Gestion de la mémoire

Exercice 1 :

Considérez un système disposant de 16 MO de mémoire physique réservée aux processus utilisateur. La mémoire est composée de cases (cadres ou frames) de taille 4 KO. L'espace logique d'un processus est composé de trois segments (le segment de code, le segment de données et le segment de pile). Chaque segment est composé d'une ou plusieurs pages. Lorsqu'un processus demande à être chargé en mémoire, le système tente d'allouer à chaque segment de l'espace logique du processus, une zone contiguë en mémoire, dans l'ordre suivant : le segment de code, le segment de données et le segment de pile.

En cas de succès, chaque segment est chargé dans la zone contiguë allouée au segment. La taille de l'espace alloué à chaque segment est exactement égale au nombre de pages qu'il y a dans le segment (mémoire à partition variable).

Un processus chargé en mémoire y séjournera jusqu'à la fin de son exécution (pas de va-et-vient et pas de relocation).

En cas d'échec, le chargement du processus est retardé et aucun espace n'est alloué au processus jusqu'à la prochaine tentative de chargement en mémoire.

Lorsqu'un processus se termine, le processus libère son espace mémoire et le système tente de charger en mémoire d'autres processus.

Question 1 : Allocation d'espace en mémoire

Considérez les 5 processus du tableau suivant :

Instant d'arrivée	Processus	Taille	Temps
0	A	5 MO + 5 MO + 1 MO	8 ms
2	B	4 MO + 1 MO + 2 MO	10 ms
4	C	1 MO + 1 MO + 1 MO	13 ms
10	D	2 MO + 1 MO + 2 MO	15 ms
11	E	6 MO + 2 MO + 1 MO	6 ms

Par exemple, la première ligne du tableau signifie que le processus A arrive à l'instant 0, son espace d'adressage logique est composé d'un segment de code de 5 MO, d'un segment de

données de 5 MO et d'un segment de pile de 1 MO. Lorsque le processus A est chargé en mémoire, il y séjournera pendant 8 ms.

1) Donnez sous forme de diagrammes de temps l'évolution de l'état de la mémoire et de la file de haut niveau, aux différentes étapes de traitement de ces processus, sous les hypothèses suivantes :

- Le mode d'allocation des trous pour les segments est le premier ajustement (First Fit) ;
- Le répartiteur de haut niveau fonctionne selon PAPS (Premier Arrivé, Premier Servi). Le répartiteur de haut niveau a pour rôle d'ordonnancer les demandesde chargement en mémoire.

2) Cette stratégie d'allocation d'espace souffre-t-elle de fragmentation interne ? De fragmentation externe ? Justifiez votre réponse.

3) Expliquez pourquoi le gestionnaire de la mémoire n'alloue aucun espace à un processus s'il n'y a pas suffisamment d'espace pour les trois segments du processus (la politique du tout ou rien).

Question 2 : Translation d'adresse

Supposez que l'adressage logique est sur 24 bits. Les 2 premiers bits (de poids fort) indiquent le numéro de segment. Les 10 bits suivants donnent le numéro de page. Les 12 derniers bits sont réservés au déplacement dans la page.

L'adresse physique est aussi sur 24 bits. Les 12 bits de poids fort indiquent le numéro de case. Les 12 derniers bits spécifient le déplacement dans la case.

1) Donnez le nombre de cases (cadres ou frames) en mémoire physique.

2) Expliquez comment convertir une adresse logique en une adresse physique.

3) Considérez l'adresse logique suivante:

01	00 0000 0010	0000 0011 0001

Donnez l'adresse physique correspondant à l'adresse logique précédente, si le segment 01 est chargé dans une zone contiguë commençant à la case (cadre ou frame) numéro 3 :

0000 0000 0011

4) Est-il possible de réaliser la translation d'adresse lors du chargement (dites pourquoi) ?

Exercice 2 :

Considérez un système de mémoire virtuelle ayant les caractéristiques suivantes :

- Taille d'une page et d'un cadre (ou cases) = 1 KO (1 kilo-octet).
- Taille de la mémoire physique (principale) = 32 MO (32 méga-octets).
- Taille de la mémoire virtuelle = 512 MO.
- Utilisation combinée des techniques de pagination et de segmentation : 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.
- Utilisation de l'algorithme de remplacement de pages LRU (i.e. la moins récemment utilisée).

1) Calculez le format d'une adresse virtuelle et le format d'une adresse physique (i.e. réelle), en spécifiant le nombre de bits réservés pour chaque champ.

2) Supposez un processus de 9 KO de segment de code et 3 KO de segment de données.

Dans l'espace virtuel du processus, le segment de code est suivi du segment de données. Par conséquent, le segment de code débute à l'adresse 0 alors que celui des données débute à l'adresse 9216 relativement au début de l'espace d'adressage virtuel.

- **Calculez l'adresse** qu'occupe en mémoire principale une donnée se trouvant à l'adresse 10728, relative au début de l'espace d'adressage. Le segment de données du processus est chargé au complet en mémoire physique dans les cadres contigus 4096, 4097 et 4098.

3) Considérez la séquence de références de pages de code R={0, 1, 0, 1, 2, 3, 4, 2, 3, 4, 5, 6, 7, 8} faite par le processus décrit en (2). Les opérandes référés par les instructions dans les pages 0, 1 et 2 se trouvent dans la page 0 du segment de données ; les opérandes des instructions des pages 3, 4 et 5 sont dans la page 1 ; les opérandes des instructions des pages 6, 7 et 8 sont dans la page 2. Supposez que toutes les instructions du processus 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 pour les données du processus à l'adresse Y. Il est à noter que les adresses X et Y ne sont pas nécessairement contiguës, et le chargement des pages dans les cadres alloués est réalisé à la demande (aucun chargement préalable). De plus, aucun cadre supplémentaire n'est alloué au processus durant son exécution.

(a) Représentez l'état d'occupation de la mémoire principale à chaque instant t_i (i.e. t_0, t_1, t_2, …) où une nouvelle page est chargée.
(b) Calculez le nombre de fautes (i.e. défauts) de page générées par l'algorithme de remplacement de pages LRU. Ce nombre est-il optimal ?

Exercice 3 :

Une firme bien connue concurrente de Apple décide de lancer un nouvel appareil qui permet d'écouter de la musique de format mp3 et d'en « stocker » plus de 80 Go, soit 40Go de plus que le fameux « iPod ». Cependant, avant de faire l'envoi sur le marché, elle décide de vous engager pour vérifier si leurs choix technologiques ont été judicieux pour ce type d'application.
Ce fameux appareil possède 64 Mo de mémoire principale et 80 Go de mémoire secondaire. Il peut en tout temps n'exécuter que trois processus (en plus de l'OS) sur son processeur embarqué. Le processus le plus important parmi ces trois est l'Afficheur chargé en mémoire dès le départ (désigné par A). De plus, un processus Transfert (nommé T) permet de transférer les données d'un ordinateur à sa mémoire secondaire à partir d'un port externe de type « USB ». Finalement, le dernier processus Son (appelé S) permet d'envoyer de la musique à partir des données de la mémoire secondaire vers un port externe d'écouteur. Il est important de savoir que l'écoute de la musique est la principale fonction de cet engin. L'affichage et le son doivent toujours être parfaits, tandis que le transfert de données n'est que secondaire.

Cette firme cherche une solution quasi-idéale pour la disposition de la mémoire principale. Elle désire obtenir une disposition qui donne une excellente rapidité d'accès et surtout le moins de fragmentation externe possible.

Elle vous spécifie que

- 16Mo sont réservés pour le système d'exploitation.
- L'afficheur nécessite 500Ko de mémoire de façon stable.
- Le processus de Son a besoin initialement de 500Ko mais sachant qu'une chanson nécessite entre 4Mo et 10Mo, il aura besoin d'un espace contigu supplémentaire (pour éviter des sauts dans la musique).
- le transfert prend initialement 5Mo mais nécessite des tampons dynamiques pour faciliter le transfert.

1- Proposez une organisation pour la mémoire principale de manière à satisfaire tous ces critères (référez-vous à l'intro pour d'autres informations). Justifiez votre proposition.

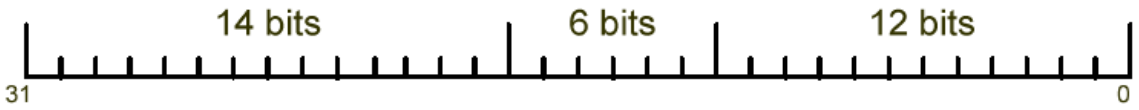
2- Quel type d'adressage proposeriez-vous? (relatif ou absolu). Donnez les avantages de votre choix et expliquer les dispositifs à utiliser pour assurer le bon fonctionnement du système.

Exercice 4 :

Considérez un système de gestion de mémoire qui a les caractéristiques suivantes :

- Un adressage virtuel sur 32 bits
- Une taille de Page de 4Ko
- Une mémoire physique de 1 Mo

a) Supposez que le système utilise la segmentation paginée et que l'adresse virtuelle est de la forme :



Quelles sont les données manquantes à ce problème pour traduire l'adresse virtuelle de 32 bits suivante : 0xAE854C9C en adresse physique ?

Si vous aviez ces informations, identifiez brièvement les étapes à suivre pour effectuer cette translation.

b) Supposez que le système utilise une pagination à deux niveaux, où les entrées des tables de pages sont sur 4 octets. La structure de l'adresse virtuelle est illustrée par la figure suivante :

b.1) Si un processus utilise tout l'espace adressable qui lui est fourni, combien de pages seront-elles nécessaires pour contenir toutes les tables de pages de ce processus.

b.2) Un second processus nécessite 22Mo pour s'exécuter entièrement (son code, ses données, pile…). La partie contenant son code est disposée dans sa mémoire virtuelle aux adresses suivantes [2Mo à 6Mo-1], les données sont quant à elles dans l'intervalle [12Mo-21Mo-1].

Si nous devons charger les tables de pages associées à ces deux parties, combien de pages de niveaux 2 seront chargées en mémoire centrale.

c) Supposez maintenant que le système utilise une pagination pure et que les bits 12 à 31 correspondent à un numéro de page. Si nous utilisons une table inversée, combien d'entrées la table de pages inversée contiendra-t-elle ?

Exercice 5 :

Un système qui implémente la pagination à la demande dispose de 4 cases de mémoire physique qui sont toutes occupées, à un instant donné, avec des pages de mémoire virtuelle. La table 3 donne, pour chaque case de mémoire, le moment du chargement de la page qu'elle contient ($t_{chargement}$), le temps du dernier accès à cette page ($t_{dernier\ accès}$) et l'état des bits référencé (R), modifié (M) et présence (P). Les temps sont donnés en tops d'horloge.

1.

Case	$t_{chargement}$	$T_{dernier\ accès}$	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

Table 3

Indiquez quelle est la page qui sera remplacée en cas d'un défaut de page si l'algorithme de remplacement de page est :

a) LRU
b) FIFO
c) Horloge (seconde chance)

2. Dans le même système, avec pagination à la demande, le temps d'accès à une page chargée en mémoire physique est de 100ns. Le temps d'accès à une page qui n'est pas en mémoire physique est de 10mS s'il y a une case libre en mémoire physique ou si la page qui sera retirée, pour faire place à la page manquante, n'a pas été modifiée. Si la page qui sera retirée, pour faire place à la page manquante, a été modifiée, le temps d'accès est de 20ms. Sachant que le taux de défauts de page est 35%, et que dans 70% des cas de défaut de page, la page à retirer a été modifiée, **calculez** le temps d'accès moyen à la mémoire.

3. On considère un système avec une mémoire virtuelle segmentée paginée où la taille d'une page est de 4Ko et une mémoire physique de 64Ko. L'espace d'adressage d'un processus P est composé de trois segments S1, S2 et S3 de taille, respectivement 16Ko, 8Ko et 4Ko. À un moment donné, pour le processus P, les pages 2 et 3 du segment S1, la page 2 du segment S2 et la page 1 du segment S3 sont chargées en mémoire physique, respectivement dans les cases 2, 0, 9, 12.
Pour une donnée située dans l'espace d'adressage du processus P à l'adresse décimale 8212, **indiquez** :

a) le segment
b) le numéro de page dans le segment
c) le déplacement dans la page
d) le numéro de case

e) le déplacement dans la case

f) l'adresse physique

Exercice 6 :

I) Soit un programme dont le code occupe 1024 octets en mémoire et qui utilise un vecteur avec 1000 éléments de type caractère (un caractère = un octet en mémoire). Ce programme est exécuté dans un système qui utilise la pagination de la mémoire dont la taille de la mémoire réelle est de 1 Mo, la taille d'une page est de 512 octets et les instructions à référence mémoire ont un champ d'adresse de 24 bits.

a) **Donnez :**

1) la taille de l'espace logique d'adressage
2) le nombre de bits du déplacement
3) le nombre de bits du numéro de page virtuelle
4) le nombre de bits d'une adresse réelle
5) le nombre de bits du numéro de page réelle (case)
6) le nombre d'entrées de la table des pages

b) Le chargement de ce programme en mémoire **engendre-t-il** une fragmentation interne? Justifiez votre réponse.

II) **Donnez** le format d'une adresse virtuelle de 32bits avec des pages de 256 octets et une table des pages à trois niveaux.

Si toutes les tables ont le même nombre d'entrées, **donnez** le nombre de tables ainsi que leur nombre d'entrées.

III) On appelle "anomalie de Belady" le fait que le nombre de défauts de pages augmente quand on augmente le nombre de cadres de pages disponibles en mémoire physique. On appelle "algorithme à pile" un algorithme présentant la propriété d'inclusion :

M(m,r) inclus dans M(m+1, r),

où m est le nombre de cadres de mémoire et r un index dans le vecteur de références aux pages (ω). M(m,r) représente l'ensemble des pages présentes en mémoire après r références mémoire, lorsque l'on dispose de m cadres de page.

Une condition suffisante pour qu'un algorithme ne présente pas l'anomalie de Belady est qu'il soit à pile.

a) **Montrez** l'anomalie de Belady pour l'algorithme FIFO avec m=3, puis m=4, pour le vecteur de références aux pages suivant :

ω = { 1, 2, 3, 4, 1, 2, 5, 1, 2, 3, 4, 5 }.

b) **Montrez** sur cet exemple que l'algorithme FIFO n'est pas un algorithme à pile, en indiquant les états pour lesquels la propriété d'inclusion n'est pas vérifiée.

Exercice 7 :

Dans les premiers systèmes UNIX, un processus est créé en mémoire de réserve (partie d'un disque) appelée aussi mémoire de swap puis amené en mémoire centrale pour y être exécuté. La place occupée en mémoire de réserve est libérée dès que le processus a été introduit en mémoire centrale. La taille de la place occupée en mémoire centrale est la même que celle occupée en mémoire de réserve. Plusieurs processus peuvent être présents en mémoire centrale en même temps. Inversement, un processus en mémoire centrale bloqué en attente d'un événement ou se trouvant plus de deux secondes en mémoire centrale alors que des processus prêts attendent en mémoire de réserve plus de trois secondes, doit être recopié en mémoire de réserve pour libérer la mémoire centrale. Un événement attendu par un processus en mémoire peut n'être produit que par un processus en mémoire de réserve.

a) Sachant que la mémoire centrale et la mémoire de réserve ont une taille fixe, peut-on avoir des interblocages ? Si oui, comment traiter ce problème ?

b) Une autre cause de recopie en mémoire de réserve est le fork. On recopie le processus père en mémoire de réserve pour constituer l'image mémoire du fils qui est ensuite introduite en

mémoire centrale (duplication du père). Ceci peut-il causer des interblocages ? Si oui, comment éviter ce problème ?

Exercice 8 :

On considère un système dont l'espace mémoire usager compte 1MO. On choisit la multiprogrammation à partitions variables sans va-et-vient pour ce système.
On suppose la chronologie suivante pour notre système.

Instant t	**Événement**
$t = 0$	A(300, 55) arrive
$t = 10$	B(400, 35) arrive
$t = 30$	C(500, 35) arrive
$t = 40$	D(300, 105) arrive
$t = 50$	E(200, 35) arrive
$t = 60$	F(100, 55) arrive
$t = 70$	G(400, 35) arrive
$t = 90$	H(700, 35) arrive
$t = 110$	I(200, 25) arrive
$t = 120$	J(400, 45) arrive

Bien entendu, un processus qui ne peut pas être chargé en mémoire est placé sur une file d'attente (File de haut niveau). La première ligne du tableau signifie que :
- le processus A arrive à l'instant 0,
- la taille de son espace logique est de 300 K et
- lorsque le processus A est chargé en mémoire centrale, il y séjournera (en mémoire centrale) pendant exactement 55 unités de temps.

1) Donnez les états successifs d'occupation de la mémoire si :

- Le répartiteur de haut niveau fonctionne selon PAPS et le mode d'allocation des trous utilise l'algorithme premier ajustement *(First Fit).*
- Le répartiteur de haut niveau fonctionne selon PAPS et le mode d'allocation des trous utilise l'algorithme pire ajustement *(Worst Fit).*

Exercice 9 :

On considère un système disposant de 16 MB de mémoire physique, avec la partie résidente du système sur 4 MB. On suppose que l'exécution de chaque processus se compose d'**un** temps processeur suivi d'**une** demande d'E/S. On suppose de plus que les processus n'attendent pas pour leur E/S (par exemple, ils utilisent tous un périphérique différent). Le tableau suivant donne un exemple de séquences de tâches pour le système :

Instant t	**Processus**	**Taille**	**Temps CPU**	**Durée E/S**
0	A	3 MB	9 ms	2 ms
4	B	5 MB	6 ms	9 ms
6	C	5 MB	4 ms	4 ms
8	D	4 MB	2 ms	6 ms
10	E	1 MB	4 ms	3 ms
12	F	1 MB	5 ms	1 ms
16	G	1 MB	3 ms	2 ms
18	H	3 MB	3 ms	8 ms

On suppose qu'un processus chargé y séjournera jusqu'à la fin de son exécution.

a) Donnez les états d'occupation de la mémoire aux différentes étapes de traitement de ces processus, sous les hypothèses suivantes :

- Partitions fixes de tailles 6 MB, 4 MB, 2 MB et 4 MB (pour le système) ;
- Le mode d'allocation des trous utilise l'algorithme meilleur ajustement *(Best Fit)* ;
- Le répartiteur de haut niveau fonctionne selon PAPS ;

- Le répartiteur de bas niveau fonctionne selon *SJF (Shortest Job First*).

b) Donnez les états d'occupation de la mémoire aux différentes étapes de traitement de ces processus, sous les hypothèses suivantes :

- Partitions variables ;
- Le mode d'allocation des trous utilise l'algorithme premier ajustement *(First Fit)* ;
- Le répartiteur de haut niveau fonctionne selon PAPS ;
- Le répartiteur de bas niveau utilise l'algorithme du tourniquet avec un quantum de 3 ms.

Exercice 10 : Gestion de la mémoire virtuelle

On considère un ordinateur dont le système de mémoire virtuelle dispose de 4 cases (frames ou cadres) de mémoire physique pour un espace virtuel de 8 pages. On suppose que les quatre cases sont initialement vides et que les pages sont appelées dans l'ordre suivant au cours de l'exécution d'un processus par le processeur:

1 2 3 1 7 4 1 8 2 7 8 4 3 8 1

1) Indiquez tout au long de la séquence d'exécution quelles pages sont présentes dans les cases de la mémoire physique et le nombre de défauts de page selon que l'algorithme de remplacement de pages est :

1. PAPS ;
2. L'algorithme de remplacement "optimal".

Exercice 11 : Gestion du bras du disque

On considère disque dur présentant 128 pistes numérotées en ordre croissant de l'intérieur vers l'extérieur de 0 à 127. On suppose que la tête de lecture/écriture se trouve présentement placée à

la verticale de la piste 15, que sa position précédente était sur la piste 29 et que des requêtes arrivent pour des accès aux pistes suivantes : 100, 30, 27, 55, 16, 122, 44, 63, 56 (dans cet ordre). Quel serait le déplacement total de la tête de lecture/écriture après que toutes les demandes aient été satisfaites si l'algorithme de planification des déplacements de la tête est :

1. Premier arrivé premier servi (PAPS) ?
2. Le plus petit déplacement d'abord (SSTF ou shortest-seek-time-first) ?

Exercice 12 :

1. Expliquez la fonction principale d'un éditeur de liens dans la génération de l'espace adresse d'un processus. Pourquoi doit-on relocaliser les adresses dans le fichier exécutable qui est généré par l'éditeur de liens ? Quels éléments du code doivent être relocalisés ? Pourquoi doit-on aussi relocaliser des adresses lors de l'exécution d'un programme ?
2. Expliquez l'une des raisons pour laquelle un processus peut être déalloué de la mémoire principale avant la fin de son exécution.
3. Comparez la fragmentation interne à la fragmentation externe pour la mémoire, et expliquez laquelle pénalise le plus la performance du système.
4. Identifiez les zones de mémoire où sont sauvegardés les divers éléments d'un programme en état d'exécution.
5. Donnez un avantage pour l'utilisation de partitions fixes de mémoire, et un avantage pour l'utilisation de partitions variables de mémoire.
6. Laquelle des stratégies d'allocations de partitions (first fit, best fit, worst fit) minimise le plus la fragmentation ?
7. Quel avantage procure au système d'exploitation l'utilisation de registres de relocation?
8. Lequel des algorithmes d'allocation de mémoire nous fournirait le meilleur degré de multiprogrammation ? Expliquer.
9. Parmi les algorithmes d'ordonnancement de processus, lequel résulterait en un meilleur degré de multiprogrammation ? Expliquez.
10. Considérez un système dont l'espace mémoire utilisateur compte 1 Mo. On décide d'effectuer une **partition fixe** de cet espace mémoire en trois partitions de tailles respectives 600

Ko, 300 Ko, 100 Ko. On suppose que les processus arrivent dans le système comme est montré sur la table 9.2.

A(200 Ko, 35) veut dire que le processus A nécessite une partition de 200 Ko et que son temps de séjour en mémoire centrale est 35 unités de temps. Bien entendu, un processus qui ne peut pas être chargé en mémoire est placé sur la file des processus en attente de chargement en mémoire. Un processus chargé en mémoire y séjournera jusqu'à la fin de son exécution. Donnez les états successifs d'occupation de la mémoire si :

Instant t	**Événement**
$t = 0$	A(200 Ko, 35) arrive
$t = 10$	B(400 Ko, 65) arrive
$t = 30$	C(400 Ko, 35) arrive
$t = 40$	D(80 Ko, 25) arrive
$t = 50$	E(200 Ko, 55) arrive
$t = 70$	F(300 Ko, 15) arrive

TAB. 9.2 – Besoins de mémoire et temps d'arrivage.

– L'ordonnanceur de haut niveau fonctionne selon le schéma plus court d'abord (SJF) et le mode d'allocation des trous utilise un algorithme du type best fit.

– L'ordonnanceur fonctionne selon PAPS (FCFS) et le mode d'allocation des trous utilise un algorithme du type *First Fit*.

11. Considérez un gestionnaire de mémoire utilisant la stratégie de partitions variables. Supposez que la liste de blocs libres contient des blocs de taille de 125, 50, 250, 256, 160, 500, et 75 octets. Le bloc de 125 octets est en tête de liste et le bloc de 75 octets est en fin de liste.

– Lequel des blocs représente le meilleur espace disponible pour un processus nécessitant une taille de mémoire de 84 octets.

– Lequel des blocs représente le plus grand espace disponible pour un processus nécessitant une taille de mémoire de 221 octets.

– Lequel des blocs représente le premier espace disponible pour un processus nécessitant une taille de mémoire de 178 octets.

12. Pour chacune des stratégies d'allocation de blocs de mémoire libre (meilleur, plus grand, ou premier), expliquez la façon que la liste de blocs libres devrait être organisée afin d'obtenir une performance optimale.

13. Considérez un système n'utilisant pas de registres de relocation. Par conséquent, c'est le chargeur de programme qui s'occupe de la relocation des instructions dans l'image mémoire à partir de l'exécutable. Serait-il possible de décharger et de recharger l'image mémoire du programme durant le cycle d'exécution ? Expliquez comment cela pourrait être réalisable ou pourquoi cela ne pourrait pas être réalisable.

Exercice 13 : Gestion de la mémoire virtuelle

1. Expliquez un avantage majeur que procure l'utilisation de la mémoire virtuelle sur la performance du système.
2. Quelle est la différence principale entre un algorithme de remplacement statique de pages et un algorithme de remplacement dynamique de pages ?
3. Considérez une mémoire virtuelle avec une taille de mémoire physique (principale) de 1 Mo, et supportant des blocs de 128 octets. Aussi, supposez un processus occupant un espace d'adresse logique de 22 Ko.

– Calculez le nombre de cadres dans l'espace d'adresse physique et le nombre de pages dans l'espace d'adresse logique.

– Montrez les formats des adresses physique et logique, soit le nombre de bits pour les blocs, les cadres et les pages.

– Déterminez l'adresse physique dans laquelle se situe l'adresse logique 10237, si l'on suppose que la page contenant l'adresse 10237 se trouve dans le cadre 1839.

4. Considérez la séquence de références de pages 0, 1, 2, 3, 4, 5, 4, 5, 6, 7, 8, 9 faite par un processus. Montrez la séquence d'allocation des pages en mémoire pour chacun des algorithmes énumérés ci-dessous. Aussi, calculez le taux de faute de pages produit par chacun des algorithmes de remplacement de pages de mémoire.

(a) L'algorithme optimal de Belady.

(b) L'algorithme du moins récemment utilisé.

(c) L'algorithme du moins fréquemment utilisé.

(d) L'algorithme du premier-arrivé-premier-servi (PAPS).

(e) L'algorithme de l'espace vital avec une taille de fenêtre V=3. Variez la valeur de V afin de déterminer le point d'allocation optimal. Que survient-il si l'on alloue plus de cadres au processus que son niveau optimal ?

5. On considère un ordinateur dont le système de mémoire virtuelle dispose de 4 cases (frames ou cadres) de mémoire physique pour un espace virtuel de 8 pages. On suppose que les quatre cases sont initialement vides et que les pages sont appelées dans l'ordre suivant au cours de l'exécution d'un processus par le processeur : 1, 2, 3, 1, 7, 4, 1, 8, 2, 7, 8, 4, 3, 8, 1, 1. Indiquez tout au long de la séquence d'exécution quelles pages sont présentes dans les cases de la mémoire physique et le nombre de défauts de page selon que l'algorithme de remplacement de pages est :

(a) PAPS.

(b) L'algorithme de remplacement optimal.

6. Supposez un système de mémoire paginée ayant 2g+h adresses virtuelles et 2h+k adresses en mémoire principale, d'où g, h, et k sont des nombres entiers. Spécifiez la taille des blocs et le nombre de bits pour les adresses virtuelle et physique.

7. Supposez une taille de blocs de 1 Ko, déterminez le numéro de page et le numéro de ligne pour chacune des adresses virtuelles suivantes.

(a) 950

(b) 1851

(c) 25354

(d) 11842