Ordonnancement - Scheduling

Programmation Système — R3.05

Clément Raïevsky

Département Informatique

BUT Informatiques 2ème année

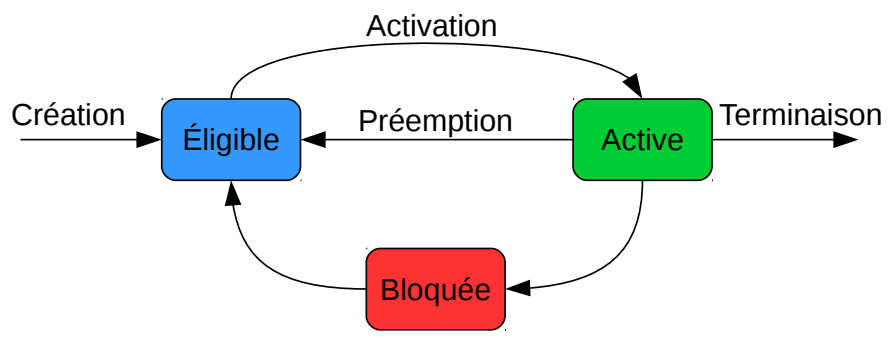
Ordonnancement

Ensemble des procédés permettant de choisir la ou les tâches actives d'un système

Plus généralement, attribution de ressources à des tâches qui en ont besoin

États possibles d'une tâche existante

- **Éligible**
- **Active**
- **Bloquée**



Pourquoi ?

Qu'y a-t-il de mal à ne pas avoir d'ordonnancement ?

Systèmes mono-processeur

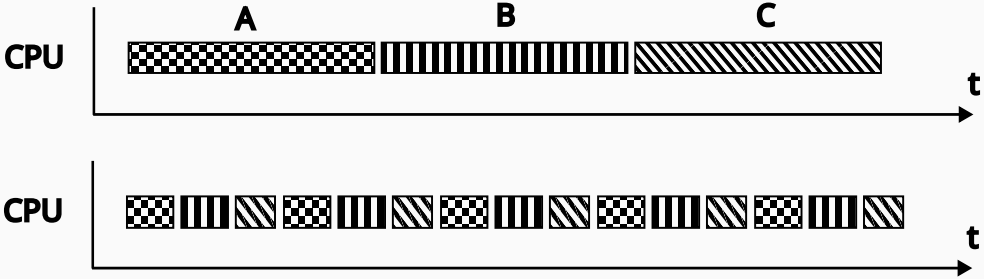
- Une seule tâche à la fois, *jusqu'à sa fin*
- Gaspillage de cycles processeur
  - En cas d'entrées-sorties (disque, réseau, utilisateur)
  - En cas de défaut de page
- Risque de blocage du système (boucle infinie)

Systèmes multi-processeurs

- Nombre de tâches limité
- Les mêmes gaspillages

Mono-processeur - Multi-tâches

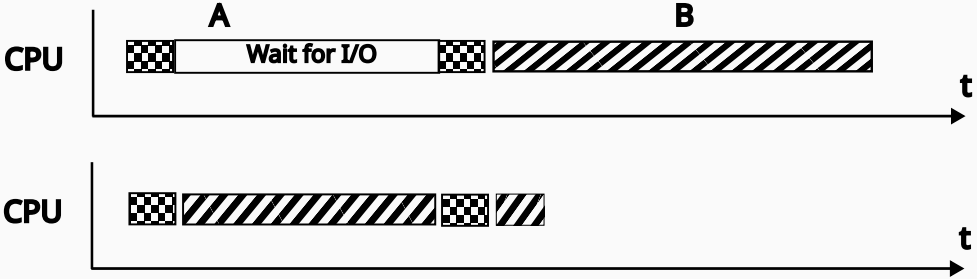
- Donner l'impression du parallélisme
- Éviter le blocage du système



On ne gagne pas de temps de calcul

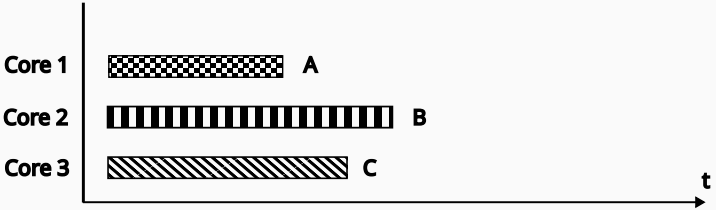
Mono-processeur - Multi-tâches - Attente d'entrées sorties

Sauf si on met les tâches en pause au bon moment



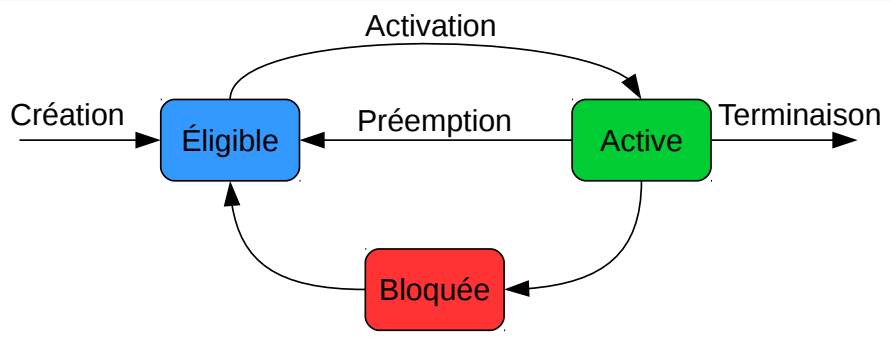
Multi-processeur

- Parallélisme effectif
- On gagne du temps sur les tâches parallélisables



États possibles d'une tâche existante

- **Éligible** (prête à devenir active)
- **Active** (en cours d'exécution)
- **Bloquée** (en attente d'I/O, Page Fault, etc.)



Éléments noyau impliqués

Ordonnanceur - *Scheduler*

Élément du noyau responsable de sélectionner la tâche éligible qui sera la prochaine à être active

Distributeur - *Dispatcher*

Élément du noyau en charge d'effectuer le basculement entre la tâche courante et la tâche choisie par l'ordonnanceur

Problèmes qui se posent

- Quand changer de tâche ?
- Quelle tâche choisir ?

Quand changer de tâche ?

Critères de choix

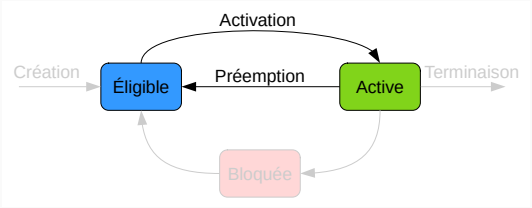
- Réactivité du système (toutes les tâches doivent avoir du temps)
- Équité entre les tâches (en fonction de leur priorité)
- Minimiser le nombre de changement de tâche (ça coûte cher)

Mesures associées

- Temps maximal entre deux créneau attribués à une tâche
- Différence maximale de temps alloué à deux tâches de même priorité
- Efficacité : proportion de temps passé à changer de tâche

Illustration - Méthode du tourniquet

- Les tâches **éligibles** sont dans une FIFO circulaire
- Un changement de tâche à intervalle de temps fixe



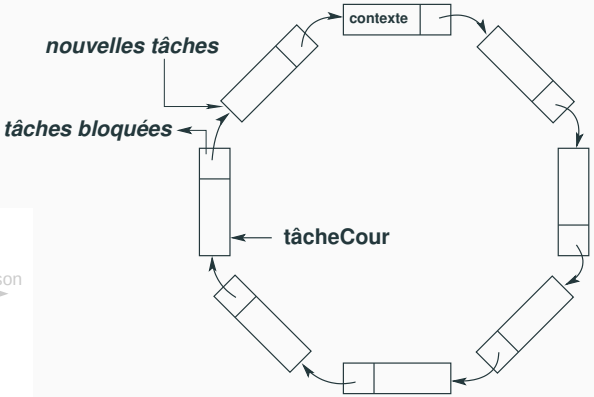
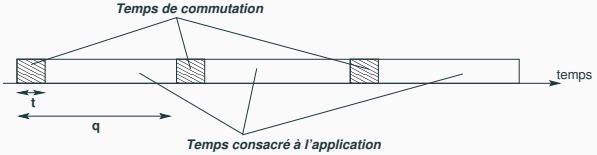


Illustration - Méthode du tourniquet

- **Méthode du tourniquet**
  - **Sans réquisition de fin de quantum :** très efficace s'il y a peu de tâches qui se bloquent souvent (E/S).
  - **Avec réquisition de fin de quantum :** c'est une méthode impartiale pour les tâches de même priorité.
- **Choix du quantum : compromis entre débit et efficacité**



  - *Efficacité* : temps consacré à l'application / temps total
    $$E = \frac{q-t}{q}$$
  - *Débit* : nombre de tâches traitées par seconde
    $$D = \frac{1}{q}$$

Illustration - Méthode du tourniquet

- Choix du quantum : exemples

$t = 1ms$ **et** $q = 5ms$

$E = 0.8$ (80% du temps consacré à l'application)

$D = 200$ (200 tâches traitées par seconde)

$t = 1ms$ **et** $q = 50ms$

$E = 0.98$ (98% du temps consacré à l'application)

$D = 20$ (20 tâches traitées par seconde)

- Meilleur débit $\Rightarrow$ meilleur temps de réaction
- Meilleure efficacité $\Rightarrow$ temps total d'exécution plus court
- Bien entendu, t doit être le plus petit possible (un t petit augmente l'efficacité sans diminuer le débit).

Illustration - Méthode du tourniquet

- Choix du quantum : exemples

$t = 1ms$ **et** $q = 5ms$

$E = 0.8$ (80% du temps consacré à l'application)

$D = 200$ (200 tâches traitées par seconde)

$t = 1ms$ **et** $q = 50ms$

$E = 0.98$ (98% du temps consacré à l'application)

$D = 20$ (20 tâches traitées par seconde)

- Meilleur débit $\Rightarrow$ meilleur temps de réaction
- Meilleure efficacité $\Rightarrow$ temps total d'exécution plus court
- Bien entendu, t doit être le plus petit possible (un t petit augmente l'efficacité sans diminuer le débit).

Illustration - Méthode du tourniquet

- Choix du quantum : exemples

$t = 1ms$ **et** $q = 5ms$

$E = 0.8$ (80% du temps consacré à l'application)

$D = 200$ (200 tâches traitées par seconde)

$t = 1ms$ **et** $q = 50ms$

$E = 0.98$ (98% du temps consacré à l'application)

$D = 20$ (20 tâches traitées par seconde)

- Meilleur débit $\Rightarrow$ meilleur temps de réaction
- Meilleure efficacité $\Rightarrow$ temps total d'exécution plus court
- Bien entendu, t doit être le plus petit possible (un t petit augmente l'efficacité sans diminuer le débit).

Illustration - Méthode du tourniquet

- Choix du quantum : exemples

$t = 1ms$ **et** $q = 5ms$

$E = 0.8$ (80% du temps consacré à l'application)

$D = 200$ (200 tâches traitées par seconde)

$t = 1ms$ **et** $q = 50ms$

$E = 0.98$ (98% du temps consacré à l'application)

$D = 20$ (20 tâches traitées par seconde)

- Meilleur débit $\Rightarrow$ meilleur temps de réaction
- Meilleure efficacité $\Rightarrow$ temps total d'exécution plus court
- Bien entendu, t doit être le plus petit possible (un t petit augmente l'efficacité sans diminuer le débit).

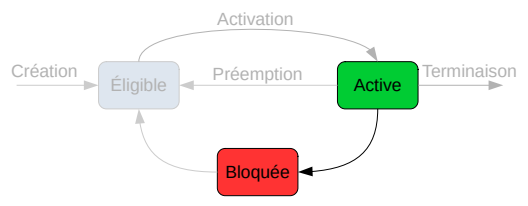
Quand changer de tâche ? - Cas particuliers

Cas particuliers : Opérations bloquantes

- Tâche effectuant une opération d'entrée/sortie (Disque, Réseau, Utilisateur)
- Tâche nécessitant du swaping (Page Fault)

→ Passage de la tâche active dans l'état "**Bloquée**"

- Elle sera remise dans les tâches éligibles une fois l'opération bloquante terminée



Quelle tâche choisir ?

- La plus prioritaire ?
- Celle qui a été active il y a le plus longtemps ?
- Celle qui est interactive ?
- Celle qui affiche des choses à l'utilisateur ?
- Le serveur ou le client minecraft ?

Dépend de la nature du système

- Serveur de calcul : FIFO peut convenir
- Machine personnelle : tâches interactives à prioriser
- Système embarqué temps réel : Priorité pure (calcul freinage)

Stratégies d'ordonnancement

- **Priorité pure**
  - Une et une seule tâche par niveau de priorité
  - La préemption présente peu d'intérêt : la tâche la plus prioritaire s'exécute jusqu'à sa fin ou jusqu'au blocage. L'ordonnancement est donc lié au interruptions externes.
- **Réalisation avec une liste chaînée**

  - Accès immédiat à la première
  - Insertion après recherche de la position
- **Réalisation avec une table**

  - Accès à la première par recherche de la première case non vide
  - Insertion directe avec le niveau de priorité

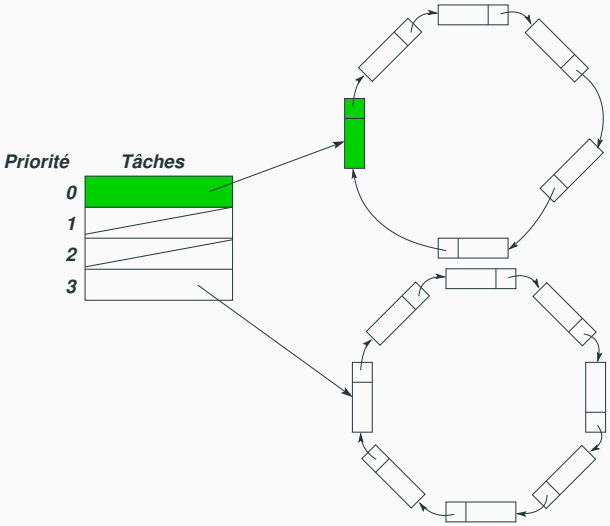
Stratégies d'ordonnancement

Méthode mixte : tourniquet multi-niveau

Plusieurs niveaux de priorité et plusieurs tâches par niveau

Souplesse et généralité

- une tâche par niveau ⇒ priorité pure
- toutes les tâches au même niveau ⇒ tourniquet simple



Stratégies d'ordonnancement (5)

Autres méthodes

- Systèmes à temps partagé de type Unix ou Windows

Algorithmes beaucoup plus complexes

- privilégier les tâches courtes et les tâches interactives
- vieillissement des priorités : la priorité diminue avec le temps...
- ... mais peut augmenter à nouveau
- le quantum n'a pas une durée fixe

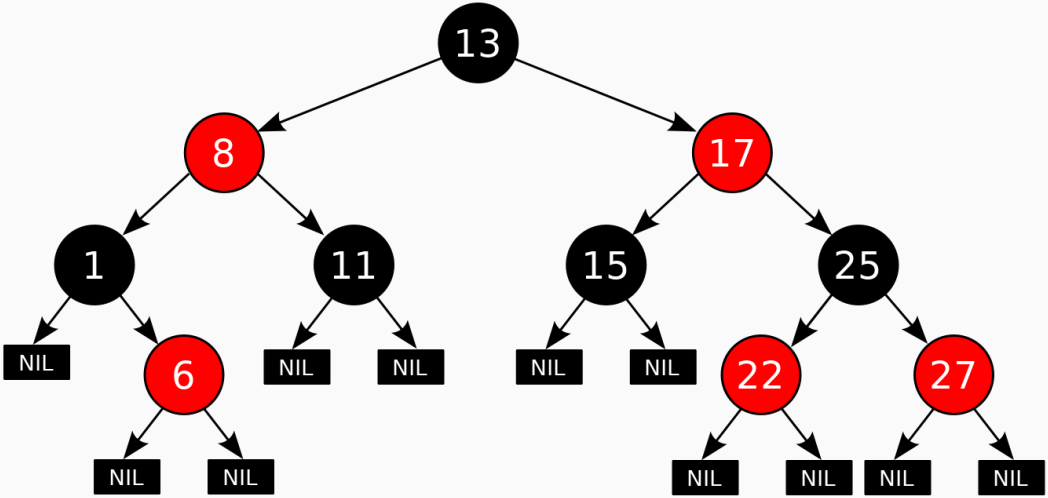
Conséquences

- Gestion plus lourde $\Rightarrow$ baisse d'efficacité
- Ordonnancement complètement contrôlé par le système $\Rightarrow$ l'utilisateur n'a que très peu accès à la priorité

Noyau Linux

Completely Fair Scheduler - CFS

- Tâches classées dans un arbre de recherche binaire
- par ordre décroissant de temps alloué au processeur (*v_runtime*).
- L'ordonnanceur choisi la tâche ayant le temps alloué minimal.
- La priorité d'une tâche modifie le temps maximum qui lui est alloué.



By Cburnett - Own work, CC BY-SA 3.0, https ://commons.wikimedia.org/w/index.php ?curid=1508398