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

Licence d'Informatique L2
Introduction aux Systèmes et Réseaux

TD n ° 8/9 : 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).

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

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 ?

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 $C_0 = C \times c$: $\Phi = C_0 + d$

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

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.

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

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.

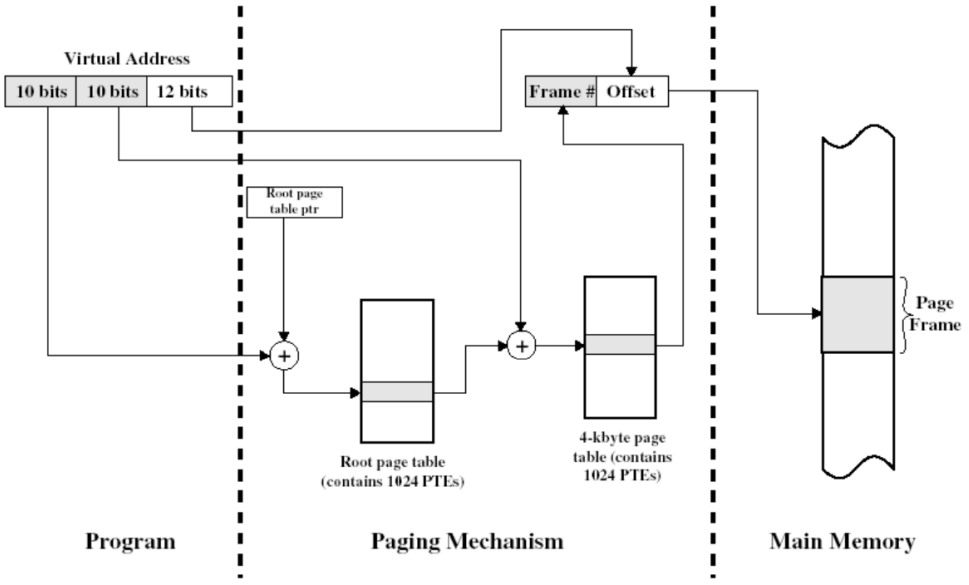
   (b) Le programme dispose de 300 mots en mémoire centrale (répartis dans 3 cadres). Calculer le taux de défauts de page (en supposant la mémoire initialement vide) pour les algorithmes OPT, FIFO (First Page In, First Page Out), LRU (Least Recently Used), FIFO de la seconde chance (FINUFO) et LFU (Least Frequently Used).

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 $4Ko$.

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

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

3 Transformation des adresses paginées

Prenons un système de mémoire paginée (serveur intel 32 bits avec drapeau PAE (*physical address extension* activé) dans lequel les adresses sont codées sur 32 bits et chaque page mesure 2^{12} octets. Le système associe à chaque processus une table des pages comme dans l'exemple ci-dessous :

...
<drapeaux et protection, 100>
<drapeaux et protection, 120>
<drapeaux et protection, 10>
<drapeaux et protection, 30>
...

Chaque entrée de la table des pages est un couple `<drapeaux et protection (8 bits), numéro de page physique (24 bits)>`. Dans ces conditions,

1. quelle est la taille maximum de la mémoire logique d'un processus et la taille maximum de la mémoire physique ?
2. quelles sont les adresses physiques qui correspondent aux adresses logiques 9000, 3000, 12288 et 4100 ?

4 Les algorithmes de remplacement

En cas de défaut de page dans un système paginé, les algorithmes de remplacement doivent déterminer une page physique victime qui sera sauvée sur disque (si nécessaire) et remplacée par la page virtuelle que l'on désire consulter. Il existe principalement quatre stratégies pour choisir la page victime :

1. Algorithme FIFO : la victime est la page physique la plus anciennement chargée.
2. Algorithme LRU (Least Recently Used).
3. Algorithme FINUFO (First In Not Used, First Out) ou algorithme de la 2ème chance.
4. Algorithme LFU (Least Frequently Used).

On vous donne deux listes `dpp` et `dpv` de taille `NB_PP` (nombre de pages physiques) et `NB_PV` (nombre de pages physiques). Les classes décrivant les pages physiques et virtuelles sont décrites ci-dessous.

```
class DPV:
    presente = False # page physique associée ?
    npp = None       # numéro de page physique associée

class DPP:
    libre = False  # non utilisée par un autre processus ?
    npv = None       # page virtuelle associée
    modif = False    # accédée en écriture ?
    info = 0         # usage selon l'algo choisi...
```

4.1 Transformation des adresses

À l'aide de ces déclarations, on vous demande d'écrire la fonction `npv2npp(npv)` qui détermine un numéro de page physique (`npp`) à partir d'un numéro de page virtuelle (`npv`). Elle utilise la fonction `defaut_de_page(npv)` qui fera le gros du travail en traitant l'apparition d'un défaut de page sur la page virtuelle `npv` en chargeant une page physique. `npv2npp` se borne donc à tester si on doit faire un défaut de page, puis à modifier éventuellement certains champs de la page physique correspondant à la page logique. Note : il ne faut pas oublier que cette opération est matérielle, donc pas de gros algorithme coûteux en temps et/ou en silicium.

4.2 Traitement du défaut de page

Écrivez la routine `defaut_de_page(npv)` en utilisant les fonctions et variables ci-dessous.

`sauver_page(npp, npv)` # mém. centrale → mém. secondaire
`charger_page(npv, npp)` # mém. centrale ← mém. secondaire
`choisir_victime()` # choisir une page physique victime
`nb_pp_occupees` # variable globale

4.3 Choix de la victime

La fonction `choisir_victime()` choisit la page physique victime en cas de défaut de page. On vous demande de programmer cette fonction pour les quatre algorithmes décrits précédemment.
Note :

- Pour la version FIFO vous utiliserez une variable globale victime que donne la prochaine page physique victime.
- Pour la version FINUFO vous utiliserez une variable globale finufo qui donne la dernière page physique victime.