Remplacement de pages

Exercice 1. Questions de cours Les questions de cours sont à destinées à vous permettre de vérifier votre compréhension du cours. Elles sont à travailler à l'avance et ne seront pas traitées en TD ou TP.

1. Donnez les deux grandes classes d'algorithmes de remplacement de pages.
2. En pratique, quel est le meilleur algorithme ? Pourquoi ?
3. Quel est l'inconvénient de l'algorithme FIFO avec seconde chance (FIFO-2) ?
4. Quel est la particularité de l'algorithme NRU ?
5. Quel est l'inconvénient de l'algorithme NRU ?

Exercice 2. Gestion de la mémoire On se place dans un système de mémoire de 1Go de mémoire géré de manière paginée et segmentée avec des cadres de page de 4ko. Chaque processus peut utiliser jusqu'à 1024 segments. Chaque segment peut occuper 4Mo de mémoire. Le système d'exploitation autorise jusqu'à 1024 processus.

1. Quelle est la taille (en bits) de l'adresse logique ?
   Correction: *Un processus peut occuper* 2^{10} *segments* $\times$ 2^{22} *octets de mémoire, donc l'adresse logique est sur 32 bits.*
2. Quelle est la taille (en bits) de l'adresse physique ?
   Correction: $1\text{Go} = 2^{30}\text{o}$ *donc l'adresse physique est sur 30 bits.*
3. Combien y a-t-il de cadres de page dans la RAM ?
   Correction: $4\text{Ko} = 2^{12}$. *Il y a donc* $2^{(30-12)} = 2^{18}$ *cadres de pages.*
4. On suppose que l'OS utilise un mécanisme de mémoire virtuelle équitable : chaque processus dispose de la même proportion de mémoire physique. Complétez le tableau suivant en indiquant la probabilité de faire un défaut de page pour un processus, en fonction du nombre total de processus prêts, en exécution ou en attente dans le système et de la taille dudit processus. On suppose que toutes les pages du processus sont équiprobables.

Taille du processus $\rightarrow$	512ko	16Mo	1Go
32 proc. présents			
512 proc. présents			
1024 proc. présents			

   Correction: *Principe : si j'ai N processus, chaque processus dispose de* $\text{M} = 2^{30}/\text{N}$ *de mémoire physique. Si le processus est plus petit, il n'y a pas de défaut de page. S'il est plus gros (et de taille T), j'ai une probabilité de* M/T *de ne pas faire un défaut (donc une probabilité* $1 - \text{M}/\text{T}$ *de défaut).*

   Par exemple, avec 512 processus présent, chaque processus dispose de $2^{30}/2^9 = 2^{21}\text{o}$ *de mémoire ou* $2^{21}/2^{12} = 2^9$ *cadres de page.*

   Un processus de $16\text{Mo} = 2^{24}\text{o}$ *occupe* 2^{12} *pages ce qui est plus que le nombre de cadres dont il dispose. Il a donc une probabilitée de* $1 - 2^9/2^{12} = 0,875$ *de subbir un defaut de page.*

Taille	*512ko*	*16Mo*	*1Go*
32 proc. présents	*0*	*0,000*	*0,969*
512 proc. présents	*0*	*0,875*	*0,998*
1024 proc. présents	*0*	*0,937*	*0,999*

5. Pour réduire le nombre de défauts de page, on décide de passer à une politique d'allocation proportionnelle. On se place dans le cas 512 processus avec un nouveau processus de 16Mo. On suppose que les 511 processus précédents ont une moyenne de 2Mo. Indiquez la probabilité d'obtenir un défaut de page.

   Correction: *Attention : le fait de rajouter un processus réduit la proportion de pages des autres processus. Tous les processus reçoivent la même proportion de l'espace total et donc*

auront la même probabilité de défaut. Il n'est pas nécessaire de calculer combien reçoit le nouveau processus : il suffit de calculer la proportion d'espace manquant. Nous avons en tout $(512-1)*2\text{Mo}+16\text{Mo}$ *de mémoire virtuelle utilisée, soit* $1\text{Go}+14\text{Mo}$*. La probabilité d'erreur est donc de* $14/(1024+14)=0,013$*.*

6. On suppose toujours que nous avons 1 processus de 16Mo et 511 processus de 2Mo. Si l'accès disque est 1000 fois plus lent que l'accès mémoire, que nous fixons à 1 unité de temps, donnez le temps moyen de traitement de 1000 accès mémoire par processus (512000 accès en tout) dans les deux cas suivants :

- Politique équitable ;
- Politique proportionnelle.

Qu'en concluez-vous ?

Correction: *Attention : il faut compter pour tous les processus, pas seulement notre nouveau processus.*

- *Politique équitable : les 511 petits processus ne feront pas de défaut de page. Notre processus fera environ 875 défauts. Total :* $511\times 1000+125+875\times 1000=1386125$
- *Politique proportionnelle : les 512 processus feront 13 défauts de page environ. Total :* $512\times(13*1000+987)=7161344$
  Conclusion : la méthode équitable est bien meilleure sur cet exemple !

Exercice 3. Remplacement de page Durant son exécution, un programme accède successivement à la liste suivante de pages virtuelles de son espace d'adressage :

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

On suppose que le système d'exploitation a alloué 3 cadres de pages au processus et que le cache est initialement vide.

1. Déroulez l'algorithme FIFO simple sur cet exemple et indiquez le nombre de défauts de pages.

Correction: *9 défauts de page.*

	0	1	4	2	0	1	3	0	1	4	2	3
0	0			2			3					
1		1			0					4		
2			4			1					2	
	*	*	*	*	*	*	*			*	*	

2. Déroulez l'algorithme FIFO avec bit de seconde chance sur cet exemple et indiquez le nombre de défauts de pages.

Correction: *9 défauts de page.*

	0	1	4	2	0	1	3	0	1	4	2	3
0	0			-2			-3			-		
1		1		-	0		-	+		-4		
2			4	-		1	-		+	-	2	
	*	*	*	*	*	*	*			*	*	

Les - indiquent que le bit de seconde chance est mis à 0 tandis que les + indiquent qu'il est remis à 1.

3. Déroulez l'algorithme LRU sur cet exemple et indiquez le nombre de défauts de pages.

Correction: *10 défauts de page.*

	0	1	4	2	0	1	3	0	1	4	2	3
0	0			2			3			4		
1		1			0			=			2	
2			4			1			=			3
	*	*	*	*	*	*	*			*	*	*

Les = indiquent que la page à été réutilisée et n'est donc plus la plus récement utilisée.

4. Quel serait le remplacement optimal (si on pouvait deviner à l'avance les appels que va faire le programme) et donc le taux de défaut de page minimum ?

Correction: *7 défauts de page.*

	0	1	4	2	0	1	3	0	1	4	2	3
0	0									4		
1		1									2	
2			4	2			3					
	*	*	*	*			*			*	*	

Exercice 4. Remplacement de page On s'intéresse aux algorithmes de remplacement de pages dans un cache capable de contenir 4 pages. Le gestionnaire de mémoire accède successivement aux pages suivantes :

7, 1, 8, 2, 3, 1, 6, 1, 2, 5

Initialement, le cache est vide.

1. Déroulez l'algorithme FIFO simple sur cet exemple et indiquez le nombre de défauts de pages.
Correction: *8 défaut de page*

	7	1	8	2	3	1	6	1	2	5
0	7				3					
1		1					6			
2			8					1		
3				2						5
	*	*	*	*	*		*	*		*

2. Déroulez l'algorithme FIFO avec bit de seconde chance sur cet exemple et indiquez le nombre de défauts de pages.
Correction: *7 defauts de page.*

	7	1	8	2	3	1	6	1	2	5
0	7				-3					-
1		1			-	+	-	+		-
2			8		-		6			-
3				2	-				+	-5
	*	*	*	*	*		*			*

On note sur le dernier accès que, même si le cadre 2 vient d'être utilisé et donc vient de récupérer sa seconde chance, il est quand même sélectionné pour être remplacé car tous les autres cadre ont aussi leur seconde chance.

3. Déroulez l'algorithme LFU sur cet exemple et indiquez le nombre de défauts de pages.
Correction: *7 defauts de page.*

	7	1	8	2	3	1	6	1	2	5
0	7				3					5
1		1								
2			8				6			
3				2						
	*	*	*	*	*		*			*

Les choix de cet algorithme ne diffèrent du précédent que lors du dernier accès. On peut voir que le compteur d'accès ici protège mieux la page 2 qui a de plus forte chance d'être réutilisée car elle à déjà été accédée 2 fois notament au temps précédent.

4. Quel est le nombre de défauts de page minimal sur cet exemple ?
Correction: *7 defauts de page, on ne peut pas faire mieux ici que ce que l'on a obtenu aux deux questions précédentes. En effet, chaque defaut se produit sur une page qui n'a encore jamais été accédée.*

5. Quel serait le nombre de défauts de page minimal sur cet exemple avec un cache de taille 3 ?
Correction: *7 defauts de page là aussi. Seules deux pages sont réutilisées, il suffit donc d'éviter de les retirer et d'utiliser le dernier cadre pour les pages qui ne sont accédées qu'une seule fois.*

	7	1	8	2	3	1	6	1	2	5
0	7			2						
1		1								
2			8		3		6			5

Exercice 5. Remplacements de page On s'intéresse aux algorithmes de remplacement de pages dans un cache capable de contenir 5 pages. Le gestionnaire de mémoire accède successivement aux pages suivantes :

`1, 7, 8, 2, 3, 1, 6, 1, 2, 7, 3, 5, 6`

Initialement, le cache est vide.

1. Déroulez l'algorithme FIFO simple sur cet exemple et indiquez le nombre de défauts de pages.
Correction: *9 defauts de page*

	1	7	8	2	3	1	6	1	2	7	3	5	6
0	1						6						
1		7						1					
2			8							7			
4				2								5	
5					3								
	*	*	*	*	*		*	*		*		*	

2. Déroulez l'algorithme FIFO avec bit de seconde chance sur cet exemple et indiquez le nombre de défauts de pages.
Correction: *9 defauts de page*

	1	7	8	2	3	1	6	1	2	7	3	5	6
0	1					+	-6					-	+
1		7					-	1				-	
2			8				-			7		-	
4				2			-		+			-5	
5					3		-				+	-	
	*	*	*	*	*		*	*		*		*	

3. Déroulez l'algorithme LRU sur cet exemple et indiquez le nombre de défauts de pages.
Correction: *9 defauts de page. On peut voir ici que même le meilleur algorithme ne peut améliorer le nombre de defaut sur cet exemple. Lors de l'acces à la page 6, on ne peut savoir que la page 7 sera réutilisée et le choix fait est raisonable. De même lors de l'accès à la page 5.*

	1	7	8	2	3	1	6	1	2	7	3	5	6
0	1					=		=					6
1		7					6					5	
2			8							7			
4				2					=				
5					3						=		
	*	*	*	*	*		*			*		*	*

4. Quel est le nombre de défauts de page minimal sur cet exemple ? Justifiez.
Correction: *7 defauts de page, il faut remplacer la page qui sera utilisée le plus tard dans le futur. C'est donc la page 8 qui saute en premier (elle n'est pas réutilisée). Au final seuls les accès à de nouvelles page font des defauts.*

	1	7	8	2	3	1	6	1	2	7	3	5	6
0	1											5	
1		7											
2			8				6						
4				2									
5					3								
	*	*	*	*	*		*					*	