Université Côte d'Azur
Faculté des Sciences - Département d'Informatique

Licence d'Informatique L2
Introduction aux Systèmes et Réseaux

TD n ° 8 (Corrigé) : Gestion de la mémoire

Rappel sur l'allocation de ressources

Rappelons qu'un processus représente l'exécution d'un programme. Pour faire exécuter un ensemble de processus sur un ordinateur, il faut partager la mémoire et le processeur entre ces processus. Pour cela, on utilise la notion de *ressource virtuelle* : une ressource virtuelle est une image d'une ressource réelle. Pour permettre l'exécution effective d'un processus, le système d'exploitation matérialise les ressources virtuelles par les ressources physiques correspondantes (on dit qu'il alloue les ressources physiques).

Dans un premier temps, nous considérons deux ressources : la mémoire et le processeur. On examinera plus tard les communications entre processus, les fichiers et les entrées-sorties.

— Sur un système 32 bits, chaque processus dispose d'une *mémoire virtuelle* dont la taille est (virtuellement) égale à la capacité d'adressage du processeur, soit 2^{32} octets (4 giga-octets) pour une taille d'adresse de 32 bits. Il faut bien faire attention ici à la signification du mot virtuel : lorsque l'on dit que la taille est *virtuellement* de 4 giga-octets, c'est un peu comme lorsque l'on dit que l'est virtuellement millionaire après avoir acheté un ticket de loto : c'est certes une possiblité, mais elle ne deviendra *réalité* que très rarement. C'est au système d'exploitation d'allouer au processus la mémoire physique correspondante, à mesure de ses besoins. Lorsque le processus n'est pas actif, le contenu de sa mémoire virtuelle est recopié sur le disque pour pouvoir allouer à d'autres processus la mémoire physique qu'il occupe. Nous n'examinons pas les détails de la gestion de la mémoire virtuelle.

— Le processeur est alloué tour à tour aux processus. Pour respecter l'équité entre processus, chacun d'eux reçoit le processeur pendant une tranche de temps fixée appelée *quantum*, dont la durée est de l'ordre de 10 à 50 millisecondes. Néanmoins, lorqu'un processus doit attendre (par exemple parce qu'il lit un fichier sur disque), il ne peut pas utiliser le processeur. Il doit alors le céder à un autre processus.

1 Segmentation

On considère la table des segments suivante pour un processus P1 :

segment	base	longueur
0	540	234
1	1254	128
2	54	328
3	2048	1024
4	976	200

Calculer les adresses réelles (physiques) correspondant aux adresses virtuelles (logiques) suivantes, ou signaler les éventuelles erreurs d'adressage qui engendreront un déroutement puis le postage à P1 d'un signal du type *segmentation fault* :
$(0, 128), (1, 100), (2, 465), (3, 888), (4, 100), (4, 344)$

Rappel : Les adresses données sont de la forme (segment, offset) où 'segment' correspond au numéro du segment et 'offset' au déplacement dans le segment (numéro d'octet dans le segment). L'adresse physique s'obtient en ajoutant l'adresse de base du segment au déplacement dans le segment, à condition que le déplacement ne soit pas supérieur à la taille du segment moins 1 (car on compte le déplacement en partant de 0).

- *(0,128) : Le segment 0 est long de 234 octets, donc le déplacement 128 est valide. L'adresse physique est* $\Phi = @base + offset = 540 + 128 = 668$.
- *(1,100) : valide car* $100 < 128$ *donc* $\Phi = 1254 + 100 = 1354$.
- *(2,465) : invalide car* $465 > 328$ *!*
- *(3,888) : valide car* $888 < 1024$ *donc* $\Phi = 976 + 888 = 1864$
- *(4,100) : valide car* $100 < 200$ *donc* $\Phi = 976 + 100 = 1076$
- *(4,344) : invalide car* $344 > 200$.

Question : (4,200) est-elle valide ?

Non car dans un segment de longuer 200, les déplacement valides sont dans l'intervalle $[0 - 199]$.

2 Pagination

1. Dans un système paginé donné, les pages font 256 mots mémoire et chaque processus est autorisé à utiliser au plus 4 cadres de la mémoire centrale pour stocker ses pages. On considère la table des pages suivante du processus P1 :

page	cadre
0	3
1	i
2	0
3	i
4	i
5	i
6	5
7	1

i signifiant que la page n'est pas chargée.

Sachant que l'espace d'adressage du processus est l'espace d'adressage virtuel formé par les pages du processus, quelle est la taille de l'espace d'adressage du processus P1 ?

Comme il y a 8 pages, la taille de l'espace virtuel est de $8 \times 256 = 2048$ *mots.*

Calculer les adresses réelles correspondant aux adresses virtuelles suivantes émises par P1 au cours de son exécution :
$240, 546, 1578, 2072$

Rappel : Les adresses réelles, par opposition aux adresses virtuelles, sont les adresses dans la mémoire physique. La conversion d'une adresse virtuelle en adresse réelle est réalisée par le MMU (Memory Management Unit) de la façon suivante :

(a) Calcul du numéro de la page et du déplacement dans la page. Le premier correspond au résultat p de la division entière de l'adresse virtuelle v par la taille c d'une page, et le second correspond au reste d de cette division (le modulo).

(b) On recherche dans la table de pages l'entrée qui correspond à la page p de façon à en déduire le numéro du cadre C.

(c) On obtient l'adresse physique (réelle) Φ en ajoutant le déplacement d à l'adresse physique de début du cadre $C_0 = C \times c$: $\Phi = C_0 + d$

Voici le détail des calculs pour les adresses demandées :

— $240 = 0 \times 256 + 240$ *: $p = 0$ et $d = 240$. D'après la table $C = 3$ et donc $C_0 = 3 \times 256$. Donc* $\Phi = 3 \times 256 + 240 = 1008$
— $546 = 2 \times 256 + 34$ *donc $C = 0$ et* $\Phi = 0 \times 256 + 34 = 34$.
— $1578 = 6 \times 256 + 42$ *donc $C = 5$ et* $\Phi = 5 \times 256 + 42 = 1322$.
— *2072 est en dehors de l'espace d'adressage virtuel du processus (2048 mots). Une référence à cette adresse provoquera donc une erreur fatale (via un déroutement).*

Supposons que P1 continue son exécution en générant comme adresse virtuelle : 770. Que se passera-t-il ?

$770 = 3 \times 256 + 2$. *Il s'agit d'une adresse située dans la page 3. Or d'après la table des pages, cette page n'est pas présente en mémoire. Une référence à cette adresse provoquera donc un déroutement de type "défaut de page". Ce déroutement n'est pas une erreur fatale. Il va déclencher l'exécution par le noyau de l'algorithme de pagination, qui va remplacer l'une des 4 pages présente en mémoire par la page manquante (chargement à partir du disque dur).*

2. Reprenez l'exemple de gestion mémoire paginée donné dans le cours "Gestion de la Mémoire" dont le titre du transparent est *Application pratique p. 32-34).*

Considérez l'adresse virtuelle suivante : `0000, 000000000111`. Dites comment s'écrira l'adresse réelle, en binaire.

Les 4 bits de poids fort désignent le numéro page. D'après la table de pages (p. 33 du cours), cette page se trouve dans le cadre 010. L'adresse physique s'obtient donc simplemnt en substituant au 4 bits de poids fort de l'adresse virtuelle les 3 bits du numéro de cadre :
`010 000000000111`.

Toujours en vous référant à ce transparent, combien de cadres (et donc de quantité de mémoire vive) dispose l'ordinateur pris en exemple ?

Comme les cadres sont numérotés seulement sur 3 bits, on peut raisonnablement supposer qu'il n'y a que $2^3 = 8$ *cadres et donc seulement* $8 \times 2^{12} = 2^{15} = 32768$ *mots (de 16 bits) de mémoire physique.*

3. Un programme a un espace virtuel de 600 mots. On considère la suite des adresses virtuelles :
   $34, 123, 145, 510, 456, 345, 412, 10, 14, 12, 234, 336, 412$.

   (a) Donner la suite des numéros de pages référencés, sachant qu'elles comportent 100 mots.

   Il suffit de diviser l'adresse par 100, ce qui donne la liste suivante :
   0, 1, 1, 5, 4, 3, 4, 0, 0, 0, 2, 3, 4

   (b) Le programme dispose de 300 mots en mémoire centrale répartis dans 3 cadres (on a donc toujours 100 mots par page). En supposant la mémoire initialement vide, représenter pour les algorithmes OPT, FIFO (First Page In, First Page Out), LRU (Least Recently Used), FIFO de la seconde chance (FINUFO) et LFU (Least Frequently Used), l'occupation des 3 cadres au cours du temps.

   En déduire pour chacun d'eux le taux de défauts de page.

   Les tableaux suivants représentent l'occupation des 3 cadres au cours du temps. Les défauts de pages sont signalés sur la dernière ligne du tableau. Pour FIFO, le signe + désigne la tête de la FIFO. Pour OPT il désigne la ou les pages qui seront référencées le plus tard (ou jamais plus) dans le futur. Pour LRU, il désigne le cadre contenant la page la moins récemment accédée.

FIFO													
refs :	0	1	1	5	4	3	4	0	0	0	2	3	4
C_1	+0	+0	+0	+0	4	4	4	+4	+4	+4	2	2	2
C_2	-	1	1	1	+1	3	3	3	3	3	+3	+3	4
C_3	-	-	-	5	5	+5	+5	0	0	0	0	0	+0
def ?	D	D	-	D	D	D	-	D	-	-	D	-	D

OPT													
refs :	0	1	1	5	4	3	4	0	0	0	2	3	4
C_1	+0	0	0	0	0	0	0	0	0	+0	+2	+2	+2
C_2	-	+1	+1	+1	4	4	+4	+4	+4	4	4	4	+4
C_3	-	-	-	+5	+5	+3	3	3	3	3	3	+3	+3
def ?	D	D	-	D	D	D	-	-	-	-	D	-	-

LRU													
refs :	0	1	1	5	4	3	4	0	0	0	2	3	4
C_1	+0	+0	+0	+0	4	4	4	4	4	4	+4	3	3
C_2	-	1	1	1	+1	3	3	+3	+3	+3	2	2	+2
C_3	-	-	-	5	5	+5	+5	0	0	0	0	+0	4
def ?	D	D	-	D	D	D	-	D	-	-	D	D	D

FIFO 2de chance													
refs :	0	1	1	5	4	3	4	0	0	0	2	3	4
C_1	$+0_1$	$+0_1$	$+0_1$	$+0_1$	4_1	4_1	4_1	$+4_1$	$+4_1$	$+4_1$	2_1	2_1	$+2_1$
C_2	-	1_1	1_1	1_1	$+1_0$	3_1	3_1	3_1	3_1	3_1	$+3_0$	3_1	3_1
C_3	-	-	-	5_1	5_0	$+5_0$	$+5_0$	0_1	0_1	0_1	0_0	$+0_0$	4_1
def ?	D	D	-	D	D	D	-	D	-	-	D	-	D

4. On supposera dans tout l'exercice que les adresses virtuelles sont sur 32 bits.

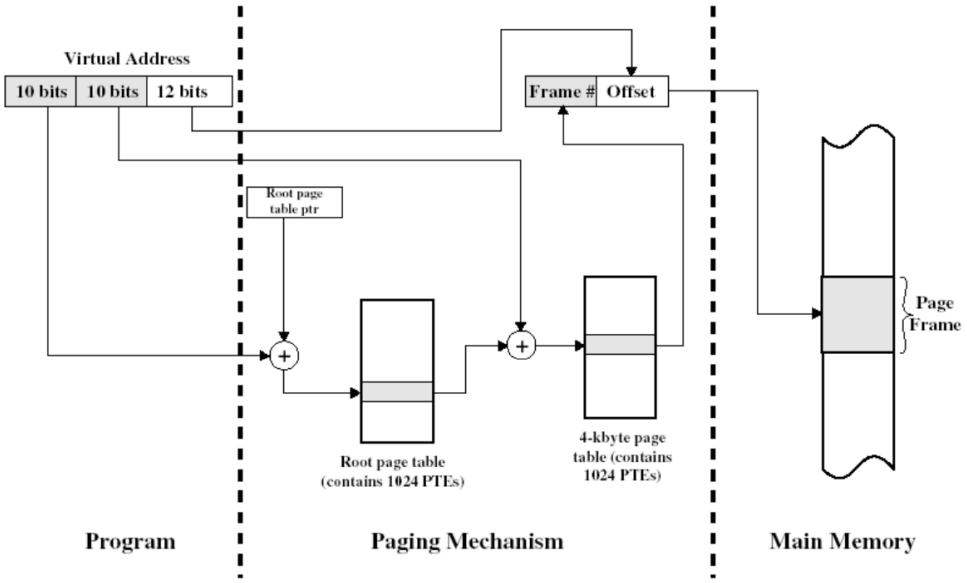
   (a) Si la gestion mémoire est paginée avec des pages de taille $8Ko$, identiques à celle des cadres, combien d'entrées contient la table des pages d'un processus ? Même question pour des pages de $4Ko$.

   $8192 = 2^{13}$ *: sur les 32 bits d'une adresse, 13 sont utilisés pour représenter le dépacement dans la page. Il reste donc 19 bits pour répresenter le numéro de page. La table de page doit donc pouvoir référencer jusqu'à* $2^{19} = 512K$ *pages. En supposant que les adresses physiques sont aussi codées sur 32 bits, un calcul identique nous montre que la mémoire physique peut comporter jusqu'à* 2^{19} *cadres différents, et dans ce cas, chaque entrée de la table devrait occuper 19 bits. La taille totale de la table serait donc de* $2^{19} \times 19$ *bits, soit environ 1Mo par processus. Mais comme il est bcp plus efficace d'aligner les données sur des puissances de 2, chaque adresse de 19 bits sera stockée sur 32 bits. Donc* $2^{19} * 32 = 2^{24} bits = 2Mo$. *Ce qui permet en outre de laisser des bits inutilisés qui seront utilisés comme drapeaux (r/w, accessed, present, ...)*

   Pour des pages de $4Ko$, *12 bits permettent de se déplacer dans la page, les tables référence* 2^{20} *pages, et chaque page est identifiée par un mot de 20 bits.* $2^{20} * 20 = 2.5Mo$. *En pratique* $2^{20} * 32 bits = 4Mo$

   (b) Soit un système de gestion mémoire paginée à deux niveaux, c'est-à-dire dans lequel la table des pages n'est pas d'un seul tenant mais se décompose en une table de tables des pages. L'intérêt de ceci est d'avoir à charger en mémoire lorsque le processus s'éxécute uniquement la table des pages de premier niveau et la table des pages de 2ème niveau active.

   Dans notre exemple les adresses virtuelles se décomposent en un champ de 10 bits servant d'index dans la table des pages de premier niveau, puis d'un autre champ de 10 bits servant d'index dans la table des pages de deuxième niveau, les 12 bits restant tenant lieu de déplacement dans la page. Cette description est très similaire au mode normal de gestion des pages sur une architecture intel 32 bits, dans un système tel que Linux.



On cherche quelle est la taille des pages, et par conséquent le nombre de pages de l'espace virtuel d'un processus.

$2^{10} = 1024$ *entrées dans la table de niveau 1. 1024 tables de niveau 2 de* $2^{10} = 1024$ *entrées. On a donc toujours* 2^{20} *pages de taille* $2^{12} = 4Ko$

Quelle est l'occupation mémoire de la table de 1er niveau, de celle de deuxième niveau ?

$2^{10} = 1024$ *entrées dans la table de niveau 1, chaque entrée faisant 10 bits. Donc 10240 bits = 1,25 Ko. La table de niveau 2 fait la même taille.*