Système

Diedler Baptiste

September 25, 2023

1 ordonnancement

L'ordonnancement existe car par la multiprogrammation, plusieurs processus veulent les mêmes ressources en même temps. Pour cela, on va donc définir un ordre de passage. IL est possible de représenter l'ordonnancement des processus par un diagramme de Gantt.

ordonnancement: Choisir un processus parmi tous les processus dans une liste d'attente pour une ressource.

diagramme de Gantt: Représentation temporelle de l'allocation de ressources.

1.1 critères d'évaluation des algorithmes

taux d'utilisation: Minimiser le temps où une ressource n'est pas utilisée.

temps d'attente: Combien de temps passé en moyenne dans la file d'attente.

rotation: Durée moyenne totale d'exécution des processus.

débit: Nombre de processus finit par unité de temps. (clique de l'horloge)

1.2 FIFO:First In, First Out

FIFO:

File d'attente simple: premier arrivé, premier servi.
(+) Très simle à implémenter.
(-) Peu efficace: les processus de calcul bloquent le processeurs.
(-) Peu réactif: les petits processus attendent longtemps.

1.3 priorité

priorité:

Associer à chaque processus une valeur de priorité.
Une file d'attente par priorité.

Le calcul de la priorité se fait en fonction de plusieurs critères. Le type de processus, la durée estimée, les ressources utilisées.

1.4 réactivité

Elle permet aux petits processus (E/S) d'accéder rapidement au processeur.
Mais le changement de processus a un coût. Il faut donc profiter des commutations naturelles. On va donc ordonnancer quand un processus se termine ou qui il passe dans l'état prêt.

1.5 algorithme préemptif

Comment rendre le système plus réactif?

préemptif: Remplacement du processus actuellement en exécution par un processus arrivant dans la file des processus prêts.

1.6 plus ocurt d'abord

Priorité en fonction du temps restant. (Le temps restant est une estimation)
Il est toujours préemptif.
En cas d'égalité. La priorité revient au processus en cours d'exécution. Et sinon, c'est en fonction de l'ordre d'arrivée.
Pour connaître le temps restant, on fait une moyenne exponentielle des cycles précédents.
(+) Il est réactif. Les petits processus passent en premier.
(+) Il est optimal sur le temps d'attente moyen.
(-) Les processus de calcul ont peu le processeur. Car ils ne font pas (E/S).
(-) Non équitable. Famine pour les gros processus.

famine: On parle de famine lorsqu'un processus en attente n'a jamais accès à une ressource.

1.7 ordonnancement équitable

FIFO -> Un processus long peut monopoliser le processeur.
Plus court d'abord -> Des processus courts peuvent monopoliser le processeur.
Une solution serai que les paramètres de l'algorithme ne soient pas manipulables. Ou alors, avoir un temps limite par processus avant préemption.

1.8 round robin

FIFO avec un quantum de temps est rarement préemptif.

quantum de temps: C'est un entier qui représente la durée du processus sur le processeur avant de changer.

(+) Equitable: tous ont l'UC aussi rapidement.
(+) Réactif: les petits processus attendent peu.
(-) Temps d'attente moyen plus élevé.
(-) Beaucoup de commutations. (surcoût)
Il est efficace seulement si 80% des processus ont un temps inférieur au quantum.