TD n° 5 : Pagination

Exercice n°1 : Conversion dans un système paginé

Chaque processus dispose d'un espace logique (virtuel) constitué d'un nombre entier de pages, ainsi que d'une table de pages. Les indices de cette table ont les numéros 0, 1, 2… attribués aux pages.

Lors de son exécution, les pages du processus sont chargées dans des cases (blocs) disponibles, non nécessairement adjacents, de la mémoire centrale. Simultanément, la table de pages est mise à jour.

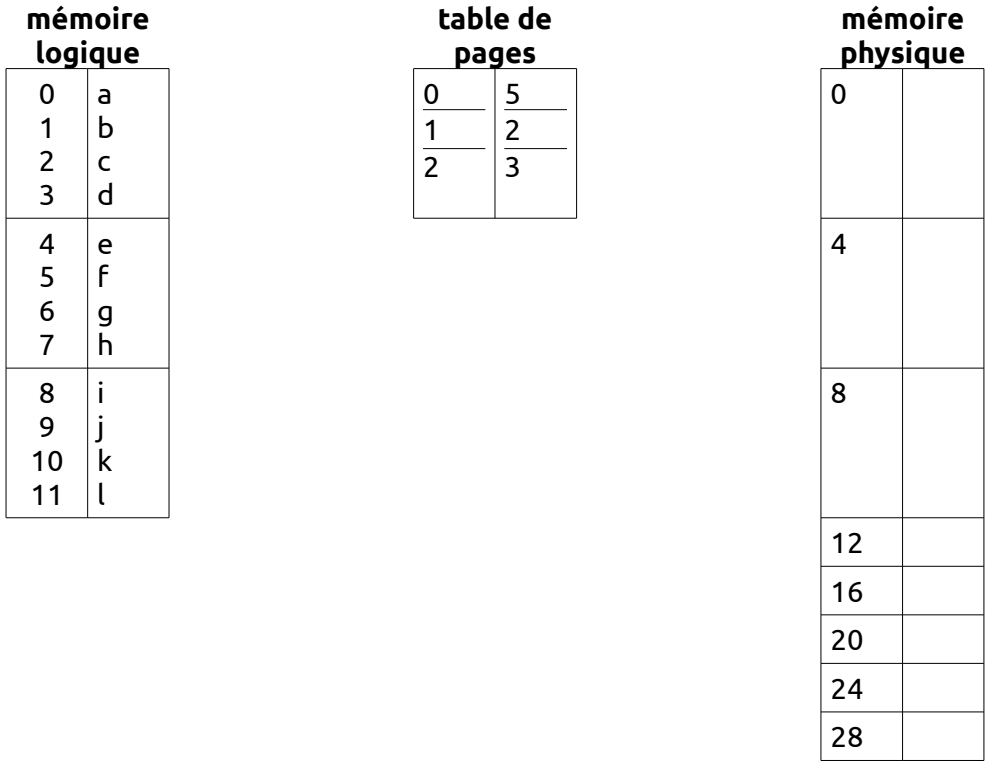

Fig. 1 : Recherche d'une adresse

On observe une fragmentation interne pour la dernière case allouée, lorsque la taille du processus n'est pas un multiple de taille d'une page.

Supposons qu'au cours de cette exécution, le processus fasse référence à une adresse logique *L*. cette adresse logique est transformée en un couple (*x, d*), où *x* est le numéro de page et *d* le déplacement dans la page. En désignant par *c* la taille d'une page, *x* et *d* sont respectivement le quotient et le reste dans la division entière de *L* par *c :*

$$x = L \text{ div } C \text{ et } d = L \text{ mod } c.$$

L'adresse logique est donc :

$$\boldsymbol{L = x.c + d.}$$

Les cases sont également numérotées 0, 1, 2… et l'entrée correspondante à *x* dans la table de pages contient le numéro de case *y* en MC. Le fait que les pages et les cases soient tous de même taille, pour calculer l'adresse physique, il suffit donc d'ajouter le déplacement à l'adresse de base de la case *y*, qui est *y.c :*

$$\boldsymbol{P = y.c + d.}$$

Pour éviter une division fastidieuse, la taille des pages est habituellement définie par le matériel comme une puissance de 2 (512 mots, 2048 ou 4096 octets).

Ainsi, lorsqu'une page contient 2^n unités adressables (octets ou mots), les *n* bits de poids faible d'une adresse logique représentent le déplacement dans la page, tandis que les bits de poids fort représentent le numéro de page.

Questions :

Nous considérons la figure 1 :

1. Donner la taille c, en unité adressable, de chaque page.

2. Donner le nombre de cases de la mémoire physique. Déduire le nombre d'unités adressables.
3. En se basant sur les informations de la table de pages, préciser le contenu de la mémoire physique.
4. Calculer l'adresse physique correspondant à la page 2 qui se trouve dans la case y = 3.
5. Proposer un schéma général relatif à ce procédé de conversion qui intervient à chaque référence du processus.

Exercice n°2 : Algorithmes de pagination

Intuitivement un algorithme de pagination efficace doit provoquer le minimum de défauts de page. Pour préciser cela et afin de pouvoir comparer les performances des algorithmes de pagination, nous définissons une notion de coût.

Soit c une fonction de coût vérifiant :
si c(q) est le coût associé à un transfert de q pages, alors :

c(0) = 0, c(1) = 1 et c(q) >= 1 pour q >= 2.

Le coût de l'algorithme *A*, opérant sur la suite de références $w = x_1 x_2 ... x_p$, dans une mémoire de taille *m* est :

$$C(A, m, w) = \sum_{k=1}^{p} c[card(X_k)]$$

où X_k est l'ensemble des pages chargées par la transition x_k.

Questions :

A. Première entrée première sortie : FIFO

Considérons la suite de références w = 0 1 2 3 0 1 4 0 1 2 3 4.

1. Calculer le coût C(FIFO, m, w) pour les valeurs de m = 3 et m =4 et après avoir établit les figures qui montre la suite des états et les défauts de page.

Hypothèse : Dans cet exemple comme dans les suivants, nous supposons que la mémoire est initialement vide, ce qui occasionne un défaut de page pour les m premières références distincts.

2. Conclure sur la variation du nombre des défauts dans les deux cas.
3. Proposer une condition de stabilité qui doit être vérifiée par les algorithmes de paginations.

B. Remplacement optimal

Soit la suite de références w = 0 1 2 3 0 1 4 0 1 2 3 4.

4. Pour les valeurs de m = 3 et m = 4, calculer les C(OPT, m, w); OPT : optimal.
5. Donner l'inconvénient de cet algorithme.

C. Moins récemment utilisée

Soit la suite de références w = 0 1 2 3 0 1 4 0 1 2 3 4.

6. Pour les valeurs de m = 3 et m = 4, calculer les C(LRU, m, w); LRU : Least Recently Used.