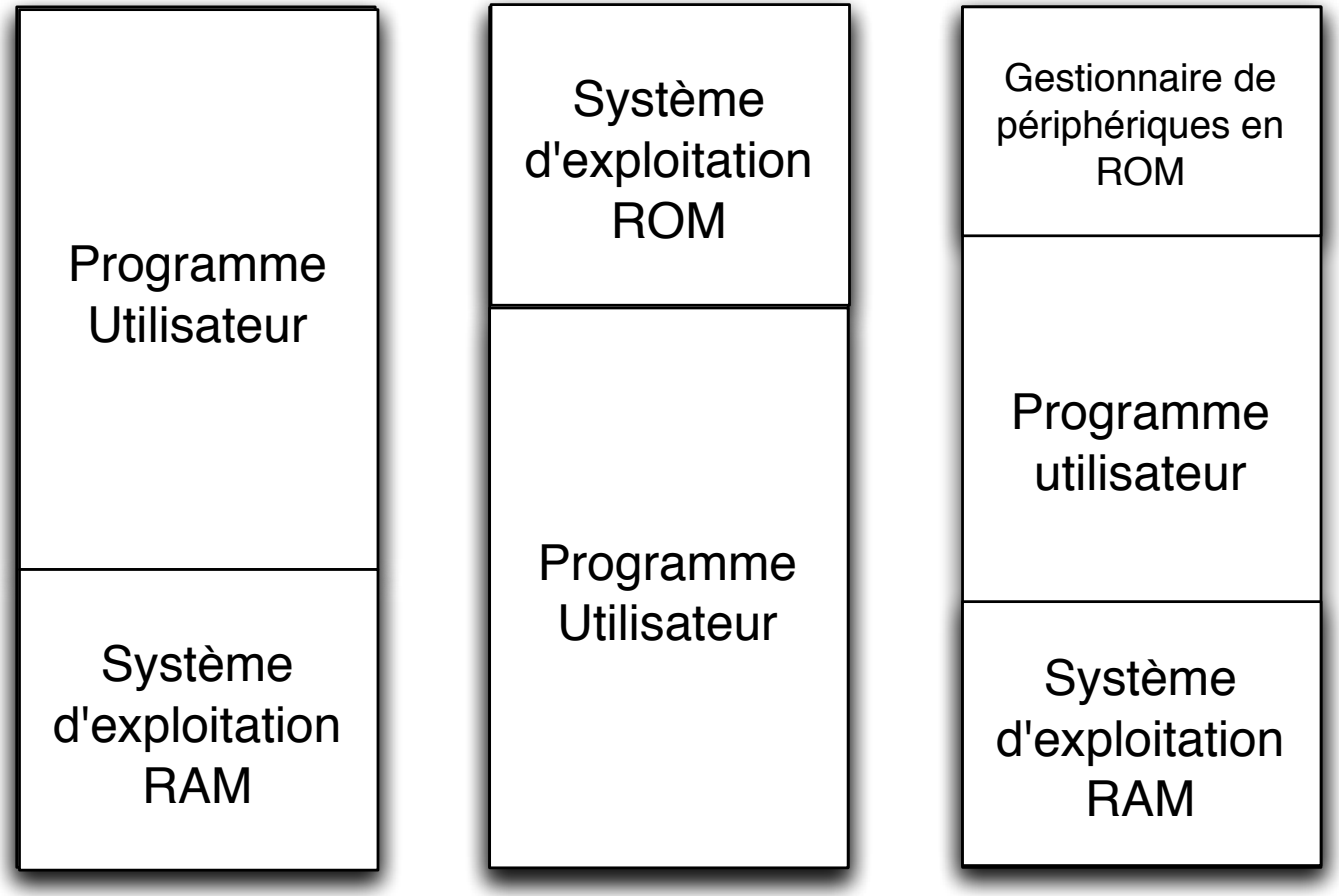
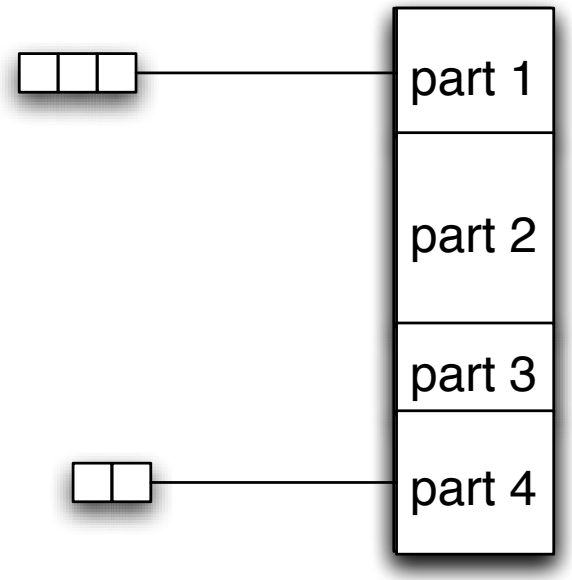
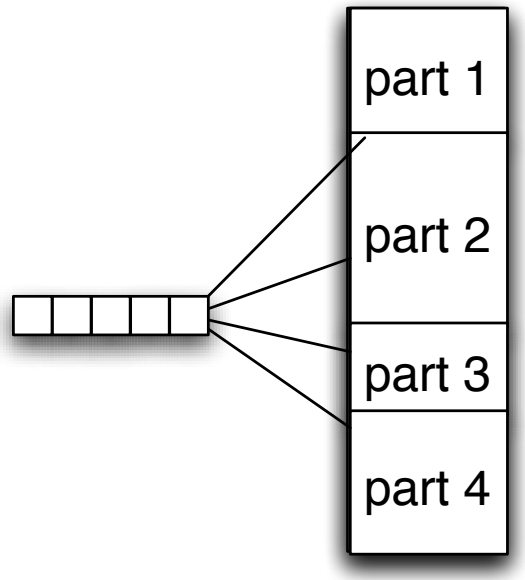
Gestion de la mémoire

Cas de la monoprogrammation



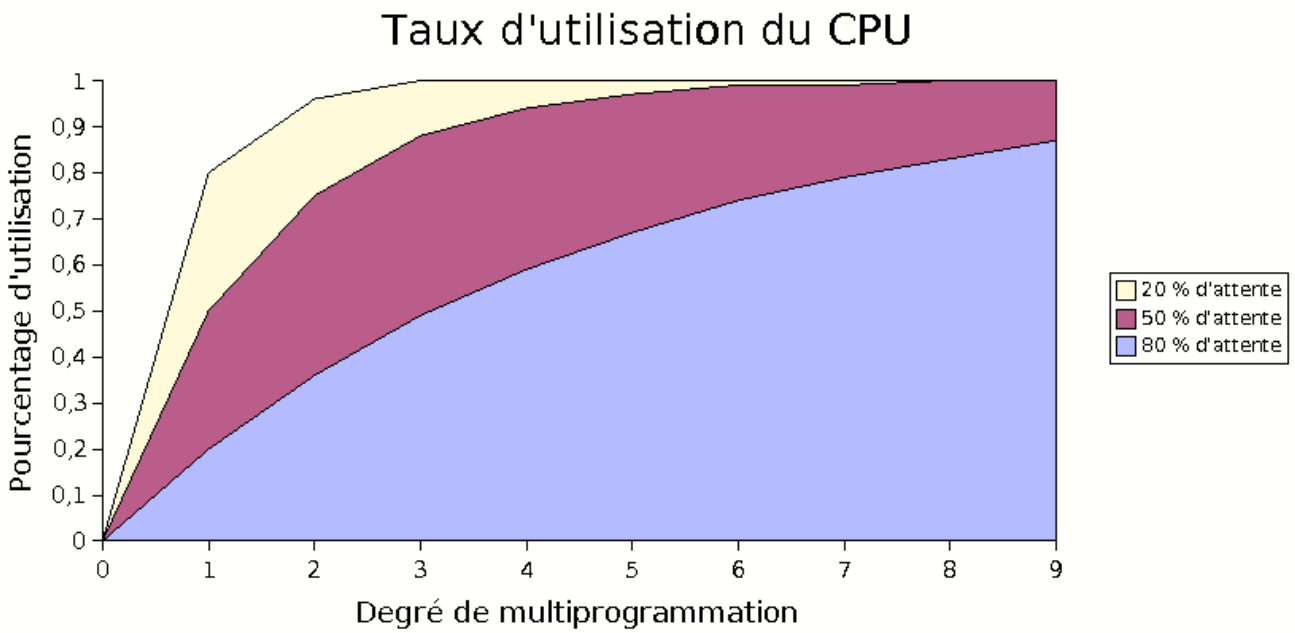
Cas de la multiprogrammation





Modélisation simplifié

- Soit p la probabilité pour un processus d'être en attente
- Pour n processus : probabilité d'attente est p^n
- D'où Taux d'utilisation du CPU : $1 - p^n$



La multiprogrammation

Sur des systèmes à temps partagé

- Quand la mémoire est insuffisante pour maintenir tous les processus courants actifs :
  - Nécessité de sauvegarder les processus sur le disque
  - Et de les charger dynamiquement
- Deux approches existent :
  - Le va et vient
    - On considère chaque processus dans son intégralité
    - Exécution et sauvegarde sur le disque
  - La mémoire virtuelle
    - Exécution des processus, même quand ils sont partiellement en mémoire

Le va et vient

- Chargement des processus complets
- Puis sauvegarde complet sur le disque
- Au chargement : nécessité de relocaliser
- Différence avec partition de taille fixe :
  - La taille varie en fonction des chargements
  - Cela complique l'allocation et la libération

Le va et vient

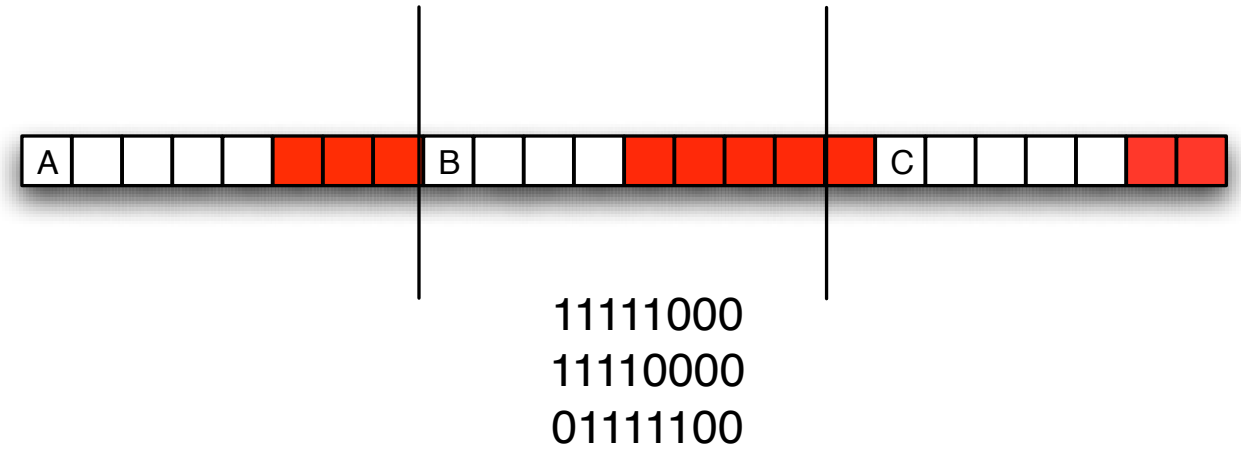
- Quand plusieurs trous en mémoire : technique du compactage de mémoire mais coûteux
- Quand la taille des processus est fixe
  - Le S.E. alloue exactement la taille nécessaire
- Quand un programme à besoin d'augmenter sa taille
  - Soit il y a de la place juste à coté : aucun problème
  - Soit il est contigu à un autre processus : déplacer tout le processus
  - Sinon, s'il n'y a plus de place en mémoire : attente ou terminaison

Gestion avec tableaux de bits

- Les tableaux de bits
- La mémoire est répartie en unités d'allocation
- Chaque unité d'allocation possède une taille comprise entre quelques octets à plusieurs kilooctets
- A chaque unité correspond un bit dans le tableau :
  - 0 si l'unité est vide
  - 1 sinon
- Problème : trouver la bonne taille pour les unités

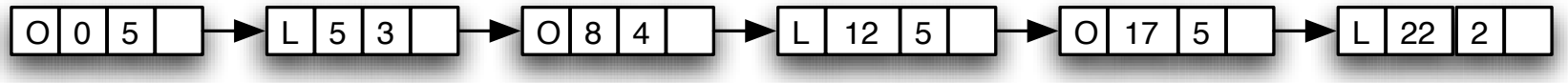
Exemple

- Considérons des unités d'allocations de 8 bits
- 3 processus A, B, et C occupant respectivement 5, 4 et 6 unités et 3 emplacements libres de taille 3, 5 et 2



Gestion par listes chaînées

- Liste chaînée des segments mémoires
- Chaque entrée indique
  - L'état du segment mémoire : libre (L), occupé (O)
  - l'adresse à laquelle il débute,
  - et sa longueur



La gestion mémoire ?

- Où placer les processus ?
- Dans quelle espace libre ?
- Peut-on optimiser la taille des espaces libres ?
- Optimiser la recherche d'un espace libre ?

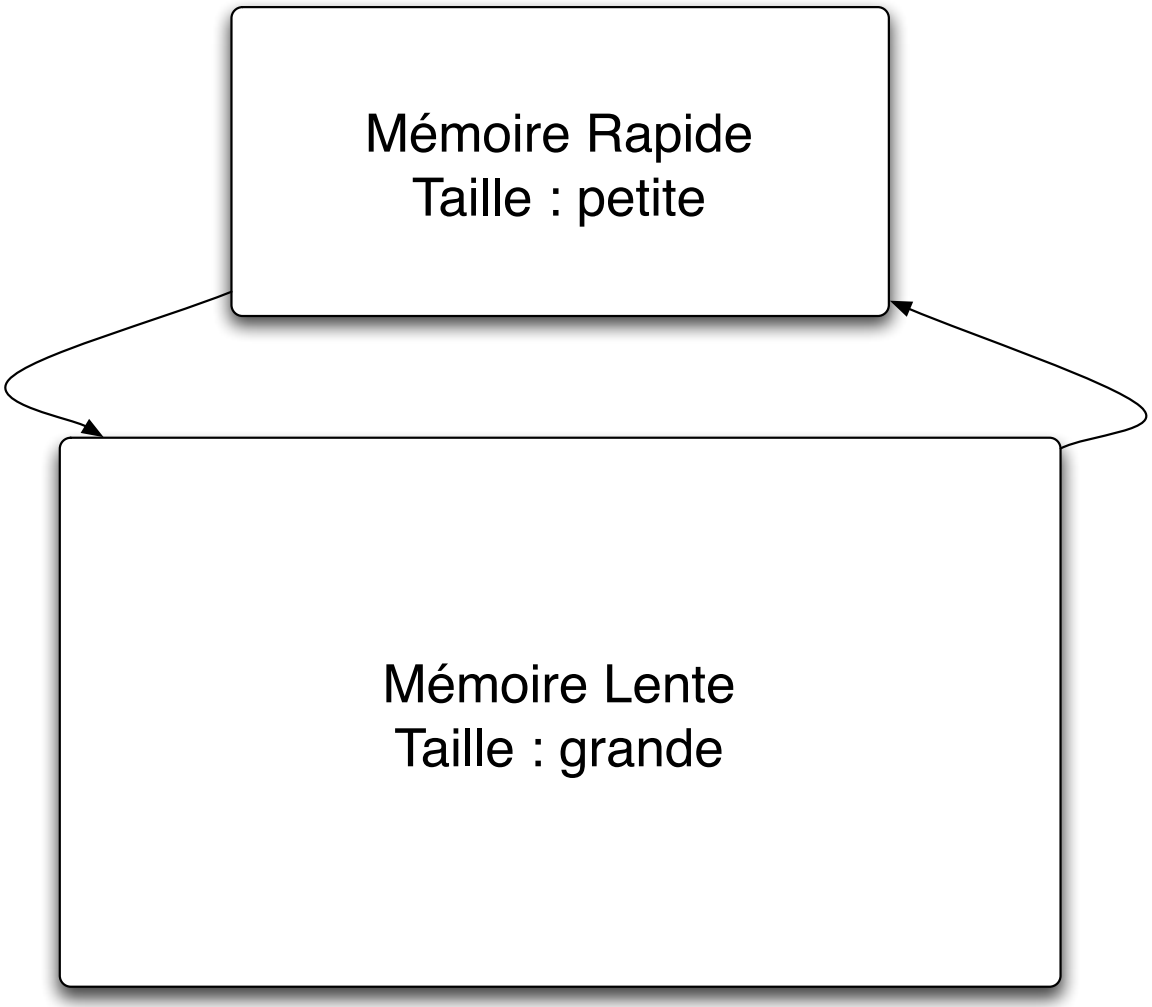
Algorithmes

- Firtst Fit
- Next Fit
- Best Fit
- Worst Fit

La mémoire virtuelle

- La taille d'un programme + données + pile peut être supérieur à la capacité mémoire physique
- Le S.E. conserve en mémoire les parties d'un programme en cours d'exécution, le reste est stocké sur le disque
- Technique utilisée : **la pagination**

La pagination



Le défaut de page

- Le défaut de page = accès à une page non présente en mémoire rapide
- Dans ce cas,
  - Le S.E. choisi une page en mémoire rapide
  - Ecrit, si nécessaire, son contenu en mémoire lente
  - Charge en mémoire la page qui faisait défaut

Algorithme de Belady

- A chaque page correspond le numéro de la ligne du programme ayant besoin de cette page
- Quand un défaut de page se produit, on remplace la page qui porte le plus grand numéro (c'est-à-dire celle qui sera utilisée le plus tard)
- Problème : impossible à implémenter
- Par simulation : on obtient le résultat de référence pour faire les comparaisons

Algorithmes

- LRU (moins récemment utilisé)
- NRU (pas récemment utilisé)
- FIFO (premier entré premier sorti)
- LIFO (dernier entré premier sorti)
- FWF (Vider quand c'est plein)
- NFU (pas fréquemment utilisé)
- 2eme chance

Algorithme NRU

- 2 bits d'états sont associés à chaque page
  - Le bit R est mis à 1 quand la page est référencée (lue ou écrite)
  - Le bit M est mis à 1 quand elle est modifiée
- Chaque page peut etre classée
  - Classe 0 : non référencée, non modifiée
  - Classe 1 : non référencée, modifiée
  - Classe 2 : référencée, non modifiée
  - Classe 3 : référencée, modifiée
- L'algorithme NRU choisit au hasard une page de classe basse pour la remplacer.

Algorithme de la seconde chance

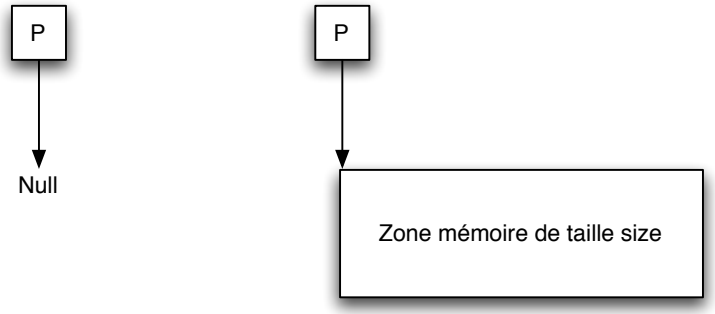
- Liste chainée des pages
- Lecture du bit R
- Si une page est ancienne et non référencée (R=0) alors on la remplace
- Sinon on met R à 0, puis on la place en fin de liste et on continue la recherche.

Algorithme NFU

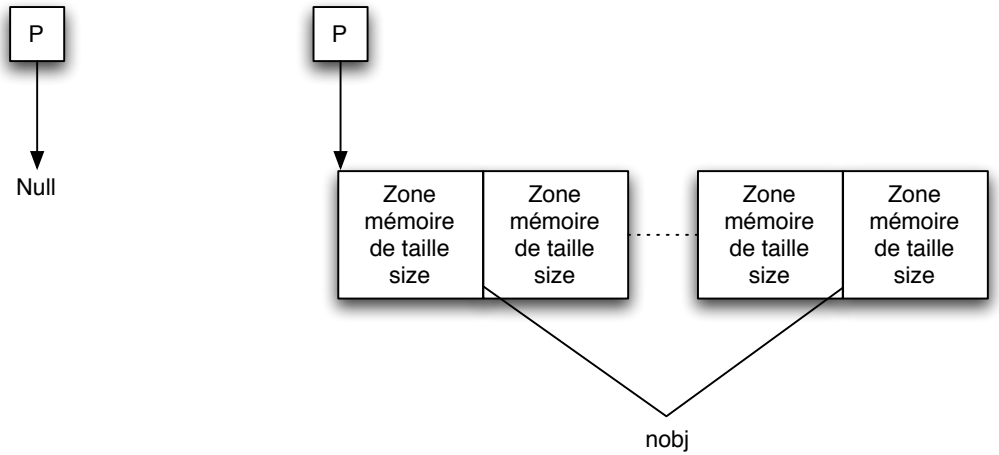
- Une solution pratique au LRU
- Associe un compteur à chaque page
- A chaque interruption d'horloge, le bit R de chaque page est ajouté à son compteur
- Celle avec le plus petit compteur qui est remplacée

Allocation Dynamique

```
void * malloc(size_t size)
```

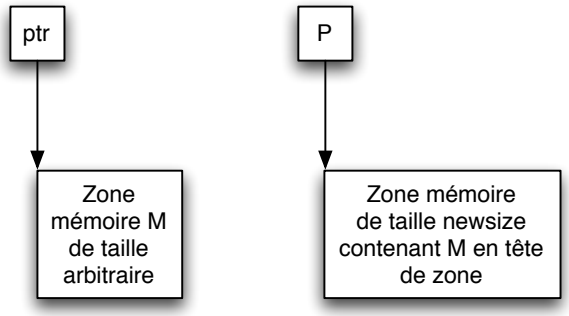



```
void * calloc(size_t nobj, size_t size)
```



Réallocation et libération

```
void * realloc(void* ptr, size_t newsize)
```




```
void free(void* ptr)
```