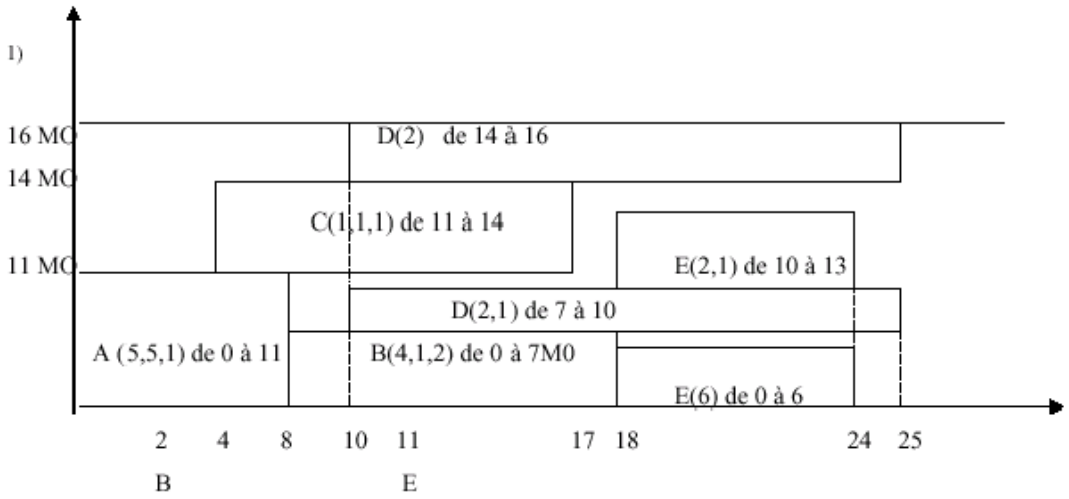
Gestion de la mémoire
Le corrigé

Solution 1

Question 1 :



2) Oui, la dernière page d'un segment peut être non pleine -> fragmentation interne.
Oui, car les segments sont de tailles variables. -> Il y a risque que la mémoire se remplisse de trous libres trop petits (aucun segment ne peut être chargé dedans) -> fragmentation externe.
3) Pour éviter les interblocages. Si le gestionnaire n'applique pas la politique « de tout ou rien » , on pourrait atteindre une situation d'interblocage : il n'y a plus d'espace en mémoire et les processus en mémoire ont été partiellement chargé (il manque pour chacun d'entre eux un ou deux segment(s)). Chaque processus occupe de l'espace mémoire nécessaire à un autre.

Question 2 :
1) Nombre de cases = 16 MO / 4 KO = 4 K = 4096
2) Récupérer le numéro de case où est chargé la page zéro du segment.
Ajouter le numéro de page au numéro de case puis compléter avec le déplacement dans la page.
3) 0000 0000 0101 0000 0011 0001
4) Oui, car il n'y a pas de va-et-vient ni de relocation

Solution 2

1)

- Adresse virtuelle

12	7	10
S	P	D

- Adresse physique

15	10
C	D

2) - La page de l'adresse logique 10728 est ⌊(10728-9216)/1024⌋ soit 1.

- Le déplacement à l'intérieur de la page 10 est 10728%1024 soit 488.

- La page 1 du segment de données se trouve dans le cadre 4097.

- L'adresse du cadre 4097 est 4097*1024 soit 4195328.

- L'adresse réelle de l'adresse logique 10728 est 4195328+488 soit 4195816.

3)

	t0	t1	t2	t3	t4	t5	t6	t7	t8
	…	…	…	…	…	…	…	…	…
X	0	0	0	0	4	4	4	4	8
		1	1	1	1	5	5	5	5
			2	2	2	2	6	6	6
				3	3	3	3	7	7
	…	…	…	…	…	…	…	…	…
Y	0	0	0	0	0	0	2	2	2
				1	1	1	1	1	1
	…	…	…	…	…	…	…	…	…
	…	…	…	…	…	…	…	…	…

(b) Il y a eu 9 fautes de pages de code et 3 fautes de page de données pour un total de 12 fautes sur 17 références soit un taux de 70.5%. Vu qu'il y a 12 pages différentes, le nombre de fautes optimal est alors de 12 ce qui a été atteint par l'algorithme LRU.

Solution 3

1- Les principaux inconvénients de la mémoire à partition fixe est que les partitions sont immuables alors qu'un ordinateur standard peut avoir plusieurs processus de tailles différentes,

dans ce cas les partitions variables sont plus adéquates. Cependant, dans le cas présent, il ne peut y avoir que 3 processus de tailles plus ou moins définie et stable. En plus, nous avons assez de mémoire pour satisfaire la majorité des demandes en mémoire des processus. Une solution efficace serait donc une mémoire a partition fixe sans fil d'attente avec du va-et-vient, car les processus ont toujours leur place de disponible (on ne peut pas faire deux transferts en même temps ou écouter deux musiques en même temps). On donne 16Mo à l'OS, 500 Ko à A, 10.5 Mo à S et 37 Mo à T. T et A pourront donc utiliser une partie de leurs mémoires comme tampon de va-et-vient si c'est nécessaire.

2- Un adressage relatif est de mise, il faut donc utiliser un registre de base, un registre limite et un dispositif de conversion d'adresse qui se charge aussi d'assurer la protection entre les trois processus et aussi avec l'OS.

Solution 4

a) Il manque la table des segments qui va nous indiquer la table de pages associée au segment désiré. Cette table des pages nous permettra d'obtenir le cadre associé à notre adresse. L'adresse physique est obtenue en remplaçant les numéros de segment et de page par le numéro du cadre

b.1) Il y a 2^{10} tables de pages de second niveau et une table de pages de premier niveau. Chaque table de pages a 2^{10} entrées. Elle occupe donc une page 2^{10} x 4octects = 4096 octets.
REP : 2^{10} +1 pages seront nécessaires pour toutes les tables de pages.

b.2) Chaque entrée de la table des pages de niveau 2 est associée à un cadre de 4Ko (2_{12}).
Sachant qu'une table de pages de niveau 2 contient 2_{10} entrées, elle référence 2_{10}* 2_{12} o =4Mo de la mémoire virtuelle. La première table de niveau 2 référence la mémoire virtuelle comprise entre [0, 4Mo-1], la seconde [4Mo, 8Mo-1] et ainsi de suite…..
Donc :

- pour le code 2 tables de pages de niveaux 2 seront nécessaires (c-à-d 2 pages)

- pour les données 3 tables de pages de niveaux 2 seront nécessaires (c-à-d 3 pages).

c) Le nombre de cadres est : $2^{20} / 2^{12} = 2^8$

1 entrée par cadre => 2^8 entrées

Solution 5

1.

a) Dans l'algorithme LRU, on retire la page la moins récemment utilisée. Il s'agit donc de choisir une page selon le critère de la colonne $t_{\text{dernier accès}}$. La page à retirer est celle chargée dans la case 1, qui a été accédée au temps 255.

b) Dans l'algorithme FIFO, on retire la page qui est en mémoire depuis le plus longtemps. Il s'agit donc de suivre le critère de la colonne $t_{\text{chargement}}$. La page à retirer est celle chargée dans la case 2 qui est en mémoire depuis le temps 110.

c) Dans l'algorithme de la seconde chance, on retire la page qui est en mémoire depuis le plus longtemps, donc selon le critère de la colonne $t_{\text{chargement}}$, sauf si son bit R est à 1, auquel cas on le remet à 0 et on poursuit la recherche dans l'ordre. Dans l'exemple, la page chargée dans la case 2 est la plus ancienne, mais son bit R est à 1. La suivante dans l'ordre est la page chargée dans la case 0 dont le bit R est à 0. C'est donc celle qui est choisie.

d) Dans l'algorithme NRU, les pages sont séparées en deux catégories basées sur les valeurs des bits R et M :

Classe 0 : R=0, M=0

Classe1 : R=0, M=1

Classe2 : R=1, M=0

Classe3 : R=1, M=1

On retire une page au hasard dans la classe la plus basse non-vide. Il s'agit donc de retirer la page 0, qui appartient à la classe 0.

2. Le taux d'accès réalisés en 100ns est de 65%. Parmi les 35% accès menant aux défauts de page, 70% ont besoin de 20ms et le reste de 30% ont besoin de 10ms.

$t_{\text{accès moyen}} = 0.65*0.0001 + 0.35*(0.7*20 + 0.3*10) = 5.950065$ mS

3.

a) segment S1
b) page 3
c) déplacement 20
d) case 0
e) déplacement 20
f) L'adresse physique est exprimée sur 16 bits (64Ko = 2^{16}), dont 4 bits pour le numéro de case et 12 bits pour le déplacement dans la case (4Ko = 2^{12}) . Alors, l'adresse physique sera : 0000 0000 0001 0100.

Solution 6

I a)

1) 2^{24} = 16Mo
2) 9 bits
3) 24 – 9 = 15 bits
4) 20 bits
5) 20 – 9 = 11 bits
6) 2^{15} = 32Ko

I b) Oui. Le programme a besoin de 2000 octets de données et de 1024 octets de code, donc un total de 3024 octets en mémoire. Le nombre de pages occupées : [3024 / 512] = 6 pages. Dans la dernière page, il reste 512 – 464 = 48 octets libres, ce qui cause la fragmentation interne.

II) Le format PT1(8bits) PT2(8bits) PT3(8bits) Offset(8bits)

Le nombre de tables : 1 + 256 + (256*256). Le nombre d'entrées : 256

III)

a) L'algorithme FIFO pour m=3

	1	2	3	4	1	2	5	1	2	3	4	5
M_0	1	1	1	4	4	4	5	5	5	5	5	5
M_1		2	2	2	1	1	1	1	1	3	3	3
M_2			3	3	3	2	2	2	2	2	4	4
	P	P	P	P	P	P	P			P	P	

Il y a 9 défauts de page

L'algorithme FIFO pour m=4

	1	2	3	4	1	2	5	1	2	3	4	5
M_0	1	1	1	1	1	1	5	5	5	5	4	4
M_1		2	2	2	2	2	2	1	1	1	1	5
M_2			3	3	3	3	3	3	2	2	2	2
M_3				4	4	4	4	4	4	3	3	3
	P	P	P	P			P	P	P	P	P	P

Il y a 10 défauts de page (anomalie de Belady)

b) L'algorithme exemplifié au point a) n'est pas à pile. Les états où l'inclusion n'est pas vérifiée sont les états d'index 7, 8 et 11 dans le vecteur de références aux pages (ω).

Solution 7

a) Interblocage si

- tous les processus en mémoire centrale sont bloqués,
- aucun processus ne peut être retiré de la mémoire centrale faute de place et

- aucun processus ne peut être chargé en mémoire centrale faute de place.

Solution : lorsqu'un processus est chargé en mémoire, l'espace qu'il occupait en mémoire de réserve n'est pas libéré. On garantit ainsi que tout processus en mémoire centrale peut être transféré, en tout moment, en mémoire de réserve.

b) Interblocage si
- un processus demande la création d'un processus fils, il n'y a pas assez d'espace en mémoire de réserve et
- il n'y pas également assez d'espace en mémoire pour charger des processus prêts de la mémoire de réserve vers la mémoire centrale.

Solution 8

Pour la réponse il suffit de donner le diagramme indiquant l'évolution de l'état de la mémoire. Ce diagramme n'est pas donné ici mais il peut être déduit de la séquence d'événements suivante :

A t =0, A arrive et est chargé en mémoire. Il occupe la zone (1 – 300 K). Il y séjournera pendant 55 unités de temps (t=55). Il libérera son espace à t=55.
A t=10, B arrive et est chargé en mémoire. Il occupe la zone (301 – 700 K). Il y séjournera pendant 35 unités de temps (t=45).
A t = 30, C arrive et ne peut être chargé en mémoire. Il est mis dans la file d'attente de haut niveau FHN (C).
A t= 40, D arrive et est chargé en mémoire. Il occupe la zone (701 – 1000 K). Il y séjournera pendant 105 unités de temps (t=145).
A t= 45, B se termine et libère la zone (301-700 K). C de FHN ne peut être chargé en mémoire.
A t= 50, E arrive et est chargé dans la zone (301-500 K). Il y séjournera pendant 35 (t=85).
A t=55, A se termine et libère la zone (1 – 300 K).

A t=60, F arrive et est chargé en mémoire dans la zone (1 – 100 K). Il y séjournera pendant 55 unités de temps (t=115).

A t=70, G arrive et ne peut être chargé en mémoire. Il est mis en attente dans FHN (C G).

A t= 85, E se termine et libère la zone (301-500 K). C est chargé en mémoire dans la zone (101 – 600 k). Il y séjournera pendant 35 (t=120). FHN contient (G)

A t =90, H arrive et ne peut être chargé en mémoire. Il est inséré FHN (G H).

A t=110, I arrive et ne peut être chargé en mémoire. Il est mis en attente dans la file FHN (G H I).

A t=115, F se termine et libère la zone (1 – 100 K).

A t = 120, C se termine et libère la zone (101 – 600 K). G est chargé dans la zone (1 – 400 K).

Il y séjournera jusqu'à t=155. I est chargé dans la zone (401 – 600 K). Il y séjournera jusqu'à t=145. J arrive et ne peut être chargé en mémoire. Il est mis dans FHN (H J).

A t = 145, I et D se terminent et libèrent les zones (701 – 1000 K) et (401 – 600 K). J est chargé en mémoire dans la zone (401 – 800 K). Il y séjournera jusqu'à t=190. FHN contient (H).

A t = 155, G se termine et libère la zone (1 – 400 K).

A t = 190, J se termine et libère la zone (401 – 600 K). H est chargé dans la zone (1 – 700 K).

Il se termine à t = 225.

(le schéma se déduit de la liste d'événements précédents).

b) se traite de la façon similaire.

Solution 9

a) A t =0, A arrive et est chargé dans la partition de 4 MO, il y séjournera jusqu'à la fin de son exécution (t=9+2).

A t = 4, B arrive et est chargé dans la partition de 6MO. Il y séjournera jusqu'à la fin de son exécution.

A t= 6, C arrive et ne peut être chargé en mémoire. Il est mis en attente dans la file (du répartiteur de haut niveau). La file contient (C).

A t= 8, D arrive et ne peut être chargé en mémoire. Il est mis en attente dans la file (du répartiteur de haut niveau). La file contient (C D)

A t= 9, A libère le processeur. L'exécution de B est entamée. B libérera le processeur à t= 9+6 et la partition à t = 9+6+9.

A t=10, E arrive et est chargé dans la partition de 2 MO. Il y séjournera jusqu'à la fin de son exécution.

A t= 11, l'exécution de A se termine, la partition de 4 MO devient libre. D est chargé dans cette partition et y séjournera jusqu'à la fin de son exécution. La file contient (C).

A t=12, F arrive et ne peut être chargé en mémoire. Il est mis en attente dans la file. La file contient (C F).

A t= 15, B libère le processeur qui est alloué au plus court d'abord c-à-d D. D libérera le processeur à t=15+2 et la partition à t=15+2+6.

A t=16, G arrive et ne peut être chargé en mémoire. Il est mis en attente dans la file. La file contient (C F G).

A t= 17, D libère le processeur qui est alloué au plus court d'abord c-à-d E. E libérera le processeur à t=17+4 et la partition à t=17+4+3.

A t=18, H arrive et ne peut être chargé en mémoire. Il est mis en attente dans la file. La file contient (C F G H). 7

A t=21, D libère le processeur mais tous les processus en mémoire sont en attente de fin d'E/S.

A t = 23, D libère la partition de 4 MO, F est chargé dans cette partition puis son exécution est entamé. Il libérera le processeur à t = 23+5 et la partition à t = 23+5+1.

A t = 24, B et E libèrent les partitions de 6 MO et 2 MO, C et G sont chargés resp. dans les partitions de 6 MO et de 2 MO. La file contient (H).

A t = 28, D libère le processeur qui est aussitôt alloué au processus G. Ce dernier libérera le processeur à t = 28+3 et la partition à t = 28+3+2.

A t = 29, F libère la partition de 4 MO, H est chargé dans cette partition. La file est vide.

A t = 31, G libère le processeur, l'exécution de H est entamé (plus court d'abord). H libérera le processeur à t = 31+3 et la partition de 4 MO à t = 31+3+8.

A t =33, G libère la partition de 2 MO.

A t = 34, H libère le processeur , l'exécution de C est entamé. C libérera le processeur à t=34+4 et la partition de 6 MO à t=34+4+4.

A t = 42, H et C libèrent les partitions de 4 MO et de 6 MO.

Il suffit maintenant de dessiner le schéma (comme celui donné en classe).

b)

A t =0, A arrive et est chargé en mémoire (occupe 5 ième , 6 ième et 7ième MO), il y séjournera jusqu'à la fin de son exécution. Le processeur lui est alloué pendant 3 ms.

A t=3, c'est la fin du quantum de A, mais comme il n'y a aucun autre processus, il continuera à s'exécuter pendant encore 3ms.

A t = 4, B arrive et est chargé en mémoire (8-12 ième MO). Il y séjournera jusqu'à la fin de son exécution. B est mis dans la file des processus prêts qu'on notera FBN. FBN contient (B).

A t= 6, C arrive et ne peut être chargé en mémoire. Il est mis en attente dans la file du répartiteur de haut niveau. Cette file notée FHN contient (C). C'est aussi la fin du quantum de A, A est mis dans la file FBN et B est élu. FBN contient (A). B s'exécute pendant 3 ms jusqu'à t=9.

A t= 8, D arrive et est chargé en mémoire (13 ième à 16 ième MO). Il est inséré dans la file du répartiteur de bas niveau. La file FBN contient (A D)

A t= 9, c'est la fin du quantum de B. Le processeur est alloué à A. B est inséré dans FBN. FBN contient maintenant (D B). L'exécution de A se déroulera jusqu'à t=12.

A t=10, E arrive et n'est pas chargé en mémoire. Il est inséré dans FHN (C E). 8

A t=12, F arrive et ne peut être chargé en mémoire. Il est mis en attente dans FHN (C E F). C'est aussi la fin du quantum de A qui passe en E/S pendant 2 ms. A libère son espace à t =14. Le processeur est alloué à D pendant 2 ms (jusqu'à t=14 ms). La file FBN contient (B).

A t= 14, A libère son espace. E et F sont chargés en mémoire (E occupe le 5 ième MO et F le 6 ième MO). FHN contient (C). C'est aussi la fin du quantum de D qui passe en E/S. Il se termine à t=20. Le processeur est alloué à B pendant 3 ms (jusqu'à t=17 ms). La file FBN contient (E F).

A t =16, G arrive et est chargé en mémoire (occupe le 7 ième MO). FBN contient (E F G).

A t=17, c'est la fin du quantum de B qui passe en E/S pendant 9 ms. Il se termine à t=26. Le processeur est alloué à E pendant 3ms. La file FBN contient (F G).

A t=18, H arrive et ne peut être chargé en mémoire. Il est mis en attente dans la file FHN (C H).

A t=20, c'est la fin du quantum de E. C'est la fin de l'exécution du processus D qui libère son espace. H est chargé (13 ième à 15 ième MO). Le processeur est alloué à F pendant 3 ms. FBN contient (G E H).

A t = 23, c'est la fin du quantum de F. Le processeur est alloué à G pendant 3 ms. FBN contient (E H F).

A t = 26, B libère son espace. C est chargé en mémoire (8 ième à 12 ième MO). C'est aussi la fin du quantum de G qui passe en E/S pendant 2 ms. Il se termine à t=28. Le processeur est alloué à E pendant 1 ms. FBN contient (H F C). FHN est vide.

A t = 27, E passe en E/S pendant 3 ms. Il se termine à t=30. Le processeur est alloué à H pendant 3 ms. FBN contient (F C).

A t = 28, G libère son espace.

A t = 30, E libère son espace. C'est aussi la fin du quantum de H qui passe ensuite en E/S. Il se termine à t=38 ms. Le processeur est alloué à F pendant 2 ms. La FBN contient (C).

A t =32, c'est la fin du quantum de F qui passe en E/S pendant 1 ms. Il se termine à t=33. Le processeur est alloué à C pendant 3 ms. FBN est vide.

A t=33, F se termine.

A t = 35, c'est la fin du quantum de C. Comme il n'y a pas d'autres processus, C continue à s'exécuter pendant encore 1 ms. Il se met ensuite en attente d'E/S pendant 4 ms. Il se termine à t= 39 ms.

A t=39, C se termine.

Il suffit maintenant de dessiner le schéma (comme celui donné en classe).

Solution 10

PAPS

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

Cases 1 1 1 1 1 4 4 4 4 7 7 7 7 7 1

- 2 2 2 2 2 1 1 1 1 1 4 4 4 4

- - 3 3 3 3 3 8 8 8 8 8 3 3 3

- - - - 7 7 7 7 2 2 2 2 2 8 8

Total : 13 défauts de page sur 15 références.

Optimal

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

Cases 1 1 1 1 1 1 1 8 8 8 8 8 8 8 8

- 2 2 2 2 2 2 2 2 2 2 2 3 3 3

- - 3 3 3 4 4 4 4 4 4 4 4 4 4

- - - - 7 7 7 7 7 7 7 7 7 7 1

Total : 8 défauts de page sur 15 références.

Remarque

Dans le cas de l'algorithme optimal, lorsque l'on arrive en fin de séquence, il n'est plus possible de "prédire" l'avenir et, par conséquent, le remplacement de page devient arbitraire.

Solution 11

PAPS

Positions consécutives de la tête de lecture/écriture :

15 - 100 - 30 - 27 - 55 - 16 - 122 - 44 - 63 - 56.

Le déplacement total est donc :

85 + 70 + 3 + 28 + 39 + 106 + 78 + 19 + 7 = 435.

SSTF

Positions consécutives de la tête de lecture/écriture :

15 - 16 - 27 - 30 - 44 - 55 - 56 - 63 - 100 - 122.

Le déplacement total est donc :

1 + 11 + 3 + 14 + 11 + 1 + 7 + 37 + 22 = 107.