[Info 524]Système d'exploitation

Paul CLAVIER

8 novembre 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). Cet espace virtuel est découpé en blocs de taille fixe (même que celle des cadres), appelés pages. Les informations sur les pages sont stockées dans la table des pages.

2.5 Table des pages

Pour chaque page, on stocke :
- n° de cadre correspondant
- boolean pour savoir si la page est in (RAM) ou out (swap)
- boolean pour savoir si la page a été lue (référence)
- boolean pour savoir si la page a été modifiée (dirty)

La traduction est faite par la MMU en utilisant la table des pages (niveau hardware).

2.6 Pagination et swaping

Pour les systèmes avec peu de mémoire, on utilise en général un morceau du disque dur comme mémoire secondaire. Cette mémoire ne peut pas remplacer la RAM. Elle sert à stocker des cadres qui ne sont pas utilisés. Lorsque la RAM est pleine, la MMU ne peut pas donner de nouveaux cadres, mais le noyau peut libérer des cadres en les copiant sur le disque. Le choix des cadres peut être fait de plusieurs manières. On parle d'algorithmes de remplacement de pages.
- FIFO : premier cadre arrivé, premier cadre supprimé.
- LRU : "least recently used"
- NRU : "not recently used".

3 Système de fichier

Un fichier est la plus petite unité d'information du disque accessible au processus.

Hypothèse : On suppose que tous les supports ont deux opérations :
- read(k) : lire l'octet k du support
- write(k, o) : écrire l'octet o à l'emplacement k

Il faut définir les fichiers uniquement à partir de ces fonctions.

3.1 abstraction des fichiers

identification des fichiers : chaque fichier à un nom avec une ou plusieurs extensions, des droits associés aux utilisateurs, des attributs.

3.2 accès aux fichiers

se fait en utilisant des fonctions "haut niveau" qui n'utilise que les deux fonctions read et write. On peut ensuite programmer des fonctions plus complexes à partir de ça.

3.3 organisation des fichiers

- en vrac : tous les fichiers au même endroit
- en arbre : chaque répertoire contient des fichier et d'autres répertoires. Chaque fichier est identifié par la branche de répertoires et son nom (chemin).
- en base de donnée

3.4 implantation des systèmes de fichiers

3.4.1 Partition

Un partition est une zone consécutive du disque.

3.4.2 Organisation des fichiers sur la partition

- les fichiers sont stockés les uns à la suite des autres
- table de blocs en mémoire (FAT)