans cette même file.

Les principales caractéristiques de cet algorithme sont :

- Ce scheduleur est avec réquisition.
- Ce scheduleur favorise les processus court sans savoir à l'avance combien de temps processeur ceux-ci vont utiliser.

2.4.6- Ordonnancement avec priorité :

Dans cet algorithme les processus sont rangés dans la file d'attente des processus prêts par ordre décroissant de priorité. L'ordonnancement est dans ce cas régit par les règles suivantes :

1. Quand un processus est admis par le système, il est inséré dans la file d'attente des processus prêts à sa position appropriée (dépend de la valeur de priorité).
2. Quand le processeur devient libre, il est alloué au processus se trouvant en tête de file d'attente des processus prêts (le plus prioritaire).
3. Un processus élu relâche le processeur s'il se termine ou se bloque (E/S, ou autre).

Remarque : Notons que si le système une en œuvre :a réquisition, quand un processus de priorité supérieure à celle du processus élu entre dans l'état prêt, le processus élu sera mi dans la file d'attente des processus prêt à la position appropriée, et le processeur est alloué au processus qui vient d'entrer.

Les principales caractéristiques de cet algorithme sont :

- ✓Ce scheduleur peut être avec ou sans réquisition.
- ✓Dans ce scheduleur, un processus de priorité basse risque de ne pas être servi (problème de famine) d'où la nécessité d'ajuster périodiquement les priorité ds processus prêts. L'ajustement consiste à incrémenter graduellement la priorité des processus de la file d'attente des processus prêts (par exemple à chaque 15 mn on incrémente d'une unité la priorité des processus prêts).