Contrôle continu - 19/11/2021

Durée : 1h30
Une feuille A4 R/V manuscrite (cours et/ou td) autorisée.

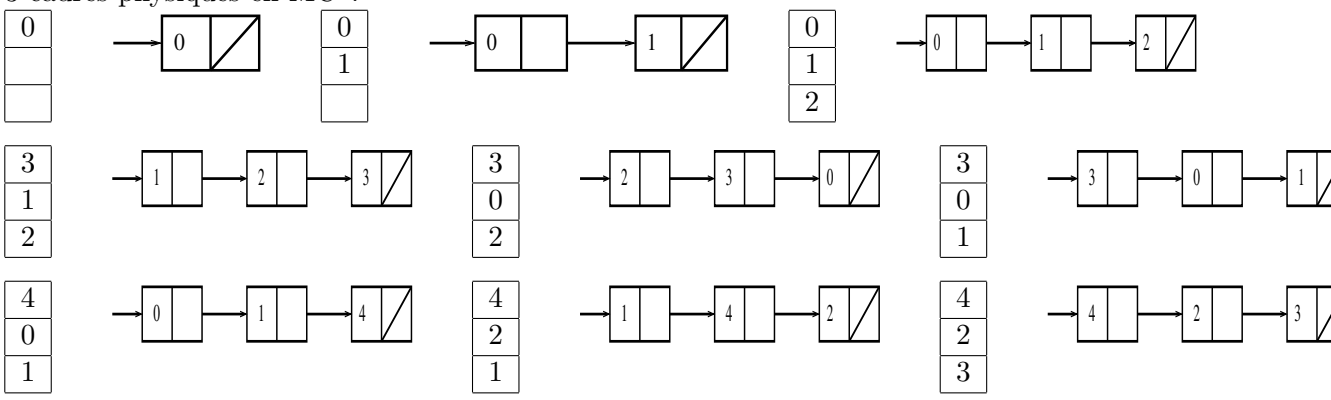
Barème donné à titre indicatif : Ex.1 : 4 - Ex. 2 : 7 - Ex. 3 : 5 - Ex 4 : 4

Exercice 1. Dans le cadre d'une stratégie FIFO, pour un processus accédant à ses pages selon la séquence suivante

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

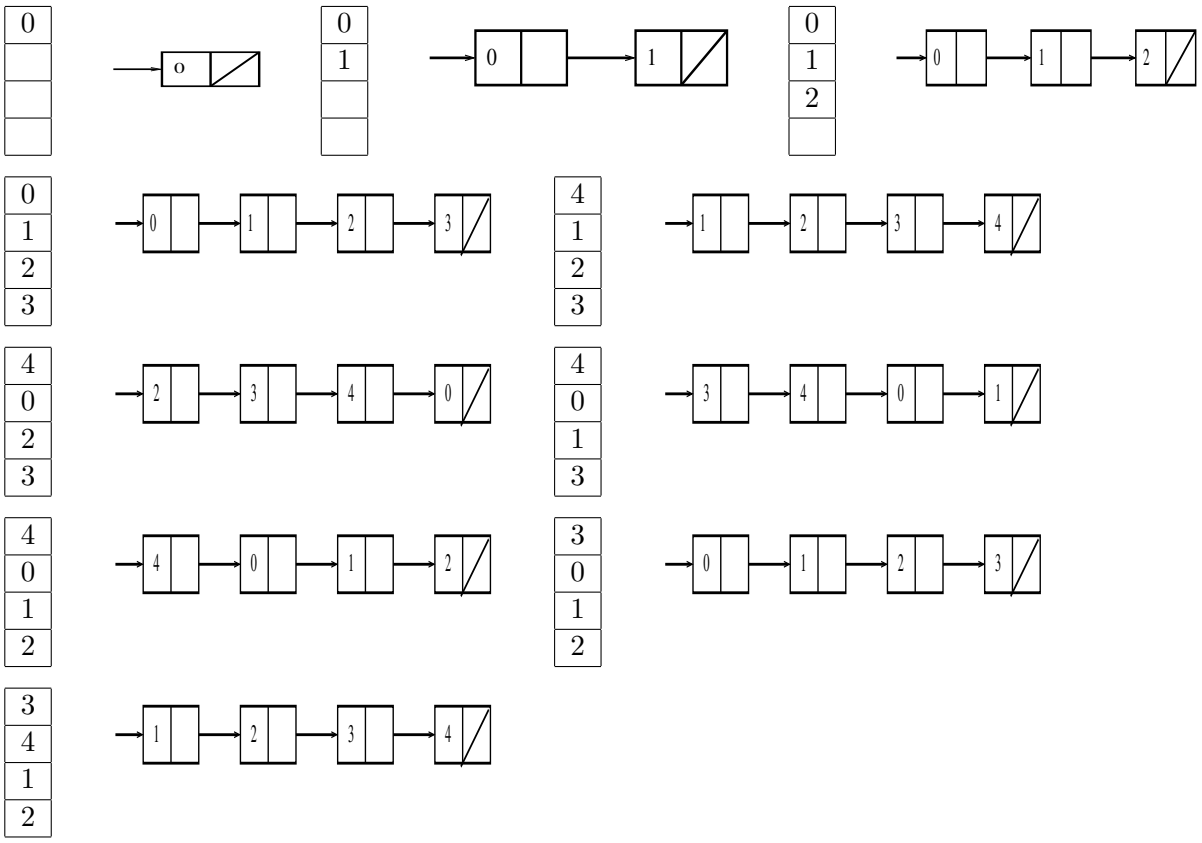
pour un nombre de cadres physiques égal à 3 puis égal à 4, donner l'état de la mémoire centrale à chaque défaut de page, ainsi que l'état de la liste chaînée des pages présentes en mémoire. On suppose qu'initialement, la mémoire est vide. Combien y a-t-il de défauts de page dans chaque cas ? Qu'observez-vous ?

1. 3 cadres physiques en MC :



   Au total, 9 défauts de page.

2. 4 cadres physiques en MC :



Au total, 10 défauts de page.

On a augmenté le nombre de cadres de pages et pourtant, le défaut de pages n'a pas diminué (au contraire, il a augmenté !) : phénomène connu sous le nom d'**anomalie de Belady**.

Exercice 2. On considère un système de mémoire virtuelle ayant les caractéristiques suivantes :

- la taille d'une page (et d'un cadre) est de `1 Ko`
- la taille de la mémoire centrale est de `32 Mo`
- la taille de mémoire virtuelle est de `512 Mo`
- on utilise la technique de la segmentation paginée : l'espace d'adressage virtuel d'un processus est composé de segments contigus. Chaque segment peut contenir entre 1 et 128 pages. La numérotation des pages d'un segment est relative au segment.

1. Calculer le format d'une adresse virtuelle et d'une adresse physique en spécifiant le nombre de bits réservés pour chaque champ.

   Technique de la segmentation paginée $\Rightarrow$ l'adrese virtuelle est sous la forme $\langle$n° segment, n° page dans le segment, déplacement dans la page$\rangle$.

   La page a une taille de `1 Ko` $= 2^{10}$ donc, le déplacement dans la page est sur 10 bits.

   Un segment peut contenir jusqu'à 128 ($= 2^7$) pages donc, 7 bits pour le n° de la page dans le segment.

   Un segment contient 2^7 pages $= 2^7 * 2^{10} = 2^{17}$ octets. La taille de la mémoire virtuelle est de $2^{29}(512Mo)$ donc on a un total de $\frac{2^{29}}{2^{17}} = 2^{12}$ segments. Donc, 12 bits pour représenter le n° de segment.

   Au final, la mémoire virtuelle est la forme

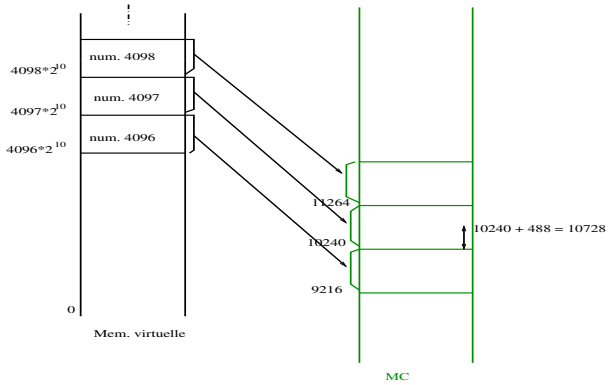
   $$\langle 12\,\text{bits},\ 7\,\text{bits}, 10\,\text{bits},\rangle$$

   Pour l'adresse physique, on a le format $\langle$n° cadre de page, déplacement dans le cadre$\rangle$. La MC a une taille de `32 Mo` $= 2^5 * 2^{20}$ octets $= 2^{25}$ octets. Le cadre physique a la même taille que la page donc sa taille est de 2^{10} octets $\Rightarrow$ on a $\frac{2^{25}}{2^{10}}$ cadres physiques de page i.e. 2^{15} cadres physiques. On a donc besoin de 15 bits pour représenter le n° du cadre. Au final, l'adresse physique a la forme :

   $$\langle 15\,\text{bits}, 10\,\text{bits},\rangle$$

2. Un processus `p` a un segment de code de `9 Ko` et un segment de données de `3 Ko`. Dans l'espace virtuel de `p`, le segment de code est suivi du segment de données.
   Si le segment de code débute à l'adresse 0, à quelle adresse débute celui des données ?
   Un segment de code de `9 Ko` est sur 9 pages de `1 Ko`, il se situe dans l'intervalle des adresses $[0..9 * 2^{10}[= [0..9216[$. Donc, le segment de données débute à l'adresse 9216.

3. Sachant que le segment de données de `p` est chargé totalement en mémoire centrale dans les cadres contigus 4096, 4097 et 4098, calculer l'adresse réelle qu'occupe en mémoire centrale une donnée qui se trouve à l'adresse logique 10728, relative au début de l'espace d'adressage. On demande un entier décimal.
   Le segment de données occupe la MC dans les cadres 4096, 4097 et 4098. Donc, la page 0 occupe le cadre dans l'intervalle d'adresses $[4096 * 2^{10}..4097 * 2^{10}[$, la page 1 les adresses $[4097 * 2^{10}..4098 * 2^{10}[$ et la page 2 les adresses $[4098 * 2^{10}..4099 * 2^{10}[$.
   De 9216 à 10240 (non compris), on a la page 0. La page 1 commence à l'adresse 10240, l'offset de la donnée est égal à $10728 - 10240 = 488$
   En MC, la page 1 du segment de données est chargée à l'adresse $4097 * 2^{10} \Rightarrow 10728$ en MV correspond à l'adresse en MC $= 4097 * 2^{10} + 488 = 4\,195\,816$.



4. Soit la séquence de références de pages suivante :

$$0, 1, 0, 1, 2, 3, 4, 2, 3, 4, 5, 6, 7, 8$$

   effectuée par `p`. Les opérandes référés par les instructions dans les pages 0, 1 et 2 se trouvent dans la page 0_d du segment de données ; les opérandes des instructions des pages 3, 4 et 5 sont dans la page 1_d ; les opérandes des instructions des pages 6, 7 et 8 sont dans la page 2_d. Supposons que toutes les instructions de `p` ont des opérandes qui réfèrent en mémoire.
   Au départ, 4 cadres contigus sont alloués pour le code du processus à l'adresse X et 2 cadres contigus sont alloués pour ses données à l'adresse Y (X et Y ne sont pas nécessairement contigües). Le chargement des pages est réalisé à la demande. De plus, aucun cadre supplémentaire n'est alloué à `p` durant son exécution.

   (a) Représenter l'état d'occupation de la mémoire centrale à chaque instant t_i (c'est-à-dire t_0, t_1, ...) où une nouvelle page est chargée en appliquant l'algorithme de remplacement de pages `LRU`.

t_0	0	0_d	t_1, t_2, t_3	0	0_d	t_4	0	0_d	t_5	0	0_d	t_6, t_7, t_8, t_9	4	0_d	
				1			1			1	1_d		1	1_d	
							2			2			2		
										3			3		
t_{10}	4	0_d	t_{11}	4	2_d	t_{12}	4	2_d	t_{13}	8	2_d				
	5	1_d		5	1_d		5	1_d		5	1_d				
	2			6			6			6					
	3			3			7			7					

   (b) Calculer le nombre de défauts de page généré par l'algorithme `LRU`. Ce nombre est-il optimal ?
   12 défauts de page, 9 pour les pages de code, et 3 pour les pages de données.

Ce nombre est optimal parce que les pages remplacées coïncident avec la stratégie de l'algorithme optimal.

Exercice 3. Un système qui implante la pagination à la demande dispose de 4 cadres physiques qui sont tous occupés, à un instant donné, par des pages de mémoire virtuelle. La table suivante donne, pour chaque case mémoire, le moment $t_{\text{chargement}}$ du chargement de la page qu'elle contient, le temps $t_{\text{dernier\_accès}}$ du dernier accès à cette page, et l'état des bits référence (`R`), modifié (`M`) et présence (`P`). Les temps sont donnés en tops d'horloge.

Case	$t_{\text{chargement}}$	$t_{\text{dernier\_accès}}$	`R`	`M`	`P`
0	126	270	0	0	1
1	230	255	1	0	1
2	110	260	1	1	1
3	180	275	1	1	1

Indiquer quelle page sera remplacée dans le cas d'un défaut de page si l'algorithme de remplacement de pages est :

1. NRU
   NRU remplace dans l'ordre 1) les pages qui ont `R=M=0`, 2) `R=0,M=1`, 3) `R=1,M=0`, 4) `R=M=1`. Donc, c'est la page qui occupe la case 0 qui sera remplacée.
2. FIFO
   Selon le temps de chargement, la première page chargée est celle qui occupe la case 2 ;c'est elle qui sera déchargée.
3. horloge
   Il y a un indicateur sur la page la plus ancienne : la page qui occupe la case 2. Le bit `R` de cette page est à 1. Elle n'est donc pas retenue, mais son bit est mis à 0. La page suivante est celle qui occupe la case 0. Son bit `R` étant à 0, c'est elle qui sera remplacée.

Exercice 4.

1. Quel est l'intérêt de la pagination à double niveau ?
   Elle permet de charger des tables de pages plus petites. Seules les tables de page correspondant aux méta-pages utilisées par le processus seront chargées.
   Par exemple, si on a des tables de pages de `4 Ko`, si toutes les pages utilisées par le processus sont dans le même groupe de pages, alors on ne chargera qu'une table des pages, au lieu de charger la table complète en pagination simple niveau.
2. Qu'est ce qu'une *commutation de contexte* ? Quand a-t-elle lieu ? Quelles sont les opérations qu'elle nécessite ?
   La commutation de contexte est une procédure réalisée par le SE lorsque le processus en cours d'exécution est interrompu, pour une raison quelconque (fin de quantum, demande d'E/S, préemption à l'arrivée d'un processus plus prioritaire etc...).
   - sauvegarde du contexte du processus interrompu (CO, contenu des registres et des variables, liste des fichiers ouverts, etc...),
   - restauration du contexte du processus chargé.