Partage de la mémoire

michel billaud

2020

Table des matières

1 Partage de la mémoire

Dans un système multi-tâche, la mémoire est partagée entre le système d'exploitation et les processus qui s'exécutent.

On regarde comment on en arrive là, et les mécanismes matériels qu'il faut introduire pour que ça marche correctement, en allant jusqu'à la mémoire virtuelle.

1.1 Premiers ordinateurs (modèle linéaire)

Dans les premiers ordinateurs, l'opérateur manipule les clés du pupitre pour piloter l'ordinateur.

L'opérateur commence par introduire en mémoire un petit programme, puis lance son exécution.

Ce programme est un "chargeur" (*loader*) comportant quelques instructions qui recopie en mémoire le **code exécutable** d'un programme lu sur un ruban perforé, puis lance son exécution :

```
PSEUDO-CODE DU CHARGEUR :
  pointeur  destination = adresse(zone programme) ;

  tant qu'il reste des données à lire {
     lire un octet ;
     *destination = octet ;
     destination++
  }
  registre PC = adresse programme
```

Par conséquent, chaque programme que l'on fait exécuter doit contenir le code qui gère les divers périphériques (imprimantes, lecteurs de cartes, de bandes, imprimantes, traceurs…) dont il a besoin.

Ça pose plusieurs problèmes

- la taille du programme à charger
- le fait que les programmes doivent être mise à jour à chaque modification/amélioration des fonctions.

1.2 Moniteur d'enchaînement des travaux

Une idée qui vient rapidement :

- au démarrage de la machine, le loader charge, dans une partie de la mémoire, les bibliothèques communes, et un "**moniteur d'enchaînement des travaux**". C'est un système d'exploitation primitif.
- Le moniteur est un "loader" qui
  - charge un programme utilisateur dans une autre zone mémoire
  - lance leur exécution
  - reprend la main quand le programme est terminé, et passe automatiquement au suivant.

C'est un système d'exploitation primitif, qui permet de mieux tirer profit des ressources de la machine. Les programmes utilisateurs sont déchargés de la gestion du matériel.

Éventuellement, ce moniteur peut comporter une boucle de dialogue avec l'opérateur, qui tape des commandes pour piloter le fonctionnement.

1.2.1 Protection de la mémoire

La mémoire comporte deux parties

- celle réservée au système d'exploitation
- celle qui sert au programme utilisateur.

Pour que ça marche correctement, il faut prendre quelques précautions, pour empêcher le programme utilisateur de "flinguer" le système, en allant modifier des variables du système, ou en agissant à tort et à travers sur les périphériques.

C'est ici où on utilise les deux **modes** : le système d'exploitation s'exécute en mode privilégié, les programmes utilisateur en mode normal.

On y joint une **protection de la mémoire** pour que les programmes utilisateurs ne puissent pas accéder à la mémoire qui contient les données (et le code) du système.

Les programmes utilisateurs sont donc contraints de passer par des **appels systèmes** pour utiliser les périphériques.

1.2.2 Mécanisme de protection : registre limite.

Le mécanisme le plus simple pour la protection matérielle est d'avoir, dans le processeur, un **registre limite** qui indique où se trouve la séparation entre les deux parties de la mémoire. Ce registre est chargé par le système d'exploitation quand il active le programme utilisateur.

Un circuit (comparateur) compare à chaque instant le contenu de ce registre avec les adresses utilisés par le processeur.

- En **mode normal**, une **exception** est levée quand le processeur tente d'accéder à la partie protégée où se trouve le système.
- En **mode privilégié**, le processeur peut accéder à toute la mémoire.

1.2.3 Systèmes multi-tâches, modèle de mémoire linéaire

Dans les système multi-tâches les plus simples, les zones mémoires des différents processus sont situées les unes après les autres. C'est ce qu'on appelle un **modèle mémoire linéaire**, ou **plat**.

La **région** qui peut être accédée en mode normal, par un processus utilisateur, doit être délimitée plus précisément, par exemple par une **paire de registres** qui indiquent le but et la fin de la région (ou le début et la longueur).

Encore mieux : dans certains micro-contrôleurs (par exemple ARM CORTEX M3), la MPU (*Memory Protection Unit*) décrit **plusieurs régions**, avec des **droits** différenciés (modifier, lire, exécuter) Ceci permet de partager de la mémoire entre processus.

1.2.4 Limitations du modèle linéaire

Le modèle linéaire, tel qu'il est exposé ci-dessus, convient à un système dans lequel les tâches sont chargées au début et restent présentes *ad vitam aeternam*. C'est le cas de beaucoup d'applications embarquées.

Par contre il n'est pas adapté aux systèmes multi-utilisateurs où des programmes sont lancés et arrêtés. En effet, pour lancer un programme, il faut lui **allouer** un **espace mémoire contigu** (en un seul bloc) assez grand, à prendre dans les espaces rendus disponibles (**libérés**) par la fin d'autres programmes.

1.2.5 Allocation contiguë

L'occupation de la mémoire varie, avec l'arrivée et la fin des processus.

TODO image

1.3 Adresses logiques,

1.3.1 Translation des programmes

Objectif : pouvoir déplacer (translater) les zones occupées par les processus, pour constituer un gros bloc libre à partir de plusieurs petits.

Jusqu'ici, en mode normal, le processeur manipule des adresses physiques qui ne se réfèrent qu'à une "région" encadrée par les registres limites.

Changement : en mode normal, le processeur utilise des adresses logiques, qui sont des positions relatives au début de la région.

1.3.2 Matériel pour la génération d'adresses

Pour calculer l'adresse physique qui correspond à une adresse logique en mode normal (génération d'adresse), on a besoin de

- un registre "base" qui contient l'adresse physique du début de la région,
- un additionneur entre adresse logique et registre de base, qui génère l'adresse physique ;

et pour la protection mémoire :

- un registre qui indique la *taille* de la région,
- un comparateur entre adresse logique et taille de la région.

Comme précédemment, ces registres sont manipulés par des instructions privilégiées.

1.4 Mémoire paginée

Pour ne pas avoir à compacter la mémoire.

Comme précédemment, le mode utilisateur utilise des adresses logiques. Une adresse logique est formé deux parties : un numéro de page (bits de poids fort), et une position (offset, déplacement) sur les bits de poids faibles dans la page.

Exemple, avec des pages de 4k (2 puissance 12) l'adresse logique `0x2345` (en hexa) est dans la page `0x2`, position `0x345`.

```
hex          2    3    4    5
bin        0010 0011 0100 0101
         ======  ============== 12 bits de droite
          0x2      0x345
```

A chaque page de l'espace logique d'un processus correspond à une page physique, dont le numéro est donné par une table de correspondance.

Si la **table des pages** contient `[ 0x12, 0x34, 0x56, 0x78 ]` l'adresse physique est `0x56345`, parce que la page logique 0x2 se trouve dans la page physique 0x56.

L'espace physique occupé par un processus n'est plus contigu. Pour allouer de l'espace, on n'a plus besoin de compacter, il suffit de prendre les pages inutilisées.

Matériel nécessaire : les ordinateurs qui utilisent la pagination ont un circuit "MMU" (memory management unit) qui génère les adresses réelles à partir d'une adresse logique et d'une table des pages.

Remarque

Inconvénient cité traditionnellement : on alloue par pages entières, donc en moyenne on gaspille 1/2 page par processus. Ça pouvait être beaucoup dans les années 60.

Maintenant si on chiffre en 2020 : poste de travail sous Linux, environ 250 processus. Avec des pages de 16 Ko, la perte est de 8Ko x 250 = 2 Mo, négligeable sur 8 Go de mémoire.

1.5 Mémoire Virtuelle

Exploite l'idée de la pagination, et la remarque qu'on n'a pas besoin de toutes les pages en mémoire tout le temps.

Idée : on peut sauvegarder sur disque les pages dont on ne sert pas, ça libérera de la place en mémoire. On les ramènera quand on en aura besoin.

Notion de Working set : les pages dont on a besoin à un moment donné, et probablement bientôt. Notion de localité dans le temps, et dans l'espace.

TODO,

- détailler le mécanisme de remplacement
- stratégies d'éviction de pages.