Ordonnancement

Rappel

- L'ordonnanceur partage le temps processeur entre les processus
- Un processus peut être dans 3 états : elu, éligible ou bloqué
- À chaque entrée sortie un processus passe dans l'état bloqué

Objectif

- Équité
  - propriété d'un ordonnanceur garantissant à tous les processus les mêmes chances d'obtenir le processeur
- Famine (en anglais starvation)
  - situation dans laquelle un processus attend indéfiniment le processeur

Préemption

- Sur un système préemptif le système d'exploitation peut arrêter un processus et le bloqué à tout moment.
- Sur un système non préemptif, il faut attendre que le processus rende la main (appel système yield)

FIFO sans préemption

- Premier arrivé, premier servi
- Ordonnancement coopératif (pas de préemption) : les processus rendent la main « de leur plein gré »,
  - lorsqu'ils se terminent,
  - lorsqu'ils se bloquent,
  - lorsqu'ils font l'appel système yield.

FIFO sans préemption

- simple
- surcoût faible
- temps de réponse dépend du processus qui a la main : tant qu'il ne rend pas la main, les autres doivent attendre
- Adapté pour les machines avec nombreux cœurs de calcul
- Adapté pour les machines peu puissante, ou le surcoût doit être minimisé

FIFO avec pré emption

- FIFO avec préemption
- Chaque processus reçoit un quantum de temps
- Une fois le quantum épuisé, le processus passe la main au retourne dans la file d'attente
- En anglais : round robin (ruban rond)

Tourniquet

- FIFO avec préemption
- Chaque processus reçoit un quantum de temps
- Une fois le quantum épuisé, le processus passe la main au retourne dans la file d'attente
- En anglais : round robin (ruban rond)

Tourniquet

- En diminuant la durée du quantum :
- le temps de réponse diminue, mais…
- le surcoût augmente
- Le quantum idéal dépend de la durée moyenne d'une interaction et du nombre de processus…

Tourniquet

- temps de réponse borné, indépendamment des processus (calculs ou E/S)
- très sensible au choix du quantum
- défavorise les processus orientés E/S (bloqués avant la fin de leur quantum)

Shortest job next

- On donne toujours la main à celui qui va mettre le moins de temps avant de se bloquer / terminer
- suppose d'avoir une connaissance / estimation de ce temps pour chaque processus : hypothèse forte

Shortest job next

- Avantages
  - maximise le temps de réponse, le débit (nombre de processus terminés par unité de temps)
- Inconvénients
  - surcoût
  - inéquitable, famine possible (processus calculatoires)

Highest Response Ratio Next

- Variante de la stratégie précédente :
- on prend en compte le ratio du temps que le processus a passé à attendre sur le temps dont il a besoin
- Supprime le problème de famine
- plus un processus attend, plus il augmente ses chances d'obtenir la main
- Mais suppose toujours la connaissance du temps d'exécution

Stratégie avec plusieurs files d'attente

- Plusieurs files, ordonnées par priorité
- Lorsqu'un processus se bloque ou se termine, il retourne dans la même file
- Lorsqu'il épuise son quantum, il passe dans la file suivante
- Favorise le temps de réponse des processus orientés E/S

Priorité

- priorité externe

  → propriété définie par l'utilisateur, variant peu
- priorité interne

  → propriété gérée par l'ordonnanceur, variant plus souvent

Nice

- Exemple sous UNIX: la commande nice permet de modifier la priorité externe
- Influence de la priorité externe:
  - en fixant leur priorité interne de départ
  - en limitant la plage de priorité interne dans laquelle ils peuvent évoluer

Partage équitable

- Le processus n'est pas forcément le bon grain pour mesurer l'équité d'un ordonnanceur
- Dans certain cas, on peut souhaite partager le temps processeur équitablement entre
  - les utilisateurs

  → indépendamment du nombre de processus lancé par chacun d'eux
- des groupes d'utilisateurs

  → indépendamment du nombre d'utilisateur et de processus

Temps réel

- Objectif : garantir autant que possible aux processus certains délais
- Inconvénient : le système doit faire des estimations « dans le pire des cas »

  → dégrade les performances (temps de réponse, débit)

Conclusion

- Problème ouvert
- Il faut connaître les choix de l'utilisateurs : nombre de processus, puissance nécessaire, temps d'exécution, nombre d'entrées sortie
- Les systèmes modernes mixent de nombreuses stratégies parmi celles étudiées