Gestion des processus

à la Linux

Multiprocessus, à chaque instant t, 1 seul processus sur le processeur → Le systeme fait l'entrelacement entre les processus, ce qui donne une illusion de parallelisme

→ Gestion de priorité avec 1 ordonnanceur

Pour gérer ceci on utilise une table des processus (lorsque l'on fait 1 appel système à fork c'est l'index dans cette table qui est retourné).

Par exemple 1 case de cette table peut contenir 3 classes d'information :

- informations sur la gestion du processus : valeurs registres, compteur ordinal, pointeur sur la pile, numéro de processus parent, tableau d'interruptions, …
- inforlations sur la mémoire : pointeur segement du code, pointeur segment données, pointeur segment pile
- informations sur les fichiers : description de fichiers, indentifiant groupe utilisateur, …

Pour ces informations une partie doit se trouver en mémoire (autant que possible) (autant que possible) ces informations sont regroupées en 2 catégories celle pour le systeme (à ne pas déplacer en disque)

- celle utilisateur que l'on n'a pas besoin de garder en mémoire

Lorsque le système décide changer de processus, il fait ce que l'on appelle 1 changement de contexte. Cela consiste à stocker les informations du processus courant et à charger les informations du processus suivant (choix de ce dernnier délégué à l'ordonnanceur) et à demander au processus de commencer l'exécution du processus choisi.

Exemple d'opérations

- Au niveau matériel on stocke le compteur ordinal; …
- Au niveau matériel on charge le nouveau compteur ordinal (pour executer du code du Système d'exploitation)
- Par exemple à ce stade on peut continuer le stockage des informations du processus courant.

On peut commencer l'execution l'execution du code du systeme d'exploitation

- chargement pile (au niveau assembleur par exemple)
- typiquement l'ordonnaceur qui choisit le prochain processus)

On retourne au niveau assembleur pour charger les informations du processus courant et commencer son exécution

Politique d'ordonnancement

L'ordonnanceur choisit le processus à exécuter en maintenant souvent les structures de données suivantes :

- le processus courant
- la liste des processus prêts à s'exécuter, en attente, zombies, … (ceci peut-être codé avec 1 drapeau)
- la liste des interruptions. Pour chacune les processus en attente (et peut-être traitement à faire !)

Automate des états du processus

(trouver image)

L'ordonnanceur choisit à chaque changement de contexte 1 param les prêt à s'exécuter, il peut choisir 1 politique de :

- FIFO le processusŝ prêts à s'exécuter sont dans 1 file. Inconvénient le processus avec bcp d'appels bloquants sont désanvantagés
- On peut améliorer le FIFO en gérant des priorités. L est 1 file de priorités
- Anneaux : chacun a 1 temps qui lui est imparti et ce temps doit être utilisé avant 1 changement de contexte. On désavantage les autrres qui ne sont pas en attente (ceux en attente occupant le processus sans rien faire)

Solution pour ne pas désavantager ceux qui ne son pas en attente :

- si 1 processus en attente et son temps pas terminé, donné la main à 1 autre.
- lorsque le processus en attente est reveillé :
  - soit lui donner la main automatiquement (avec le temps qui lui restait)
  - soit la prochaine fois lui ajouter ce temps à son temps imparti

Les priorités dépendant du nomvre d'exécutions à exécuter (par ex on choisit toujours le plus court on le 2 plus courts, …)

Comment décider du nombre d'instructions à executer ? On fait des stats basées sur le passé.

L'ordonnanceur peut choisir 1 polotique de 'la liste des pcs à s'executer = L.)

- Susteme interactif : on prend en compte les types d'instructions et on met à jour les priorités
  - On fait des statistiques basées sur le passé (et ou futur danger peut etre instructions non autorisées)
- Mix : priorité + système interactif

Le diner des philosophes

5 philosophes assis autour d'une table et on commandé un plat (supposé infini) de spaghettis. Cinq fourchettessont disposées autour de la table, une entre chaque assiette. Chaque philosophe passe alternativement dans 3 états : *en train de penser, affamé, en train de manger*. Tout d'abord, il pense pendant un temps aléatoire. Puis, il est affamé et attend que la fourchette à sa gauche et que la fourchette) sa droite se libèrent. Ensuite, il prend les deux fourchettes et commence à manger pendant un temps aléatoire. Quand il a fini de manger, le philosophe pense (jusqu'à ce qu'il soit à nouveau affamé).

Question 1 *Pourquoi chaque philosophe est-il modélisé par un processus ? Est-ce qu'autre chose est modélisé par un processus ?*

Un philosophe est un processus car il effectue des calculs. Non les autres éléments sont des ressources.

Question 2 *Quelles sont les ressources critiques ?*

La fourchette.

Question 3 *Quand un philosophe doit-il demander la ressource ?*

Quand il est affamé.

Question 4 *Quand un philosophe doit-il libérer la ressource ?*

Quand il a fini de manger.

Question 5 *Quelle est la section critique ?*

C'est "Manger", c'est la seule fonction qui utilise des ressources critiques.

Question 6 *Pour protéger l'utilisation des fourchettes dans la section critique, nous allons implémenter une sémaphore par fourchette? Décriver le pseudo-code d'un programme qui implémente le diner des philosophes de la manière suivante : - Chaque philosophe pense pendant un temps aléatoire, - Chaque philosophe mange pendant un temps aléatoire, - L'accès aux ressources critiques est protégé.*

Exclusion mutuelle sur chaque fourchette variable de verrouillage

```
while(true) {
    penser();
    prendre_semaphore_gauche();
    prendre_fourchette_gauche();
    prendre_semaphore_droite();
    prendre_fourchette_droite();
    manger();
    liberer_fourchette_gauche();
    liberer_semaphore_droite();
    librerer_fourchette_gauche();
    liberer_semaphore_gauche();
```

```
}
```

Question 7 *Qu'est-ce qu'un interblocage ?*

Quand aucun processus ne peut entrer en section critique (ici manger).

Question 8 *Est-ce que votre solution est susceptible de faire apparaitre un interblocage ?*

Oui (exemple du carrefour priorité à droite).

Exercice 2 *Est-ce que l'interblocage est susceptible d'arriver en pratique ?*

Oui même si la probabilité est très faible.

Question 9 …

Question 10 *Comment faire en sorte que le problème soit résolu sans interblocage ?*

1. Ne donner que deux fourchettes à la fois, pas une.
2. Un processus "serveur" est chargé de distribuer les ressources. Accepter 2 au plus qui mangent