Systèmes d'Exploitation 1

Correction Série TD N°7
Gestion de la mémoire : Allocation de mémoire non contiguë (3)

Exercice 1: **Structure des programmes**

Le tableau étant stocké en mémoire ligne par ligne, un élément du tableau étant de taille 1 mot (32 bits) et une page de taille 128 mots, le contenu de la mémoire virtuelle sera semblable à celui-ci :

Page0	A[0][0] A[0][1] … A[0][127]
Page1	A[1][0] A[1][1] … A[1][127]
	…
Page127	A[127][0] A[127][1] … A[127][127]

Si on exécute le programme comme décrit dans l'exercice, et puisque les éléments du tableau sont appelés en commençant d'abord par les colonnes ensuite par les lignes, les pages seront appelées dans l'ordre suivant : page0, page1, …, page127, page0, page1, …, page127…

D'où, pour une mémoire physique de taille 128 cases, on aura 128*128 défauts de page.

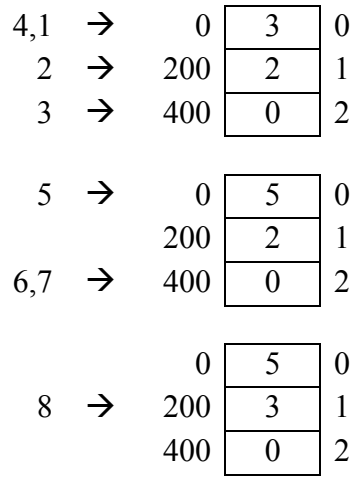
Pour diminuer le nombre de défauts de page, on propose d'inverser les boucles *for* du programme. Ainsi, quand une page est appelée, toutes les données qui sont dessus sont lues en une fois, et on peut la supprimer sans avoir à la charger de nouveau. Ainsi, les pages sont appelées dans l'ordre suivant : page0, page1, …, page127. Et le nombre de défaut de page sera (quel que soit la taille de la mémoire physique) 128 DP.

Exercice 2: **Pagination**

1) Pour remplir le tableau suivant, on doit d'abord déterminer le couple (page, offset) pour chacune des adresses logiques données. Pour cela, on divise l'adresse logique par la taille d'une page. Le résultat est le numéro de page, et le reste est l'offset.

Demande	Conversions d'adresse= (page, offset)			Mémoire Centrale	
	Adresse	Page	Offset	Cadre	Adresse
1	657	3	57	0	57
2	523	2	123	1	323
3	170	0	170	2	570
4	725	3	125	0	125
5	1133	5	133	0	133
6	145	0	145	2	545
7	190	0	190	2	590
8	658	3	58	1	258
9	573	2	173	2	573
10	56	0	56	0	56
11	598	2	198	2	598
12	888	4	88	1	288
13	1134	5	134	2	534
14	170	0	170	0	170
15	472	2	72	0	72

Ensuite, pour déterminer le couple (cadre, adresse), on doit utiliser l'algorithme FIFO pour placer les pages dans la mémoire centrale. L'ordonnancement donne le résultat suivant :



		0	5	0
		200	3	1
9	→	400	2	2

10	→	0	0	0
		200	3	1
11	→	400	2	2

		0	0	0
12	→	200	4	1
		400	2	2

14	→	0	0	0
		200	4	1
13	→	400	5	2

15	→	0	2	0
		200	4	1
		400	5	2

NDP →12 = 8 ; NDP total = 10

2)

4,1	→	0	3	0
2	→	200	2	1
3	→	400	0	2

8	→	0	3	0
5	→	200	5	1
6,7	→	400	0	2

		0	3	0
11,9	→	200	2	1
10	→	400	0	2

12	→	0	4	0
		200	2	1
		400	0	2

		0	4	0
		200	2	1
13	→	400	5	2

14 →	0	4	0
	200	0	1
	400	5	2
15 →	0	2	0
	200	0	1
	400	5	2

NDP →12 = 6 ; NDP total = 9

3)

4,1 →	0	3	0
2 →	200	2	1
3 →	400	0	2

3	2	0		
2	1	1		

8 →	0	3	0
5 →	200	5	1
7,6 →	400	0	2

3	2	0	5	
3	1	3	1	

	0	3	0
11,9 →	200	2	1
10 →	400	0	2

3	2	0	5	
3	3	4	1	

12 →	0	4	0
	200	2	1
	400	0	2

3	2	0	5	4
3	3	4	1	1

13 →	0	5	0
15 →	200	2	1
14 →	400	0	2

3	2	0	5	4
3	4	5	2	1

NDP →12 = 6 ; NDP total = 7
Cette méthode donne un bon nombre de défauts de page, mais on ne peut pas l'appliquer, car on doit maintenir un compteur pour chaque page logique, ce qui cause une grande perte d'espace mémoire.

4) *Question de réflexion* : si les demandes sont mises en attente, alors tous les algorithme donneraient le même résultat : pour un ordre d'exécution donné, les pages existantes en mémoire seront exécutées en premier, ensuite elles seront remplacées une à une par les pages qui attendent dans la file d'attente. Si tous les algorithmes donnent le même résultat, alors il faut choisir le plus simple, soit l'algorithme FIFO.

5) Pour une taille de page = 300, il faut recalculer les couples (page, offset).

Demande	Mémoire Secondaire	
	Adresse	Page
1	657	2
2	523	1
3	170	0
4	725	2
5	1133	3
6	145	0
7	190	0
8	658	2
9	573	1
10	56	0
11	598	1
12	888	2
13	1134	3
14	170	0
15	472	1

L'ordonnancement pour l'algorithme LRU sera le suivant :

1	→	0	2	0
2	→	300	1	1
3	→	0	0	0
		300	1	1
		0	0	0
4	→	300	2	1
5	→	0	3	0
		300	2	1
		0	3	0
7,6	→	300	0	1
8	→	0	2	0
		300	0	1

		0	2	0
9	→	300	1	1
10	→	0	0	0
11	→	300	1	1
12	→	0	2	0
		300	1	1
		0	2	0
13	→	300	3	1
14	→	0	0	0
		300	3	1
		0	0	0
15	→	300	1	1

NDP = 12

Le nombre de défauts de page augmente, car en diminuant le nombre de cases en mémoire, nous avons moins de chance de trouver la page que nous utilisons.

6) Si une page a une taille de 10 unités (très petite par rapport à la taille du programme), alors le programme sera très fragmenté, ce sui résultera en un grand nombre de pages, et donc de défauts de pages.

➔ **Conclusion** : La taille d'une page dans une mémoire paginée ne doit être ni trop grande ni trop petite : il faut choisir une taille qui soit optimale par rapport à la taille des programmes usuels.