Travaux Dirigés Informatique
Module Système d'exploitation
Année 2005-2006
TD N°2 : Gestion de la mémoire et des fichiers

Pierre.Hyvernat@univ-savoie.fr
Cyril.Vachet@etu.univ-savoie.fr

PARTIE 1 : GESTION DE LA MEMOIRE

Exercice 1 : Partition contiguë et translation de code

Un programme contenant du code translatable a été créé, en partant du principe qu'il serait chargé à l'adresse 0. Dans son code, le programme fait référence aux adresses suivantes : 50, 78, 150, 152, 154. Si le programme est chargé en mémoire en commençant à l'emplacement 250, comme doivent être ajustées les adresses ?

Exercice 2 : Partitions contiguës variables et algorithmes de sélection

Q1 (First Fit) : Sur un système qui utilise l'allocation de la première zone libre, supposons que la mémoire est allouée comme spécifiée dans la figure 1, avant que d'autres requêtes de 20 Ko, 10 Ko, et 5 Ko (dans cet ordre ci) soient reçues. A quelle adresse de départ vont être allouées chacune des autres requêtes ?

Utilisé	Libre	Utilisé	Libre	Utilisé	Libre	Utilisé	Libre	Utilisé	Libre	Utilisé	Libre
10 Ko	10 Ko	20 Ko	30 Ko	10 Ko	5 Ko	30 Ko	20 Ko	10 Ko	15 Ko	20 Ko	20 Ko

Figure 1 : Allocation de la mémoire à partitions variables

Q2 (Best Fit) : Sur un système qui utilise l'allocation du meilleur ajustement, supposons que la mémoire est allouée comme spécifiée dans la figure 1, avant que d'autres requêtes de 20 Ko, 10 Ko, et 5 Ko (dans cet ordre ci) soient reçues. A quelle adresse de départ vont être allouées chacune des autres requêtes ?

Q3 (Worst Fit) : Sur un système qui utilise l'allocation du pire ajustement, supposons que la mémoire est allouée comme spécifiée dans la figure 1, avant que d'autres requêtes de 20 Ko, 10 Ko, et 5 Ko (dans cet ordre ci) soient reçues. A quelle adresse de départ vont être allouées chacune des autres requêtes ?

Q4 (Next Fit) : Sur un système qui utilise l'allocation de la zone libre suivante, supposons que la mémoire est allouée comme spécifiée dans la figure 1, avant que d'autres requêtes de 20 Ko, 10 Ko, et 5 Ko (dans cet ordre ci) soient reçues. A quelle adresse de départ vont être allouées chacune des autres requêtes ?

Exercice 3 : Zones Siamoises (Buddy System)

Sur un système doté de 1 Mo de mémoire et qui utilise le système de zones siamoises, dessinez un schéma qui illustre l'allocation de la mémoire après chacun des évènements suivants :

(a) Processus A, requête 50 Ko
(b) Processus B, requête 150 Ko
(c) Processus C, requête 60 Ko
(d) Processus D, requête 60 Ko
(e) Processus E, requête 60 Ko
(f) Processus D, sortie
(g) Processus C, sortie
(h) Processus E, sortie
(i) Processus A, sortie
(j) Processus F, requête 125 Ko
(k) Processus G, requête 150 Ko
(l) Processus F, sortie
(m) Processus G, sortie
(n) Processus B, sortie

Exercice 4 : Segmentation

Sur un système utilisant la segmentation simple, calculez l'adresse physique de chacune des adresses logiques, à partir de la table des segments ci-après. Si l'adresse génère un défaut de segment, indiquez le.

(a) Segment : 0 -> Offset : 99
(b) Segment : 2 -> Offset : 78
(c) Segment : 1 -> Offset : 265
(d) Segment : 3 -> Offset : 222
(e) Segment : 0 -> Offset : 111

Segment	Base	Longueur

0	330	124
1	810	211
2	111	99
3	498	302

Exercice 5 : Mémoire Virtuelle et Pagination

Dans ce problème, utilisez des valeurs décimales, une taille de page de 2000 octets et la table de pages suivantes

Page	Adresses	In / Out	Cadre
0	0 - 1999	In	20
1	2000 - 3999	Out	22
2	4000 - 5999	In	200
3	6000 - 7999	In	150
4	8000 - 9999	Out	30
5	10000 - 11999	Out	50
6	12000 - 13999	In	120
7	14000 - 15999	In	101

Parmi les adresses virtuelles suivantes, laquelle génère un défaut de page ? Pour celles qui ne génèrent pas de défaut de page, quelle est leur adresse physique après translation ?

(a) 10451
(b) 5421
(c) 14123
(d) 9156

Exercice 6 : Algorithmes de remplacement de pages

Question 1 : Sur un système avec quatre cadres de pages, le tableau suivant indique la page, le date de chargement, la date de la dernière référence, le bit de modification et le bit de référence.

Page	Date de chargement	Date Dernière Référence	Bit de modification	Bit de référence
0	167	374	1	1
1	321	321	0	0
2	254	306	1	0
3	154	331	0	1

(a) Quelle page va remplacer l'algorithme FIFO ?
(b) Quelle page va remplacer l'algorithme LRU ?
(c) Quelle page va remplacer l'algorithme NRU ?
(d) Quelle page va remplacer l'algorithme de la seconde chance ?

Question 2 : Compte tenu des références aux pages suivantes dans un programme :

0,9,0,1,8 1,8,7,8,7 1,2,8,2,7 8,2,3,8,3

Combien de défauts de page vont se produire si le programme possède trois cadres de page disponibles et qu'il y a recours :

(a) Au remplacement FIFO ?
(b) Au remplacement LRU ?

Au remplacement optimal ?

PARTIE 2 : GESTION DES FICHIERS

Exercice : Représentation d'un I-Node

1.1. *Définition* (Merci Wikipédia, l'encyclopédie libre ;))

*Les **inodes** (contraction de « index » et « node »; en français, nœud d'index) sont des structures de données qui contiennent des informations à propos des fichiers stockés dans les systèmes de fichiers de type Linux/Unix. À chaque fichier correspond un numéro d'inode (i-number) dans le système de fichiers dans lequel il réside, unique au périphérique sur lequel il est situé. Les inodes fournissent des informations importantes sur les fichiers telles que le propriétaire et l'appartenance au groupe, les modes d'accès (lecture, écriture, exécution) et son type.*

Les inodes sont créées lors de la création du système de fichiers. La quantité d'inodes (déterminée lors du formatage et dépendant de la taille de la partition) indique le nombre maximum de fichiers que le système de fichiers peut contenir.

*Chaque inode contient environ **64 champs**, dont 13 d'entre eux contiennent des blocs d'adressage et de données, tel que nous allons le voir ci-dessous.*

1.2. Blocs de données et d'adresses, indirections et taille maximale d'un fichier

Dans les 64 champs du inode se trouvent 13 champs dont la fonction est de pointer soit vers des blocs de données, soit vers des blocs d'adresses pointant vers des blocs de données. Voyons ce concept un peu plus en détails.

Les 10 premiers champs (sur les 13) contiennent les adresses des 10 premiers blocs de données du fichier (à raison d'une adresse par bloc). Si les blocs sur lesquels pointent les 10 premiers champs sont suffisants pour contenir le fichier, tout est parfait. Dans le cas contraire, on doit utiliser les blocs 11, 12 et 13, et utiliser les systèmes d'indirection. Il existe trois "niveaux" d'indirection : la simple indirection (champ 11), la deuxième indirection (champ 12), et la troisième indirection (champ 13). Plus le niveau d'indirection est élevé, plus le nombre final de blocs de données sur lequel pointe le champ (11, 12 ou 13) sera élevé. Chacun de ces trois champ pointe vers un bloc d'adresses, qui pourra pointer vers un ou plus blocs d'adresse ou de données.

Question 1 : Schématiser la structure d'un i-node se focalisant sur les 13 champs dont la fonction est de pointer vers des blocs de données et d'adresses.

Pour la suite, nous supposerons que les blocs d'adresses (et de donnée) ont comme taille 1024 octets (1 Ko). Les blocs d'adresses numérotés sur 4 octets, ce qui limite le nombre d'adresses à 256 par blocs (1024 / 4 = 256)

Question 2 : Supposons que nous ayons un gros fichier ne pouvant pas être contenu dans 10 blocs de données. Expliquez rapidement l'ensemble des cas possible.

Question 3 : Calculer la taille maximale d'un fichier en prenant en compte les suppositions précédentes.