[Info 524]Système d'exploitation

Paul CLAVIER

11 octobre 2013

Table des matières

1 Processus

1.1 C'est quoi ?

$$\text{processus} \neq \text{programme}$$

Un processus est un programme en cours d'exécution. En particulier, un programme peut donner plusieurs processus. Avec un processus, l'OS garde de nombreuses informations comme :

- adresse mémoire de l'exécutable
- l'utilisateur qui a lancé le programme
- l'heure de début
- le PID
- la priorité
- le compteur ordinal
- la liste des fichiers ouverts
- l'état
- la liste des processus créés

1.2 États

Un processus peut être :

- en exécution
- bloqué
- en attente

1.3 Création d'un processus

- Sous POSIX, un seule manière de créer un processus : la fonction *fork()*. Elle duplique le processus en cours. Le processus créé est identique au processus de départ. La seule différence est la valeur de retour de la fonction *fork()* :
  - dans le processus initial, la valeur est le PID du processus créé.
  - dans le processus créé, fork renvoie 0
- execl(chemin, args) : remplace le processus par un nouveau processus en utilisant l'exécutable du chemin.

Les processus POSIX sont mis dans une structure arborescente. Chaque processus créé (avec fork) est un fils du processus qui le créé.

1.4 Mort d'un processus POSIX

La fonction kill() permet de tuer un processus. Lorsqu'un processus reçoit le signal SIG.TERM, il s'arrête. L'OS le supprime de la table des processus. Lorsqu'un processus s'arrête, ses sous-processus deviennent fils de init.

Remarque : quand un processus s'arrête, le système garde de l'information pour pouvoir informer le processus père (fonction wait). Tant que le père n'a pas fait de "wait", le processus est conservé. On parle de zombie.

1.5 Ordonnancement

ordonnanceur : partie de l'OS qui décide quel processus s'exécute. C'est un problème compliqué.

1.5.1 non préemptif

L'OS n'arrete pas un processus en cours d"exécution.

1.5.2 préemptif

L'OS peut stopper un processus en exécution.

- Tourniquet avec quantum de temps : Les processus sont dans une file, et à intervalle de temps régulier, on remplace le processus en exécution par le suivant.
- Tourniquet avec priorité : chaque processus a une priorité. On exécute le premier processus le plus prioritaire pendant un intervalle de temps. En pratique, chaque priorité a sa propre file.
- Loterie : Chaque processus (pret) reçoit un nombre de tickets proportionnel à l'utilisation voulue du processeur. Les processus d'un même utilisateur se partagent les tickets, les processus plus prioritaires reçoivent plus de tickets. L'ordonnanceur choisit un processus en tirant un ticket au hasard.

2 La mémoire

2.1 espace d'adressage : abstraction de la mémoire

Avec un système multi-programmé, l'allocation naïve ne marche pas bien. La mémoire se fragmente au fur et a mesure de l'apparition des processus. On peut faire croire au processus que sa mémoire commence à 0.

2.2 Représentation de la mémoire dans l'OS

L'OS connaît le quantité de RAM à sa disposition, il doit gérer les parties libres/occupées de la mémoire.

- Avec un tableau de bits : chaque bit représente l'état d'un bloc mémoire.
- Avec une liste chaînée de descripteurs de zones libres.

2.3 Algorithme d'affectation

Il faut choisir une zone libre de mémoire à la demande :

- First Fit : On choisit le premier emplacement libre assez grand.
- Next Fit : On prends le premier emplacement assez grand en partant de l'allocation précédente.
- Best Fit : On prends le meilleur emplacement (plus proche possible).
- Worst Fit : On prends le pire.

2.4 Mémoire virtuelle, pagination

Objectif : pouvoir ajouter de la mémoire à la demande des processus. On découpe la RAM en blocs de taille fixe (4 ko). Ces blocs sont appelés *cadres*. L'espace d'adressage va de 0x000000... à 0xffffff... (toutes les adresses disponibles).