Systèmes d'exploitation, 2ème année

Mémoires

Yves Stadler

Université Paul Verlaine - Metz

20 novembre 2011

Agenda

Plan du cours

- Utilisation
- Répartition
- Mémoire swap
- Mémoire paginée
- Segmentation
- Segmentation et pagination

Utilisation de la mémoire

Que stocke la mémoire

- Variables, appels aux fonctions, calculs, code !

Comportement de la mémoire

- La mémoire est une zone de stocakge adressable.
- La mémoire n'a aucune idée d'où viennent les données
- La mémoire n'a aucune idée de l'utilisation des données

Problèmes

Organisation de la mémoire

- Est-ce qu'un programme peut utiliser toute la mémoire ?
- Est-ce que la mémoire est partagée ?
- Si oui comment ? Morceaux fixes ? Variables ? Stockée où ?

Manipulations

- Peut-on protéger la mémoire ?
- Comment adresse plus de 64KB ? (32bits = 64K adresses)
- Comment accéder à plus de mémoire que la RAM ?

Type de mémoires

Support de stockage

- Registres (volatile)
- Mémoire cache (volatile)
- RAM (volatile)
- Disques (permanente)
- Bande magnétiques (permanente)

Avantages

- Plus on descend dans la liste plus on perd on rapidité d'accès
- Plus on descend dans la liste plus on gagne en capacité de stockage

Protection de la mémoire

Idée

- On ne veux pas qu'un programme étrange puisse manipuler notre mémoire
- Le noyaux ne veux pas que l'on manipule se mémoire.

Réalisation

- Notion d'adresse privée : adresse logiques.
- Notion d'adresse physique : emplacement réel
- Le programme ne manipule que des adresses logiques
- Le CPU manipule des adresses physiques

Allocation continue

Principe

- Un seul processus en mémoire à un instant donné
- Très bien pour les architecture mono-programmation
- Pour les autres, il faut utiliser le swap

Swap

- Swap : alternance de la présence de donnée en mémoire centrale
- Swap-in : Chargement de la mémoire d'un processus en mémoire centrale
- Swap-out : Déchargement de la mémoire d'un processus (généralement en mémoire cache, ou sur disque)

Partage mémoire

Principe

- On place les processus dans la mémoire de manière séquentielle.
- Un programme à donc une limite de mémoire utilisable : de la première adresses à la première adresse du programme suivant.
- Les limites peuvent être fixées.
- Où charger les processus en mémoire ?

Algorithmes de placement

- First-fit : premier bloc disponible de taille suffisante.
- Best-fit : Plus petit bloc de taille suffisante
- Worst-fit : Bloc qui permet de garder l'espace lire le plus grand

Partage mémoire

Problème : fragmentation

- Allouer, puis libérer de la mémoire créer de la fragmentation
- On peut se retrouver avec assez de mémoire en absolu, mais pas assez d'espace continu pour allouer un nouveau processus.

Solution : compactage

- Les adresses physiques peuvent changer !
- Il faut tenir à jour une correspondance des adresses.

Pagination

Concept

- Mémoire en tranches : cadres de pages
- Dans une page : adresse logique, relative au zéro de cette page.
- Pages numérotées
- Chaque page à une adresse dans la mémoire physique
- Correspondance numéro de page/adresse page : la table de pages.
- Une page pourra être invalide (pas en mémoire)

Implications

- adresse logique : renvoyée par un malloc ;
- adresse logique est un déplacement (offset) dans la page en cours.
- adresse logique == page# :offset
- adresse physique == @page :offset
- ces conversions sont le travail du MMU memory management unit

Pagination

Pour le programmeur

- Utilisation de grands espaces mémoire : problème d'optimisation

Exemple

- taille page : `128*sizeof(int)`
- `int tab[128][128]`
- Une page peut stocker une ligne
- Comment parcourir le tableau ?

Pagination

Exemple

```
for(i = 0; i < LIGNE; i++) {
    for(j = 0; i < COLONNE; j++) {
        ...tab[i][j]... //Accède une page une fois
    }
}
```

Exemple

```
for(j = 0; i < COLONNE; j++) {
    for(i = 0; i < LIGNE; i++) {
        ...tab[i][j]... //Accède une page i fois
    }
}
```

Pagination

Plan du cours

- Le temps d'accès mémoire est doublé

Plan du cours

- Utilisation des Translation Look-aside-buffers
- Mémoire cache page#->adresse page
- Coût matériel
- le temps d'accès varie en fonction :
  - Du temps d'accès au TBL et mémoire
  - Du taux de présence des pages en TBL

Segmentation

Plan du cours

- L'espace mémoire d'un programme est segmenté
- Segment code, donnée, pile, autres.
- Permet le partage de segment
- Assembleur : pointeurs CS, DS, ES, SS
- Segment à une taille limitée.
- Problème : fragementation -> on peut ajouter la pagination

Mémoire paginée et segmentée

Plan du cours

- Les segments peuvent avoir beaucoup de pages
- Répertoire de tables de pages
- Une adresse virtuelle : (seg ; seg-offset)
- seg indique un numéro de segment dans la table des segments
- L'entrée de la table donne le début du segment en mémoire
- L'entrée donne aussi la limite : si seg-off hors intervalle, segfault.
- seg-off = (page ; page-offset)
- L'entrée segment dispose d'un pointeur vers le début de sa table de page
- On cherche page dans la table pointée
- On trouve l'adresse mémoire correspondante
- On ajoute page-off

Mémoire paginée

Utilité

- Les pages ne sont pas nécessairement en mémoire
- Génération de "fautes de page"
- On doit charger cette page depuis le disque
- Place disponible dans les cadres ?
- Oui : on charger
- Non : on doit changer une page pour celle-ci
- Coûteux : besoin d'un algorithme d'optimisation

Stratégie de remplacement de page

Algorithmes

- FIFO, enlever la plus ancienne
- OPTIMAL, enlever la page qui est moins demandée
- LRU (Last recently used) : page la moins utilisée
- Seconde chance, FIFO avec bit référence : si seconde change, remise en queue avec changement du bit
- NRU, bits Référence/écriture