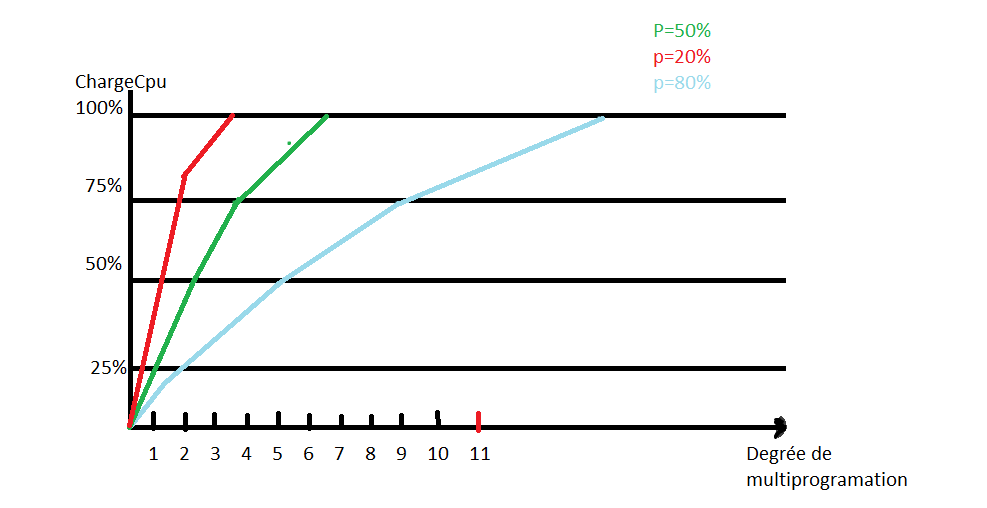
Gestion de la mémoire

Justification de la multiprogrammation

Soit p la proba un processus peut etre en attente



Quand la mémoire est insuffisante pour maintenir tous les processus actifs,

-On sauvegarde des processus

-On charge un processus

2 approches complémentaires

1. Va et vient
   1. Chaque processus est considéré dans son integralité
   2. Exécution/sauvegarde sur disque
2. La mémoire mutuelle
   1. Exécution des processus , même s’ils ne que partiellement en mémoire

Le va et vient

* Chargement couplet des processus
* Sauvegarde couplet sur disque
* Au chargement nécessite de le localiser
* Apparition de « trou »en mémoire
  + Technique de compactage mais très couteux
* Quand la taille du processus fixe, le SE alloue exactement la taille nécessaire

//Schema troue mémoire

Quand un processus a besoin de monter sa taille :

-Soit il y a suffisamment de place libre a coté du processus (contigue)

-Si il est contigue , a un autre processus

-Déplacer tout le processus

-Si il n’y pas suffisamment de place

-Attente/terminaison

Gestion

Tableau de bits

La mémoire est répartie en unités d’allocation (UA)

Chaque UA possede une taille

A chaque UA est associé un bit dans le tableau

0 Si l’UA est vide

1 sinon

Gestion par liste chainée

-Liste chainée des segments mémoire

-Chaque entrée indique :

->L’état du segment (libre ou occupé)

->L’adresse de début du segment

->La taille du segment

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| A | A | | A | A | | A |  |  |  | B | B | | B | | B |  |  | |  |  | |  | C | C | | C | C | | C |  | | |  | |
| O | | 0 | | | 5 | | L | 5 | 3 | O | | 8 | | 4 | | L | | 12 | | | 5 | | O | | 7 | | | 5 | | | L | 22 | | 2 |

Question :

-Où placer le processus a charger ?

-Dans quel espace libre

-Peut-on optimiser la taille des espaces libres ?

-Peut-on optimiser la recherche d’un espace libre

Algorithme

* First fit (on repart depuis le début)
* Next fit (on garde la position)
* Best fit (on rempli au mieux en observant du début a la fin)
* Worst fit

Exemple :

First fit

Chargement de E (2UA)

Chargement de F (4UA)

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| A | A | A | E | E |  | B | B | B | B | B |  |  | C | C | C | C | F | F | F | F | D | D |  |  |

Next fit

Chargement de E (4UA)

Chargement de F (2UA)

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| A | A | A |  |  |  | B | B | B | B | B |  |  | C | C | C | C | E | E | E | E | D | D | F | F |

Best fit

Chargement de E (2UA)

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| A | A | A |  |  |  | B | B | B | B | B | E | E | C | C | C | C |  |  |  |  | D | D |  |  |

Worst fit

Chargement de E (2UA)

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| A | A | A |  |  |  | B | B | B | B | B |  |  | C | C | C | C | E | E |  |  | D | D |  |  |

La mémoire virtuel

* La taille d’un programme + données +pile peut-être supérieur à la capacité mémoire physique
* Le système d’exploitation conserve en mémoire les parties d’un programme en cours d’exécutions, le reste est stocké sur disque
* Pagination

La pagination

Mémoire

Rapide

m

m<<<M

Mémoire

Lente

M

Le défaut de page

C’est un accès à une page mémoire non présente en mémoire rapide.

Dans ce cas :

1. Le systéme d’exploitation choisit une paye en mémoire rapide
2. Ecrit, si nécessaire son contenu en mémoire lente.
3. Charge en Mémoire rapide à l’emplacement libéré 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 page se produit, on remplace la page qui porte le plus grand numéro (celle utilisé le plus tard)
* Non implémentable
* Utilité : fournir un échantillon de comparaison

Soit une mémoire rapide pouvant contenir 4 pages mémoire et la séquence d’utilisation des pages :  = 1,4,3,1,2,4,5,4,3,7,5,6,2,3,1,4,3,5,7,6,2,8,4,3

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 1 | 1 | 1 | 1 | 1 | 1 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 7 | 7 | 7 | 8 | 8 | 8 |
|  | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 7 | 7 | 6 | 6 | 6 | 6 | 6 | 6 | 6 | 6 | 6 | 2 | 2 | 2 | 2 |
|  |  | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 |
|  |  |  |  | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 1 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 |

12 défaut page

Stratégie (implémentables !)

* FIFO
* LIFO (Last in First Out , le dernier chargé , le premier évincé)
* LRU(moins récemment utilisé)
* Algo de la seconde chance
* FWF (Vider quand c’est plein)

FIFO :  = 1,4,3,1,2,4,5,4,3,7,5,6,2,3,1,4,3,5,7,6,2,8,4,3

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 1 | 1 | 1 | 1 | 1 | 1 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 1 | 1 | 1 | 1 | 6 | 6 | 6 | 6 | 3 |
|  | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 7 | 7 | 7 | 7 | 7 | 7 | 4 | 4 | 4 | 4 | 2 | 2 | 2 | 2 |
|  |  | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 6 | 6 | 6 | 6 | 6 | 5 | 5 | 5 | 5 | 8 | 8 | 8 |
|  |  |  |  | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 3 | 3 | 3 | 3 | 7 | 7 | 7 | 7 | 4 | 4 |

17 DP

LIFO :  = 1,4,3,1,2,4,5,4,3,7,5,6,2,3,1,4,3,5,7,6,2,8,4,3

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 1 | 4 | 3 | 1 | 2 | 4 | 5 | 4 | 3 | 7 | 5 | 6 | 2 | 3 | 1 | 4 | 3 | 5 | 7 | 6 | 2 | 8 | 4 | 3 |
| 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |  |
|  | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 |  |
|  |  | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 |  |
|  |  |  |  | 2 | 2 | 5 | 5 | 5 | 7 | 5 | 6 | 2 | 2 | 2 | 2 | 2 | 5 | 7 | 6 | 2 | 8 | 8 |  |

14 DP

LRU :  = 1,4,3,1,2,4,5,4,3,7,5,6,2,3,1,4,3,5,7,6,2,8,4,3

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 1 | 4 | 3 | 1 | 2 | 4 | 5 | 4 | 3 | 7 | 5 | 6 | 2 | 3 | 1 | 4 | 3 | 5 | 7 | 6 | 2 | 8 | 4 | 3 |
| 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 3 | 3 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  | 3 | 3 | 3 | 3 | 5 | 5 | 5 | 5 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  | 2 | 2 | 2 | 2 | 2 | 7 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |

19 DP

FWF :  = 1,4,3,1,2,4,5,4,3,7,5,6,2,3,1,4,3,5,7,6,2,8,4,3

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 1 | 4 | 3 | 1 | 2 | 4 | 5 | 4 | 3 | 7 | 5 | 6 | 2 | 3 | 1 | 4 | 3 | 5 | 7 | 6 | 2 | 8 | 4 | 3 |
| 1 | 1 | 1 | 1 | 1 | 1 | 5 | 5 | 5 | 5 | 5 | 6 | 6 | 6 | 6 | 4 | 4 | 4 | 4 | 6 | 6 | 6 | 6 | 3 |
|  | 4 | 4 | 4 | 4 | 4 |  | 4 | 4 | 4 | 4 |  | 2 | 2 | 2 |  | 3 | 3 | 3 |  | 2 | 2 | 2 |  |
|  |  | 3 | 3 | 3 | 3 |  |  | 3 | 3 | 3 |  |  | 3 | 3 |  |  | 5 | 5 |  |  | 8 | 8 |  |
|  |  |  |  | 2 | 2 |  |  |  | 7 | 7 |  |  |  | 1 |  |  |  | 7 |  |  |  | 4 |  |

21 DP

Algorithme de la seconde chance

-Liste chainée des pages

-Pour chaque page, on a un bit R

- On parcourt la liste des pages depuis R

Si R =0 , on remplace cette page

Si R=1, on place R à 0 et on consulte la page suivante

-Quand une page est utilisé on R à 1