info502 : Système d'exploitation
TD 2 : gestion de la mémoire

Thibault Carron et Pierre Hyvernat
Thibault.Carron@univ-savoie.fr
Pierre.Hyvernat@univ-savoie.fr

Exercice 1 : Allocation de mémoire

On suppose que l'état de la mémoire RAM est décrit par le tableau suivant :

10	10	**20**	30	**10**	5	**30**	20	**10**	15	**20**	20

(Les tailles sont en Ko, et les blocs en **gras** sont utilisés, alors que les autres sont libres.)
Des requêtes d'allocation de mémoire arrivent dans cet ordre là : 20 Ko, 10 Ko, 5 Ko et 25 Ko.

Question 1. À quelles adresses sont alloués les blocs si on utilise la politique "First Fit" ?

Question 2. À quelles adresses sont alloués les blocs si on utilise la politique "Best Fit" ?

Question 3. À quelles adresses sont alloués les blocs si on utilise la politique "Worst Fit" ?

Question 4. À quelles adresses sont alloués les blocs si on utilise la politique "Next Fit" ?

Question 5. Pour chacune de ces politiques, chercher un exemple de demande d'allocation / desallocation qui est visiblement inefficace.

Question 6. En partant d'une mémoire libre de 1 Mo, utilisez le système des zones siamoises ("buddy system") pour allouer la mémoire des processus suivants :
- processus A, requête de 50 Ko
- processus B, requête de 150 Ko
- processus C, requête de 60 Ko
- processus D, requête de 60 Ko
- processus E, requête de 60 Ko
- processus D, fin
- processus C, fin
- processus E, fin
- processus A, fin
- processus F, requête de 125 Ko
- processus G, requête de 150 Ko
- processus F, fin
- processus G, fin
- processus B, fin

(Rappel : 1 Ko = 2^{10} o = 1024 o et 1 Mo = 2^{20} o = 1024 Ko.)

Exercice 2 : Mémoire virtuelle, pagination

Pour simplifier les "calcul", nous allons utiliser des pages de taille 2000 octets. (Normalement, la taille d'une page serait une puissance de 2 : 2048 octet dans notre cas...)

Question 1. La table de pages est la suivante

Page	Adresse virtuelle	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, lesquelles génèrent un defaut de page ?
- 10451
- 5421
- 14123
- 9156

Pour celles qui ne génèrent pas de defaut de page, quelle est l'adresse physique référencée ?

Question 2. On suppose que le système ne comporte que quatre cadres de page :

Page	**Chargement**	**Dernière référence**	**Modification**	**Référence**
0	167	374	1	1
1	321	321	0	0
2	254	306	1	0
3	154	331	0	1

Quelle page serait remplacée par :
- l'algorithme FIFO
- l'algorithme LRU ("Least Frequently Used")
- l'algorithme NRU ("Not Recently Used")
- l'algorithme de la deuxième chance

Question 3. Un programme possède 3 cadres de page et fait référence au pages suivantes :

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 sont générés si on utilise
- le remplacement FIFO
- le remplacement LRU
- le remplacement optimal ?

Question 4. (Paradoxe de Belady)

On suppose qu'un programme référence les pages suivantes :

3, 2, 1, 0, 3, 2, 4, 3, 2, 1, 0, 4

En utilisant le remplacement FIFO, combien de défauts de pages sont générés si on dispose de
- trois cadres de page
- quatre cadres de page ?

Qu'en pensez-vous ?

Exercice 3 : inode et système de fichiers EXT2/3

D'après wikipedia (`http ://fr.wikipedia.org/wiki/Inode`) :

> Ext2 est un système de fichiers courant sous Linux, bien que maintenant souvent remplacé par Ext3.
>
> Chaque inode contient environ 64 champs, dont 13 d'entre eux contiennent des blocs pouvant être de deux types :
> - Des blocs d'adresses, qui contiennent des pointeurs vers d'autres blocs ;
> - Des blocs de données, qui contiennent les données du fichier.
>
> 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, les champs 11, 12 et 13 ne sont pas utilisés.
>
> Dans le cas contraire, en plus des 10 premiers blocs, les blocs 11, 12 et 13 sont utilisés. Ces blocs fonctionnent selon un système d'indirection. Il existe trois niveaux d'indirection :

- La simple indirection, utilisée par le champ 11 ;
- La double indirection, utilisée par le champ 12 ;
- La triple indirection, utilisée par le 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. Ce système permet donc aux fichiers d'avoir une taille considérable.

De manière concrète, chacun de ces trois champs pointe vers un bloc d'adresses, qui pourra pointer vers un ou plus blocs d'adresses ou de données. En supposant que les blocs ont comme taille 1024 octets (1 Kio), et que chaque adresse (dans le cas d'un bloc d'adresses) est stockée sur 32 bits (4 octets), chaque bloc d'adresses en contiendra 256. Avec ces informations en main, il est possible de calculer la taille maximale d'un fichier.

Question 1. Représentez de manière graphique un inode avec les blocs de données correspondants.

Question 2. que se passe-t'il quand un fichier est trop grand pour être contenu dans les 10 blocs de données adressés par les champs de 1 à 10 ?

Quelle est la taille d'un tels fichier ?

Question 3. Quelle est la taille maximale d'un fichier valide pour le système EXT2 ?