Université Cheikh Anta Diop

École Supérieure Polytechnique
Département Génie Informatique
Année Universitaire 2023-2024

Travaux Dirigés de Système d'Exploitation
TD3: Gestion de la mémoire
Dr Mandicou BA

Exercice1 : Fragmentation et temps de compactage
Pour répondre à cette question, il est nécessaire de calculer le temps total nécessaire pour compacter 120 Mo en utilisant les temps de lecture et d'écriture donnés.

Premièrement, convertissons la taille de la mémoire de 128 Mo en Ko, car nous devons manipuler :

1 Mo = 1024 Ko
Donc, 120 Mo = 128 * 1024

Maintenant, nous devons considérer le fait que chaque octet nécessitera une opération de lecture et une opération d'écriture pour être déplacé pendant le compactage.

Temps total pour lire et écrire = temps de lecture + temps d'écriture = 10 ns + 10 ns = 20 ns

Ainsi, pour compacter 128 Mo de mémoire, nous devrons effectuer des opérations de lecture et d'écriture pour chaque Ko. Le temps total nécessaire serait donc :

. Un kilooctet (ko) équivaut à 8,192 bits
donc 32bits=32/8192=0,0039 ko
0,0039ko-----→20^10-9s
131072------→ ?

Calculons cela :

Temps total = (131072 *20 * 10^-9)/0,0039=0,67s

Donc, il faudra environ 0,67secondes pour compacter 128 Mo de mémoire en utilisant ce système de va et vient avec compactage.

Exercice 2 :
la gestion de l’espace libre par on note que 1 correspond a occupée libre correspond a 0
1-tableau de bits

1 1 1 0 0 0 0 0 0 1 1 1 1 0 0 0 0 0 1 1 1 1 1 1 1 1 0 0 0 0 0 0 0 1 1 1 0 0 0 1 1 1 1 1
2-par liste chaînées

o 1 3 ->l 4 6 ->o 10 4->l 145 ->o 19 8->l 277 ->o 34 3->l 373 ->o 40 5->null

Exercice3:stratégies de placement de données en mémoire
liste des trous avant allocation 10k 4k 20k 18k 7k 9k 12k 15k
1-first fit
on parcours tout le tableau et on alloue le premier segment libre suffisamment grande
liste
pour placer 12k 20k-12k=8k

10	4	8k	18	7	9	12	15

On va placer 10k au début il y’a une place libre de 10k pouvant le contenir
nouvelle liste

4	8k	18	7	9	12	15

On va placer 9k on fait 18k-9k=9k

4	8k	9k	7	12	15

-best fit
on choisit le meilleur ajustement
pour 12k on regarde la plus petite valeur adéquate c’est 12k donc on a
10k 4k 20k 18k 7k 9k 15k
pour placer 18k on peut prendre 18k donc on a
10k 4k 20k 7k 9k 15k
pour placer 9k on prend 9k qui est un trous libre et il y’aura pas de reste
10k 4k 20k 7k 15k

-worst fit on choisit le trous ou us auront le plus grand restant oubien on choisit la plus grande valeur du tableau on lui alloue et on prend le reste
10k 4k 20k 18k 7k 9k 12k 15k
pour placer 12k on choisit 20k 20k-12k=8k donc
10k 4k 8k 18k 7k 9k 12k 15k
pour placer 18k on choisit 18k comme c’est la plus garnde valeur
10k 4k 8k 7k 9k 12k 15k
pour placer 9k on choisit 15k comme c’est la plus grande valeur
10k 4k 8k 7k 9k 12k 6k

Exercice 4 : ALLOCATION MÉMOIRE CONTIGUË
la mémoire est de 1700 ko repartie en 5 positons en Ko

100ko	500ko	200ko	300ko	600ko

le système doit allouer quatre processus p1,p2,p3,p4 de taille respective 212ko,417ko,112ko et 426ko
1-pour chaque algorithme donnons l'allocation obtenue et le taux de fragmentation(calcule comme étant la zone mémoire inutilisée sur la zone mémoire totale
a)first-fit (prochain bloc libre)
P1 212ko

100ko	288ko	200ko	300ko	600ko

P2 417ko

100ko	288ko	200ko	300ko	183ko

P3 112ko

100ko	76ko	200ko	300ko	183ko

P4 426ko
Après avoir parcouru toutes les partitions disponibles, nous constatons que le processus de 426ko ne peut pas être placé dans la liste de partitions donnée en utilisant la méthode du "First Fit" car aucune partition n'est assez grande pour l'accueillir.

Taux de fragmentation=(Zone me´moire totaleZone me´moire inutilise´e)×100%
TA=859*100/1700=50,52 %
b)best fit plus petit bloc libre
p1 212ko

100ko	500ko	200ko	88ko	600ko

P2 417ko

100ko	83ko	200ko	88ko	600ko

P3 112ko

100ko	83ko	88ko	88ko	600ko

P4 426ko

100ko	83ko	88ko	88ko	174ko

Le TA=533*100/1700=31,35 %

-par worst-fit plus grand bloc libre
p1 212ko

100ko	500ko	200ko	300ko	388ko

P2 417 ko

100ko	83ko	200ko	300ko	388ko

P3 112ko

100ko	83ko	200ko	300ko	276ko

P4 426ko
Après avoir parcouru toutes les partitions disponibles, nous constatons que le processus de 426ko ne peut pas être placé dans la liste de partitions donnée en utilisant la méthode du "First Fit" car aucune partition n'est assez grande pour l'accueillir.
TA=959*100/1700=56,41 %
le plus efficace est best fit car son taux de fragmentation est plus petite et donc elle est plus optimale mais il présente des inconvénients en termes de complexité, de temps d'exécution et de potentiel de fragmentation, ce qui peut limiter son efficacité dans certains contextes.

Exercice5:algorithmes de remplacements de pages

pages	Date de chargement	Date de référencement	RB	MB
0	126	279	0	0
1	230	270	1	0
2	120	272	1	1
3	160	280	1	1

Indiquons la page qu’il faudra remplacer prochainement dans le contexte de chacune des stratégies
1.FIFO
L'algorithme de remplacement de pages FIFO (First-In, First-Out) remplace la page qui a été chargée en premier dans la mémoire. Pour indiquer la page qui sera remplacée selon l'algorithme on va procéder ainsi
selon fifo la page 126 sera chargée en premier
2.LRU (least recenty used) page 1 car La page la moins récemment utilisée est la page 1 car elle a la date de référencement la plus ancienne (270). Donc, selon l'algorithme LRU, la page à remplacer serait la page 1.
3.NRU(not recently used) page 0

Exercice 6
donnons le numéro de la page virtuelle et le déplacement correspondante
les adresses virtuelles AL 20000,32768 et 64096
(a)des pages de tailles 4ko
les adresse virtuelle AL 20000
NP=AL div TP
NP=20000 div 4096
NP=4
le decalage
D=AL mod TP
D=20000 mod 4096
D=1088
les adresses virtuelles AL 32768
NP=32768 div 4096
NP=8
D=AL mod TP
D=32768 mod 4096=0

les adresses virtuelles AL 64096
NP=64096 div 4096=15
D=64096 mod 4096=2656

(b)taille pages 8ko
8ko=8*1024=8192 o
(a)des pages de tailles 4ko
les adresse virtuelle AL 20000
NP=AL div TP
NP=20000 div 8192
NP=2
le decalage
D=AL mod TP
D=20000 mod 8192
D=3616

les adresses virtuelles AL 32768
NP=32768 div 8192
NP=4
D=AL mod TP
D=32768 mod 8192=0

les adresses virtuelles AL 64096
NP=64096 div 8192=7
D=64096 mod 8192=6752

(a)calculons les adresses physiques correspondant aux adresses virtuelles 24,4300,12300
pour 24
Tailles cases=4ko=4096
NP=AL div TP
NP=24 div 4096=0

D=AL mod TP
D=24 mod 4096=24
TC[7]→3 donc appartient a l'interval [12k 16k[
AP1=NP*TP+D
AP1=3*4096 + 24=12312

-pour AL=4300

Tailles cases=4ko=4096 o
NP=AL div TP
NP=4300 div 4096=1
D=AL mod TP
D=4300 mod 4096=204

AP2=NP*TP+D
AP2=1*4096 + 204=4300

-13200
NP=AL div TP=
NP=13200 div 4096=3
D=13200 mod 4096=912
TC[3] →0
AP3=0*4096 + 912=912

Exercice7:Étude de la pagination d’un processus

Pages	0	1	2	3	4	5	6	7
Cadres	011	001	000	010	100	111	101	110
Bit de présence	1	0	1	0	0	0	1	0

1-l’espace d’adressage est composé de 8 pages
2-la memoire centrale est composé de 8 cases
3-la taille de l’espace d’adressage du processus p1
256*8=2048 mots
4-Le système dispose de 8 cadres * 256 mots mémoires par cadre, soit 2048 mots mémoires au total.
5-L'adresse virtuelle du processus est codée sur le nombre de bits nécessaires pour représenter chaque page, donc dans ce cas, c'est 3 bits (car il y a 8 pages). La taille de l'offset est déterminée par le nombre de mots mémoires dans une page, soit 8 bits pour représenter les 256 mots mémoires.
6-Les pages sont codées sur 3 bits (car il y a 8 pages), les cases sont codées sur 3 bits également (car il y a 8 cadres), et l'offset est codé sur 8 bits (car il y a 256 mots mémoires dans une page).
Illustration de la représentation d'une adresse virtuelle

page	case	offset
3bits	3bits	8bits

- Table de pages du processus P1 :

Page	Présence
0	1
1	0
2	1
3	0
4	0
5	0
6	1
7	0

8.Pour calculer les adresses réelles (physiques), nous devons utiliser la table de pages pour traduire l'adresse virtuelle en adresse physique. Voici les étapes pour chaque adresse virtuelle donnée :

- Pour 240 : Page: 240 / 256 = 0 (Page 0) Déplacement: 240 % 256 = 240 Adresse physique: Case 011, Offset 11110000 (en binaire) Conversion décimale: 01111110000 = 992
- Pour 546 : Page: 546 / 256 = 2 (Page 2) Déplacement: 546 % 256 = 34 Adresse physique: Case 000, Offset 100010 (en binaire) Conversion décimale: 000100010 = 34
- Pour 1578 : Page: 1578 / 256 = 6 (Page 6) Déplacement: 1578 % 256 = 42 Adresse physique: Page 101, Offset 101010 (en binaire) Conversion décimale: 101101010 = 730
- Pour 2072 : Page: 2072 / 256 = 8 (Erreur, car il n'y a que 8 pages) Erreur d'adressage.

9. Si P1 génère l'adresse virtuelle 770 : Page: 770 / 256 = 3 (Page 3) Déplacement: 770 % 256 = 2 Le bit de présence pour la page 3 est 0, ce qui signifie qu'elle n'est pas chargée en mémoire. Cela déclencherait une erreur de page (page fault), car la page demandée n'est pas présente en mémoire.
10. L'adresse virtuelle est 0000 0000 0000 0111. Numéro de page: 000 (en binaire) Déplacement dans la page: 0000 0000 0111 (en binaire) Selon le numéro de page, l'adresse correspond à la case 011 (en binaire). L'adresse physique est donc 011 0000 0000 0111 (en binaire).

Exercice8 :

1. La taille des cases de la mémoire centrale est de 512 octets (4K / 8 cases). On peut en déduire que chaque case peut stocker une quantité fixe de données correspondant à cette taille.
2. La taille de l'espace d'adressage de chaque processus est de 4 pages. Chaque page occupe une case dans la mémoire centrale.
3. Pour coder l'espace d'adressage de chaque processus, nous avons besoin de 2 bits pour chaque page (pour un total de 4 pages). Ainsi, l'espace d'adressage de chaque processus est codé sur 8 bits.
4. Pour coder l'espace d'adressage physique, nous avons besoin de 3 bits pour représenter les 8 cases de la mémoire centrale (2^3 = 8).
5. Pour coder les numéros de cases et le déplacement, nous utilisons 3 bits pour les numéros de cases (car nous avons 8 cases) et les bits restants (9 bits) pour le déplacement.
6. Illustration d'une adresse physique :

Adresse physique = Numéro de case (3 bits) + Déplacement (9 bits)
Exemple : 010 110110101 (en binaire)
Illustration d'une adresse logique :
Adresse logique = Numéro de page (2 bits) + Déplacement (9 bits)
Exemple : 11 110110101 (en binaire)

7. Table de page :

Page	Présence
0	0
1	1
2	0
3	1

8.Pour l'adresse virtuelle 32F du processus de rang impaire de poids faible : Page: 32F / 512 = 0 (Page 0) Déplacement: 32F % 512 = 271 Adresse physique: Case 110 (en binaire) + Déplacement (271) = 110 001010111 (en binaire) = 6AB (en hexadécimal)

8. Pour l'adresse virtuelle 2C A du processus de rang paire : Page: 2C A / 512 = 0 (Page 0) Déplacement: 2C A % 512 = 42 Adresse physique: Case 011 (en binaire) + Déplacement (42) = 011 001010 (en binaire)
9. Si le processus de rang impaire de poids fort tente d'exécuter à l'adresse virtuelle 5F5, il rencontrera une erreur car la page correspondante n'est pas présente en mémoire centrale. Cela entraînera probablement un "page fault", obligeant le système d'exploitation à charger la page depuis le stockage de masse avant de permettre à l'accès de se poursuivre.
10. Si le processus de rang impaire de poids fort tente d'exécuter à l'adresse virtuelle 39E, il y aurait un problème car cette adresse se trouve en dehors de l'espace d'adressage valide pour ce processus. Une solution serait de générer une erreur d'adressage ou d'envoyer un signal d'erreur au processus pour l'informer que l'adresse n'est pas valide.

Exercice9 :

Pour résoudre cet exercice, nous allons utiliser la table des segments fournie pour calculer les adresses réelles correspondant aux adresses virtuelles données.

1. Pour chaque adresse virtuelle (segment, offset) donnée, nous allons d'abord trouver l'adresse de base correspondante à partir de la table des segments, puis nous ajouterons l'offset pour obtenir l'adresse réelle.

- Pour l'adresse virtuelle (0;128) :
  Adresse de base: 540
  Limite: 234
  Adresse réelle = Adresse de base + offset = 540 + 128 = 668

- Pour l'adresse virtuelle (1;100) :
  Adresse de base: 1254
  Limite: 128
  Adresse réelle = Adresse de base + offset = 1254 + 100 = 1354

- Pour l'adresse virtuelle (2;465) :
  Adresse de base: 54
  Limite: 328
  Erreur d'adressage, car l'offset dépasse la limite.

- Pour l'adresse virtuelle (3;888) :
  Adresse de base: 2054
  Limite: 1024
  Erreur d'adressage, car l'offset dépasse la limite.

- Pour l'adresse virtuelle (4;100) :
  Adresse de base: 976
  Limite: Non spécifié (pas d'information sur la limite pour ce segment)
  Erreur d'adressage, car le segment n'a pas de limite spécifiée.

- Pour l'adresse virtuelle (4;344) :
  Adresse de base: 976
  Limite: Non spécifié (pas d'information sur la limite pour ce segment)
  Erreur d'adressage, car le segment n'a pas de limite spécifiée.

2. Pour l'adresse virtuelle (4;200), nous avons besoin de vérifier si le segment 4 existe et si l'offset (200) est dans la limite du segment. Cependant, les informations sur la limite du segment 4 ne sont pas fournies dans la table des segments. Par conséquent, nous ne pouvons pas déterminer la validité de cette adresse virtuelle sans cette information.