Chapitre 3 : Gestion de la Mémoire

Dr Mandicou BA
mandicou.ba@esp.sn
http://www.mandicouba.net

Diplôme Universitaire de Technologie (DUT, 2^e année)
Diplôme Supérieure de Technologie (DST, 2^e année)
en Informatique

ECOLE SUPERIEURE POLYTECHNIQUE
www.esp.sn

Plan du Chapitre

Sommaire

Importance de la mémoire 1/1

- La mémoire principale est le lieu où se trouvent les programmes et les données quand le processeur les exécute
- La mémoire est une ressource importante qui doit être gérée avec attention
- Même si la quantité de mémoire d'un ordinateur a beaucoup augmentée, la taille des programmes s'accroit aussi
- La situation idéale serait de donner à chaque programmeur une mémoire infiniment grande, infiniment rapide, non volatile et, de plus, bon marché :
  - La technologie ne fournit pas de telles mémoires.
- Hiérarchisation de la mémoire:
  1. Une petite quantité de mémoire très rapide, chère et volatile (mémoire RAM)
  2. Beaucoup de gigabytes de mémoire plus lente, bon marché et non volatile
- Le SE se charge de gérer l'utilisation des différentes mémoires

Importance de la mémoire 2/2

- La mémoire physique sur un système d'exploitation se divise en deux catégories :
  1. **la mémoire vive** : composée de circuit intégrés, donc très rapide.
  2. **la mémoire de masse (secondaire)** : composée de supports magnétiques (disque dur, bandes magnétiques...), qui est beaucoup plus lente que la mémoire vive.
- Il n'en existe qu'un seul exemplaire et tous les processus actifs doivent la partager
- Si les mécanismes de gestion de ce partage sont peu efficaces l'ordinateur sera peu performant, quelque soit la puissance de son processeur
- Le processeur sera souvent interrompu en attente d'un échange qu'il aura réclamé
- Pareil si un programme viole les règles de fonctionnement de la mémoire

La mémoire et la gestion de la mémoire

- **Mémoire** : grand tableau de mots (octets), chacun possédant sa propre adresse.
- La CPU extrait les instructions de la mémoire en fonction de la valeur d'un compteur d'instructions.
- Système de gestion de la mémoire (**Memory manager**): partie du SE qui gère la hiérarchie de stockage
  - Suivre les parties de la mémoire qui sont utilisées ou non
  - Allouer/libérer espace mémoire aux processus
  - Contrôler le **swapping** entre la mémoire principale et le disque.

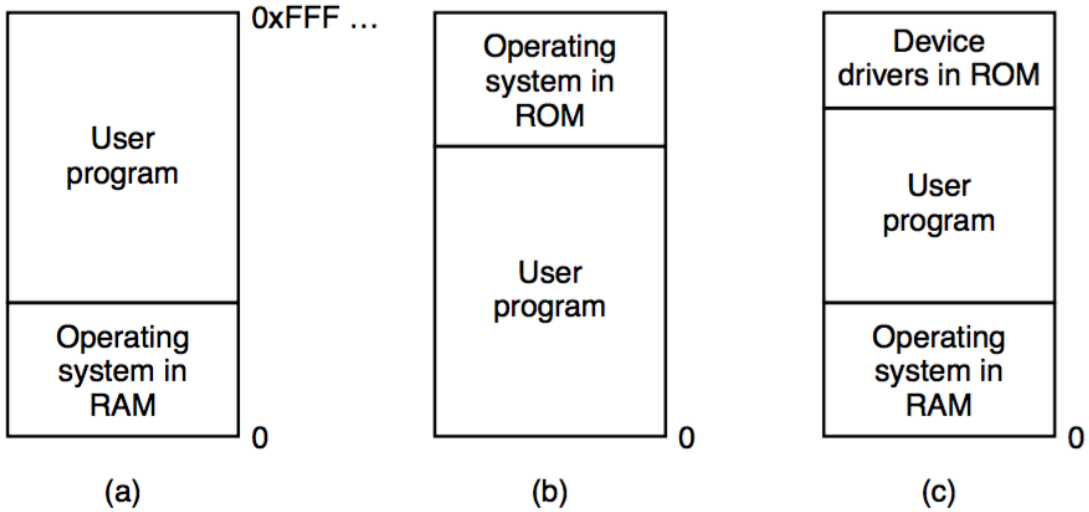
- Adresse logique ou adresse virtuelle (virtual address) : gérée par la CPU
- Adresse physique: adresse vue par l'unité de mémoire
- Memory Management Unit (MMU) : dispositif matériel qui fait la conversion des adresses virtuelles à physiques

Techniques de gestion mémoire

- La gestion de la mémoire principale de l'ordinateur est du ressort du système de gestion de la mémoire
- Les systèmes de gestion de la mémoire se classent en deux catégories :
  1. les systèmes qui peuvent déplacer les processus en mémoire secondaire pour trouver de l'espace (va-et-vient)
  2. les systèmes qui n'utilisent pas la mémoire secondaire :

Sommaire

- ☛ L'approche la plus simple pour gérer la mémoire consiste à n'accepter qu'un seul processus à la fois (monoprogrammation)
- ☛ Ce processus utiliser toute la mémoire disponible en dehors de celle qu'utilise le système.
- ☛ Cette approche était utilisée, par exemple, dans les premiers micro-ordinateurs IBM PC utilisant MS-DOS comme système d'exploitation



- Dans un système multiprogrammation, il faut gérer la répartition de l'espace entre les différents processus
- Cette partition peut être faite une fois pour toute au démarrage du système par l'opérateur de la machine, qui subdivise la mémoire en partitions fixes
- Chaque nouveau processus est placé dans la file d'attente de la plus petite partition qui peut le contenir
- Cette façon de faire peut conduire à faire attendre un processus dans une file, alors qu'une autre partition pouvant le contenir est libre
- L'alternative à cette approche consiste à n'utiliser qu'une seule file d'attente : dès qu'une partition se libère, le système y place le premier processus de la file qui peut y tenir
- Cette approche a été longtemps utilisée dans de gros ordinateurs IBM munis du système OS/360.

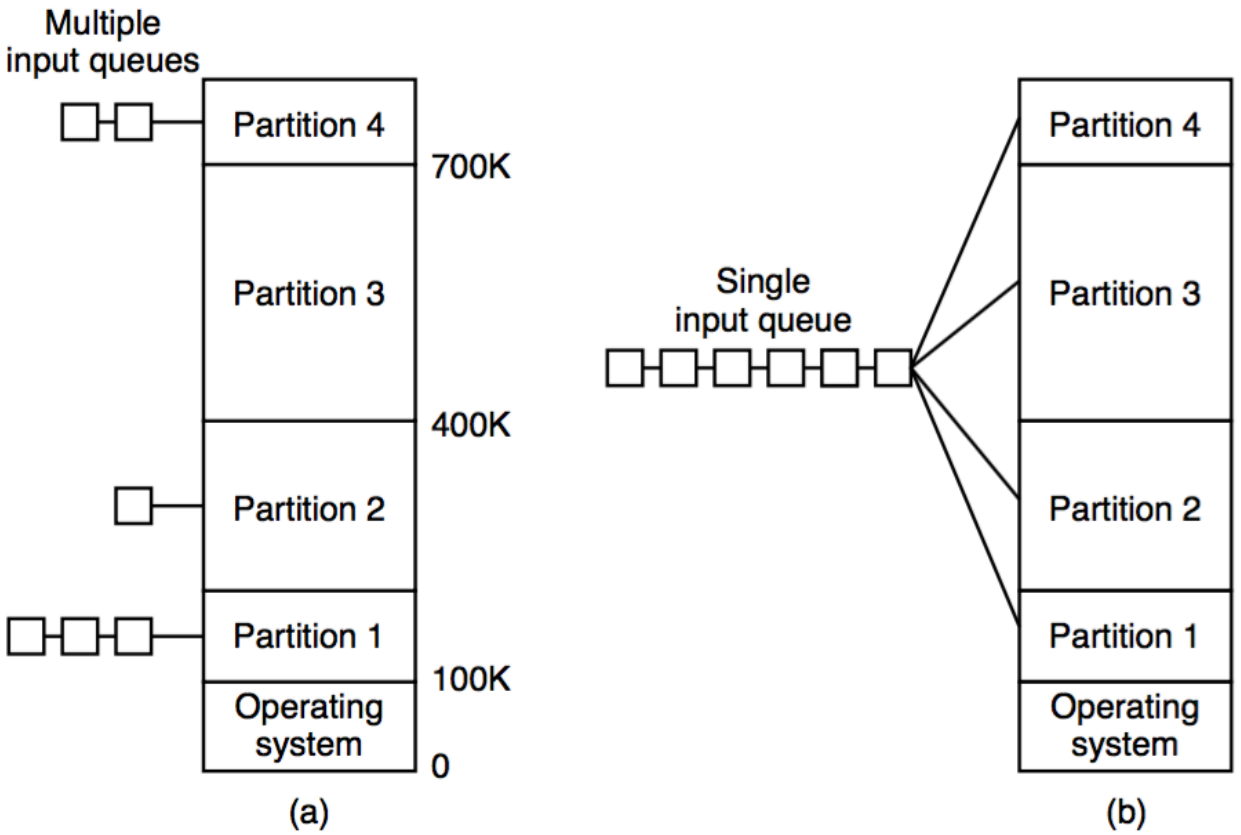


Figure: Partition fixe de la mémoire

- ☛ Le fait de ne plus fixer le partitionnement de la mémoire rend plus complexe la gestion de l'espace libre
- ☛ Cependant, il faut pouvoir trouver et choisir rapidement de la mémoire libre pour l'attribuer à un processus
- ☛ Il est donc nécessaire de mettre en œuvre des structures de données efficaces pour la gestion de l'espace libre

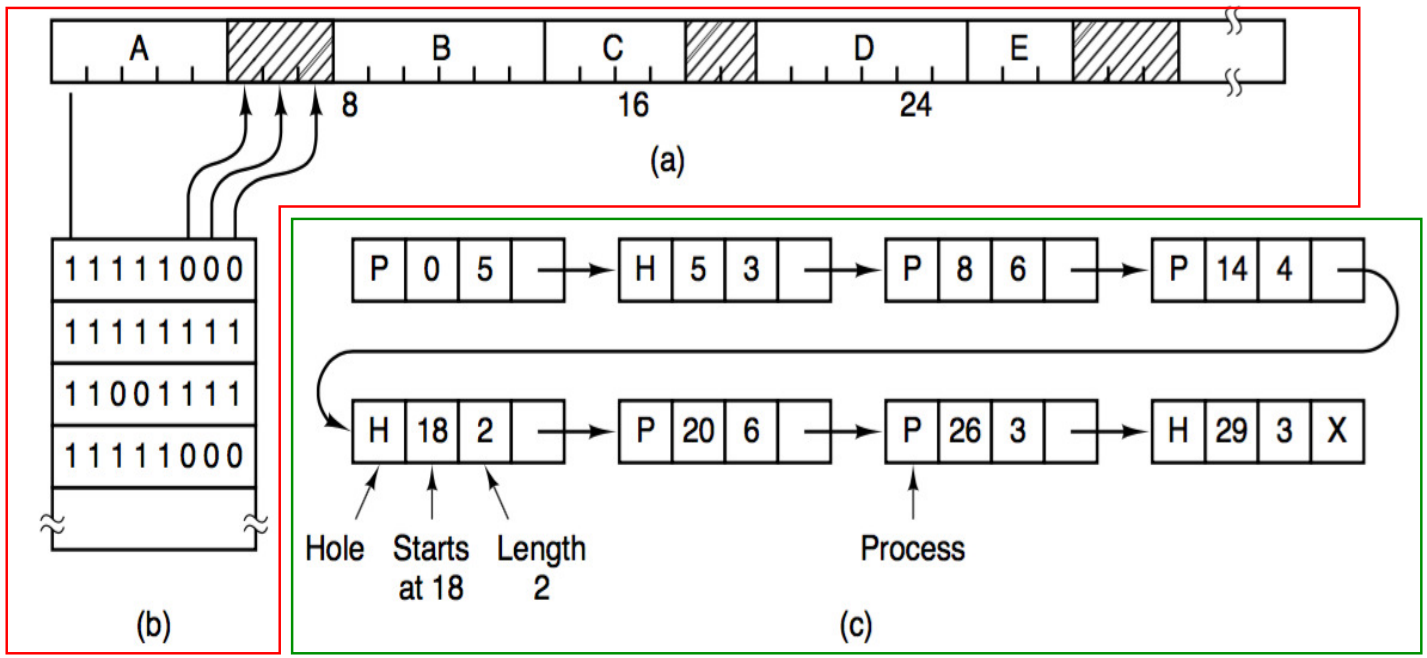
Sommaire

Notion de swapping

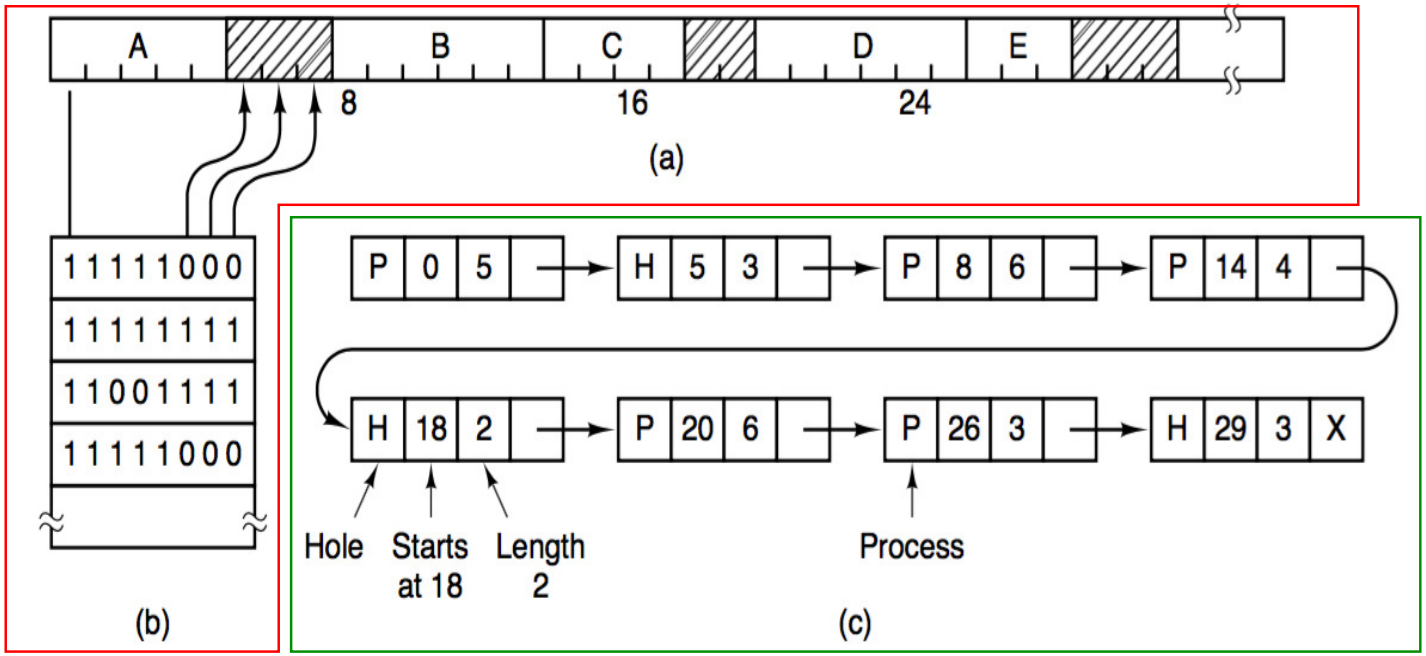
- Swapping : un système qui peut déplacer les processus sur le disque quand il manque d'espace
- Si on n'exploite pas au mieux l'espace libre en mémoire principale :
  - Le mouvement des processus entre la mémoire et le disque (va-et-vient) risque de devenir très fréquent,
  - Sachant que les accès au disque son très lents, les performances du système risquent alors de se détériorer rapidement.
- Pour exploiter au mieux l'espace libre en mémoire principale : partitionnement dynamique de la mémoire
- Le fait de ne plus fixer le partitionnement de la mémoire rend plus complexe la gestion de l'espace libre

Le swapping par tables de bits

- La mémoire est divisée en unités (quelques mots mémoire à plusieurs kilo-octets)
- A chaque unité on fait correspondre dans une table de bits :
  1. la valeur 1 si l'unité mémoire est occupée
  2. la valeur 0 si l'unité mémoire est libre

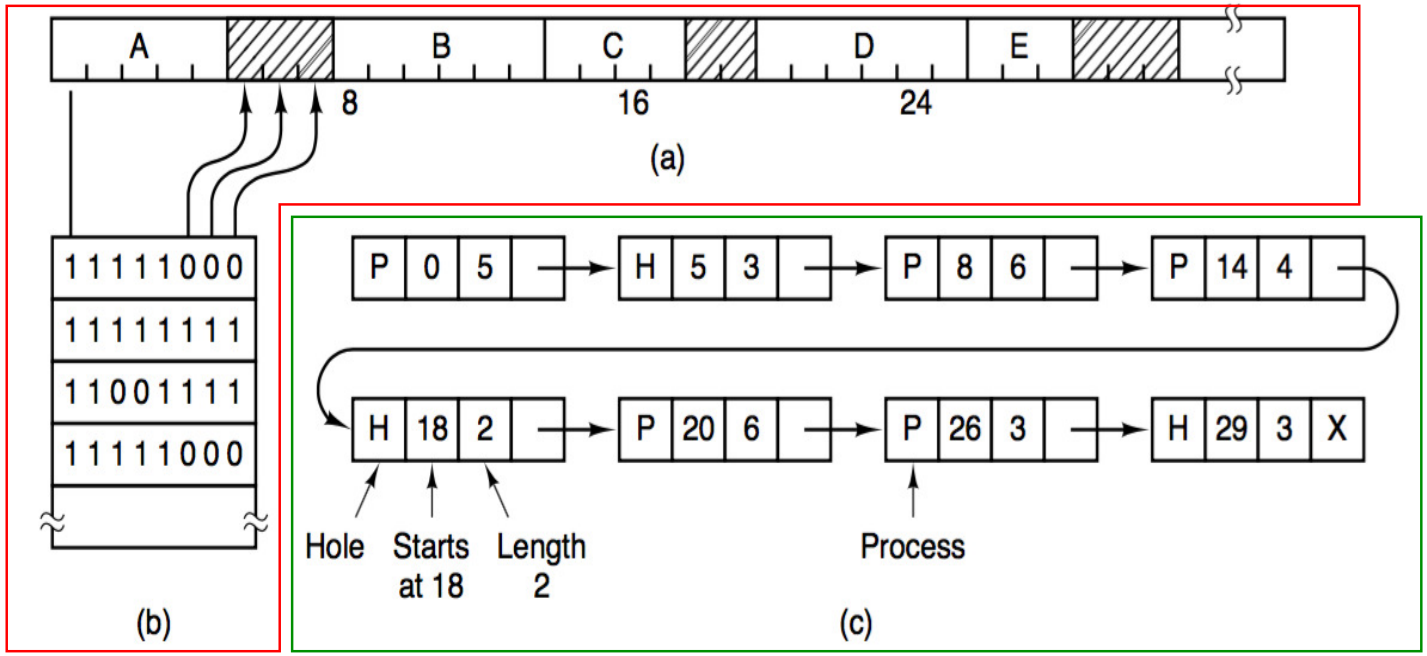


Le swapping par tables de bits



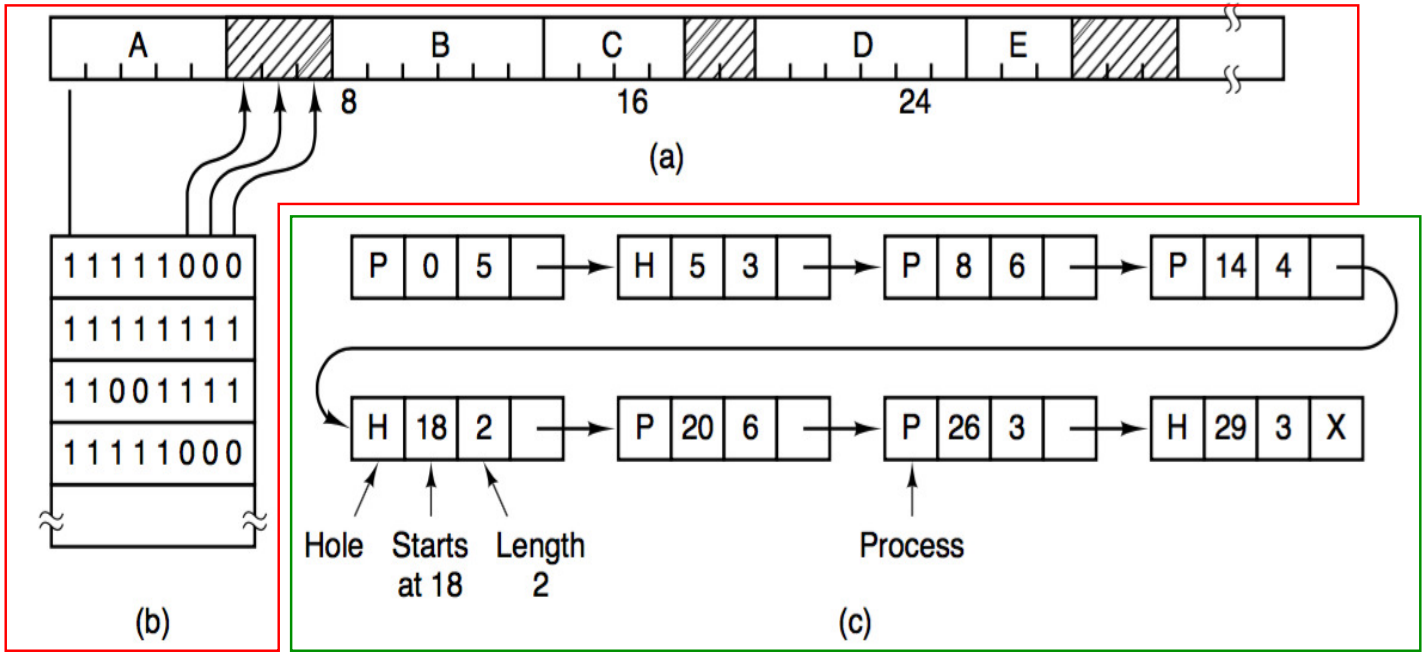
- **(a)** Une partie de la mémoire occupée par cinq processus (A, B, C, D, E) et trois zones libres
- Les régions hachurées sont libres.
- **(b)** La table de bits (0 = libre, 1 = occupée)
- **(c)** Liste chainée des zones libres.

Le swapping par tables de bits



- La figure (a) et (b) représentent un exemple de table de bits correspondant à l'occupation de la mémoire par cinq processus
- Pour allouer *k* unités contiguës le gestionnaire de mémoire doit parcourir la table de bits à la recherche de *k* zéro consécutifs
- Cette recherche s'avère donc lente pour une utilisation par le gestionnaire de la mémoire et est rarement utilisé à cet effet

Le swapping par listes chaînées



- Une autre solution consiste à chaîner les segments libres et occupés
- La figure (c) montre un tel chaînage, les segments occupés par un processus sont marqués (P) les libres sont marqués (H)
- La liste est triée sur les adresses, ce qui facilite la mise à jour.

Fusionner des segments libres

- Lorsqu'on libère la mémoire occupée par un segment, il faut fusionner le segment libre avec le ou les segments adjacents libres s'ils existent

- Quatre combinaisons de voisins possibles d'un processus *X* qui termine et libère le segment qu'il occupe

Choix de la zone libre 1/4

- On peut utiliser différents algorithmes pour le choix de la zone libre lors d'une allocation dans un système gérant une liste de segments
- L'objectif est d'éviter de trop fragmenter la mémoire, tout en ayant un algorithme relativement rapide

Algorithme de la première zone libre

- Choisir le premier segment libre de la liste qui est de taille suffisante, le segment est scindé en deux :
  1. une zone sera occupée par le nouveau segment occupé
  2. l'autre (s'il reste de l'espace inutilisé) sera un segment libre adjacent au premier
- Cet algorithme est rapide puisqu'il requiert peu de recherche.

Choix de la zone libre 2/4

- On peut utiliser différents algorithmes pour le choix de la zone libre lors d'une allocation dans un système gérant une liste de segments
- L'objectif est d'éviter de trop fragmenter la mémoire, tout en ayant un algorithme relativement rapide

Algorithme de la zone libre suivante

- Identique à la méthode précédente
- sauf que la prochaine recherche commence au segment qui suit celui dans lequel on s'est précédemment arrêté
- A la simulation ce dernier algorithme ne donne pas de meilleurs résultats que le premier

Choix de la zone libre 3/4

- On peut utiliser différents algorithmes pour le choix de la zone libre lors d'une allocation dans un système gérant une liste de segments
- L'objectif est d'éviter de trop fragmenter la mémoire, tout en ayant un algorithme relativement rapide

Algorithme du meilleur ajustement

- On cherche dans toute la liste la plus petite zone qui suffit à contenir la taille du segment à allouer
- On cherche à éviter de fragmenter la mémoire au dépend du temps
- Par contre, la recherche dans toute la liste nécessite beaucoup de temps
- Malheureusement cette méthode a tendance à trop fragmenter la mémoire du fait quelle a tendance à créer de courts espaces libres inutilisables.

Choix de la zone libre 4/4

- On peut utiliser différents algorithmes pour le choix de la zone libre lors d'une allocation dans un système gérant une liste de segments
- L'objectif est d'éviter de trop fragmenter la mémoire, tout en ayant un algorithme relativement rapide

Algorithme du plus grand résidu

- Consiste au contraire de la précédente à choisir toujours la plus grande zone libre
- cette méthode ne donne pas de meilleurs résultats

Sommaire

Approche par la mémoire virtuelle

- Le principe de la mémoire virtuelle consiste à considérer un espace d'adressage virtuel supérieur à la taille de la mémoire physique
- Sachant que dans cette espace d'adressage, et grâce au mécanisme de va-et-vient sur le disque, seule une partie de la mémoire virtuelle est physiquement présente en mémoire principale à un instant donné
- Ainsi, un peut gérer un espace virtuel beaucoup plus grand que l'espace physique **sans** avoir à gérer les changements d'adresses physiques des processus après un va-et-vient :
  - En effet, même après de multiples va-et-vient un processus garde la même adresse virtuelle
- C'est notamment très utile dans les systèmes multiprogrammations dans lesquels les processus qui ne font rien (la plupart) occupent un espace virtuel qui n'encombre pas nécessairement l'espace physique.

Approche par la mémoire virtuelle

- La plupart des architectures actuellement utilisées reposent sur des processeurs permettant de gérer un espace virtuel paginé :
  - L'espace d'adressage virtuel est divisé en **pages**
  - Chaque page occupée par un processus est :
    1. soit en mémoire physique
    2. soit dans le disque (va-et-vient)

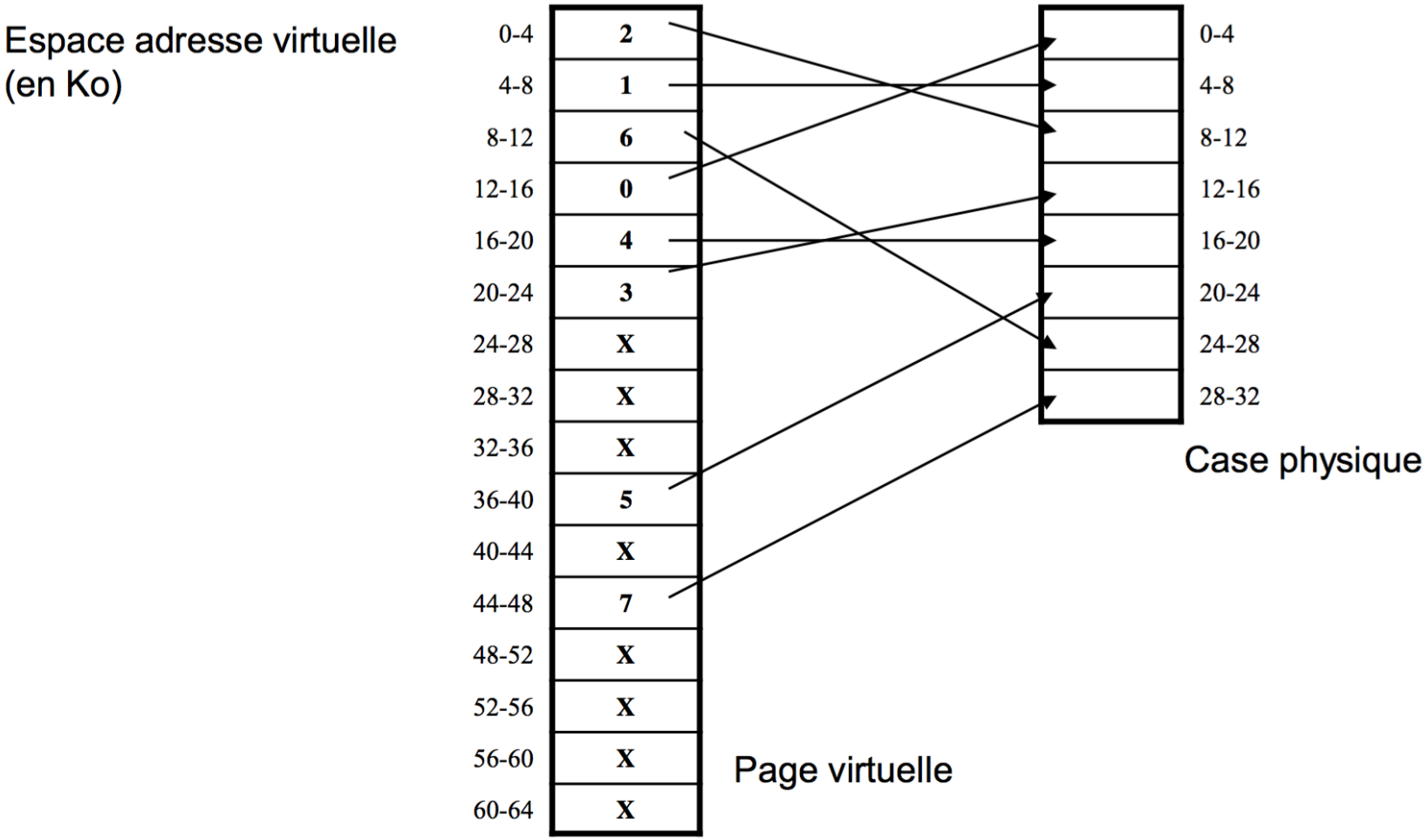
Exemple 1

- *Un PC peut générer des adresses de 16 bits de 0 à 64Ko. C'est l'adressage virtuelle. Mais ce PC n'a que 32Ko de mémoire physique.*
- *Conséquence : il est possible d'écrire un programme de 64Ko mais pas de le charger entièrement en mémoire*
- *L'image mémoire de tout le programme est stocké sur DD pour pouvoir charger, au besoin, les différentes parties dans la mémoire principale.*

Table de pages

- La **table de pages** réalise La correspondance entre une page virtuelle et la page physique à laquelle elle peut être associée (si elle est en mémoire physique)
- L'espace d'adressage virtuelle est divisé en petite unité appelée **page**.
- Les unités correspondantes de la mémoire physique sont les **cases** mémoires.
- **Pages** et **cases** ont toujours la même taille
  - Dans exemple suivant, page et case ont une taille de 4Ko
  - Avec 64Ko d'adresse ou mémoire virtuelle et 32Ko de mémoire physique, on a 16 pages virtuelles et 8 cases

Table de pages



Conception des systèmes paginés

- Système paginé : processus lancé sans que leur page ne soit en mémoire
  - Le processeur essaie d'exécuter la première instruction
    - Pas de page, le SE génère défaut de page
  - Cela se produit plusieurs fois, OS charge la page contenant cette instruction. Puis le processeur dispose de la page et exécution se continue
- Pagination à la demande. Page chargée à la demande et non à l'avance

Conception des systèmes paginés

- Ensemble de travail : ensemble de pages utilisées pendant un certain temps par le processus
- Si l'ensemble de travail est entièrement en mémoire, le processus s'exécute sans générer de défaut page.
- Système à temps partagé : processus recopié sur DD (page retirées de la mémoire) pour permettre à d'autres processus de disposer du processeur

Taille des pages

- Choisi par concepteur de l'OS
- Matériel possède page de 512 octets
- Plus la taille des pages est élevée, plus il y a de code inutile en mémoire.
- Par ailleurs, plus la taille des pages est faible, plus grande sera la table des pages.

Descripteur de pages

- Chaque entrée de table de pages consiste en un **descripteur de page** qui contient généralement ces informations (au moins) :
  - Numéro de la page en mémoire physique si celle-ci est présente en mémoire
  - Un bit qui indique la présence de la page en mémoire
  - Un bit de modification : qui mémorise le fait qu'une page a été modifiée :
    - ceci permet au gestionnaire de mémoire de voir s'il doit la sauver sur le disque dans le cas où il veut récupérer l'espace qu'elle occupe pour charger une autre page
  - Un bit de référencement : qui est positionné si on fait référence à un mot mémoire de la page :
    - ceci permet au système de trouver les pages non référencées qui seront les meilleures candidates pour être retirées de la mémoire physique s'il y a besoin et manque de mémoire

Exécution d'une instruction

- Lorsque le processeur doit exécuter une instruction qui porte sur un mot mémoire donné dont il a l'adresse virtuel, il cherche dans la table des pages l'entrée qui correspond à la page contenant le mot :
  1. Si la page est présente en mémoire il lit le mot
  2. sinon il déclenche un déroutement de type *défaut de page*
- A la suite de ce déroutement, le système de gestion de la mémoire est appelé afin de charger la page manquante à partir du disque (va-et-vient)
- La consultation de la table de pages est à la charge du processeur (tâche cablée)
- Le système d'exploitation intervient lors d'un défaut de page pour prendre en charge le va-et-vient.

Sommaire

- Lorsque le système de gestion de la mémoire manque de mémoire physique, il doit évincer des pages en les sauvant si besoin est (dans le cas où elles ont été modifiées) dans le disque (va-et- vient)
- Ceci afin de les remplacer par des pages demandés par le processeur
- Le problème du choix des pages à remplacer se pose alors.

Remplacement de page optimal

- L'algorithme consiste à retirer la page qui sera référencée dans le plus lointain avenir
- Ce qui suppose de connaître la date à laquelle chaque page sera référencée dans le futur :
  - irréalisable en pratique

- Lorsque le système de gestion de la mémoire manque de mémoire physique, il doit évincer des pages en les sauvant si besoin est (dans le cas où elles ont été modifiées) dans le disque (va-et- vient)
- Ceci afin de les remplacer par des pages demandés par le processeur
- Le problème du choix des pages à remplacer se pose alors.

Remplacement de la page non récemment utilisée

- L'algorithme essaye d'abord de retirer une page non référencée
- S'il n'en trouve pas, il retire une page référencée mais pas modifiée
- S'il n'en existe pas, il retire une page quelconque parmi celles qui restent.
- Périodiquement le bit de référence est remis à 0 afin de différencier les pages récemment référencées des autres pages

- Lorsque le système de gestion de la mémoire manque de mémoire physique, il doit évincer des pages en les sauvant si besoin est (dans le cas où elles ont été modifiées) dans le disque (va-et- vient)
- Ceci afin de les remplacer par des pages demandés par le processeur
- Le problème du choix des pages à remplacer se pose alors.

Premier entré, premier sorti

- Cet algorithme est peu utilisé tel quel, car la page la plus ancienne n'est pas forcément la moins utilisée dans le futur.

- Lorsque le système de gestion de la mémoire manque de mémoire physique, il doit évincer des pages en les sauvant si besoin est (dans le cas où elles ont été modifiées) dans le disque (va-et- vient)
- Ceci afin de les remplacer par des pages demandés par le processeur
- Le problème du choix des pages à remplacer se pose alors.

Remplacement avec seconde chance

- Cet algorithme utilise le principe du Premier entré, premier sorti modifié :
  - si la page en tête de file est référencée, elle est remise en queue de la file
  - puis la suivante est traitée de la même façon, sinon la page est choisie pour être évincée

- Lorsque le système de gestion de la mémoire manque de mémoire physique, il doit évincer des pages en les sauvant si besoin est (dans le cas où elles ont été modifiées) dans le disque (va-et- vient)
- Ceci afin de les remplacer par des pages demandés par le processeur
- Le problème du choix des pages à remplacer se pose alors.

Remplacement avec horloge

- Cet algorithme est le même que le précédent sauf qu'il utilise une liste circulaire pour implanter la file.

- Lorsque le système de gestion de la mémoire manque de mémoire physique, il doit évincer des pages en les sauvant si besoin est (dans le cas où elles ont été modifiées) dans le disque (va-et- vient)
- Ceci afin de les remplacer par des pages demandés par le processeur
- Le problème du choix des pages à remplacer se pose alors.

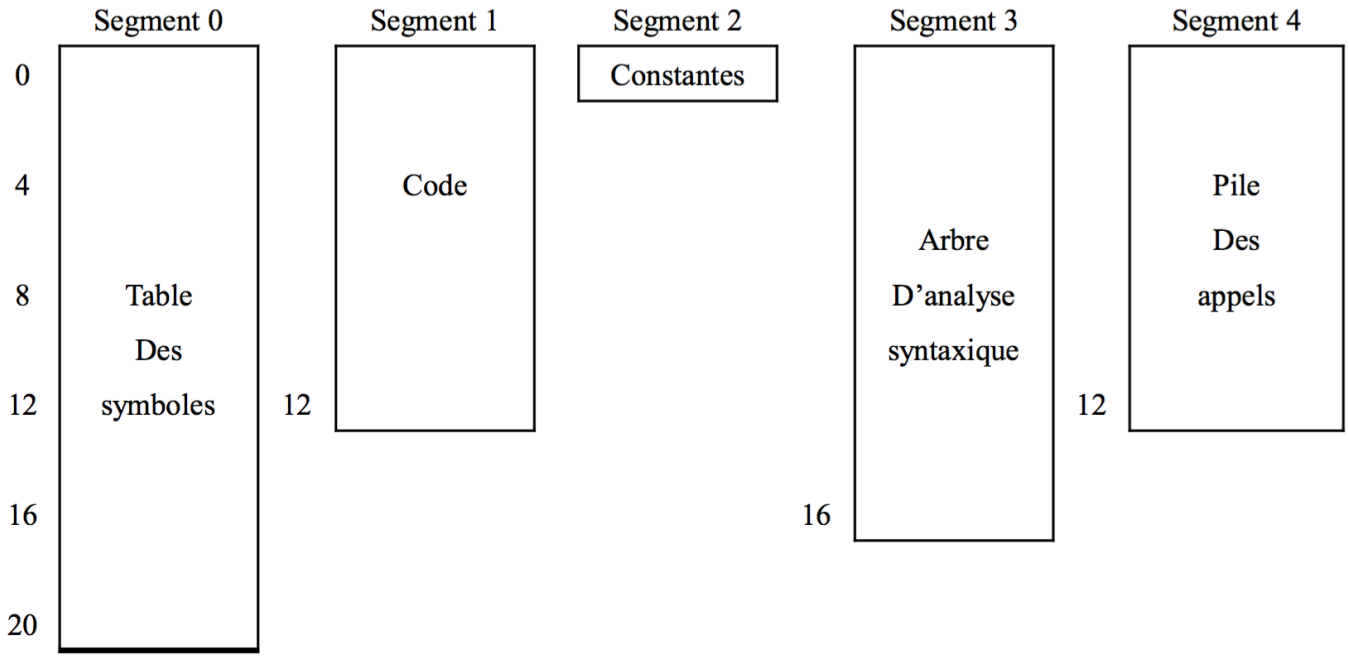
Remplacement de la page la moins récemment utilisée

- Consiste à supprimer la page restée inutilisée pendant le plus de temps.

Sommaire

- La pagination : La mémoire virtuelle étudiée jusqu'ici est à une dimension, les adresses virtuelles sont comprises entre 0 et une adresse maximale
- La segmentation : plusieurs espaces d'adresses indépendants appelés segments
- Chaque segment est une suite d'adresses continus de 0 à une adresse maximale autorisée
- Les segments ont des tailles différentes qui varient en cours d'exécution, il représente des espaces d'adressage séparés

- Exemple d'une mémoire segmentée utilisée pour les tables du compilateur



- La segmentation de la mémoire permet de traiter la mémoire non plus comme un seul espace d'adressage unique :
  - mais plutôt comme un ensemble de segments (portions de la mémoire de taille variable), ayant chacune son propre espace d'adressage
- Ceci permet aux processus de simplifier considérablement le gestion de mémoire propre.
- Le segment est une entité logique que le programmeur doit manipuler
- Il peut contenir une procédure, un tableau, mais en général, il ne contient pas un mélanger d'objets de types différents
- La segmentation simplifie le partage des procédures et des données entre plusieurs procédures.

Sommaire

Considérations	Pagination	Segmentation
Le programmeur doit-il connaître la technique utilisée ?	NON	OUI
Combien y a-t-il d'espaces d'adressage linéaire ?	1	Plusieurs
L'espace d'adressage total peut-il dépasser la taille de la mémoire physique ?	OUI	OUI
Les procédures et les données peuvent-elles être distinguées et protégées séparément ?	NON	OUI

Considérations	Pagination	Segmentation
Peut-on traiter simplement les tables dont la taille varie ?	NON	OUI
Peut-on traiter simplement les tables dont la taille varie ?	NON	OUI
Pourquoi cette technique a-t-elle été inventée ?	Pour obtenir un grand espace d'adressage linéaire sans avoir à acheter de la mémoire physique	Pour permettre la séparation des programmes et des données dans des espaces d'adressages logiquement indépendants et pour

Chapitre 3 : Gestion de la Mémoire

Dr Mandicou BA
mandicou.ba@esp.sn
http://www.mandicouba.net

Diplôme Universitaire de Technologie (DUT, 2^e année)
Diplôme Supérieure de Technologie (DST, 2^e année)
en Informatique

ECOLE SUPERIEURE POLYTECHNIQUE

www.esp.sn