Gestion de la mémoire

Catherine Faron Zucker, UNSA, EPU SI1
faron@polytech.unice.fr

Plan

- Préliminaires
- Mémoire réelle
  - Multiprogrammation
  - Partitions
  - Swap
- Mémoire virtuelle
  - Pagination
  - Segmentation

I. Introduction

Gestion de la mémoire

Introduction

- Gestionnaire de mémoire
  - Partie de l'OS dédiée à la gestion de la mémoire
  - Fonctions :
    - Savoir quelles parties de la mémoire sont utilisées et quelles sont libres
    - Allouer de la mémoire à un processus et la désallouer lorsque le processus a terminé
    - Swapper entre mémoire centrale et mémoire disque lorsque la mémoire centrale ne suffit pas pour tous les processus

Préliminaire

- Mémoire centrale et mémoire de masse
  - Primary storage, main memory
    - Stockage des données activement utilisées
    - D'accès beaucoup plus rapide que la mémoire de masse
    - Composants électroniques : RAM et ROM
  - Secondary storage
    - Stockage des données non activement utilisées
    - D'accès beaucoup moins rapide que la mémoire centrale
    - De capacité beaucoup plus grande que la mémoire centrale
    - CD, DVD, Floppy disk, Hard disk, Magnetic tape, Flash memory …

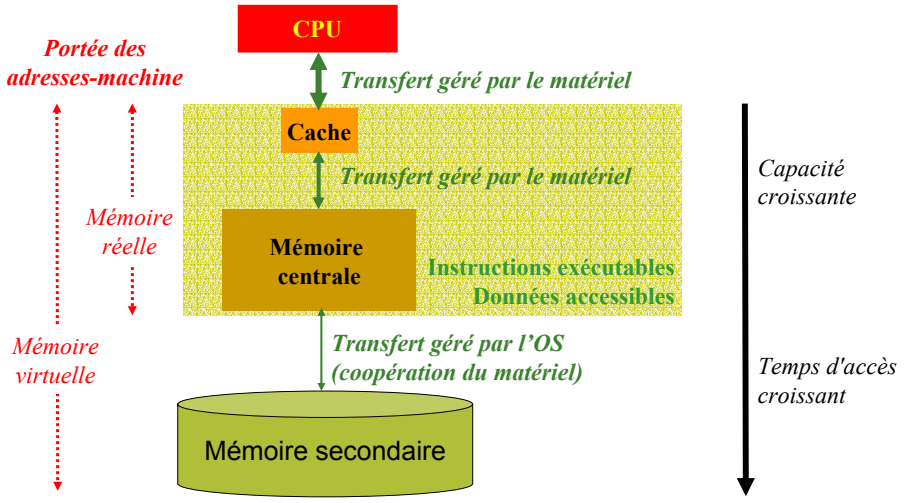
Préliminaire

- RAM: Random Access Memory
  - Mémoire vive
    - Contient les données en cours de traitement par l'unité centrale
  - Mémoire volatile
    - Les données sont perdues hors alimentation électrique
  - Système de stockage à accès aléatoire (random)
    - Par opposition à un accès séquentiel (mémoire de masse)
  - Read write memory
    - Par opposition à ROM

Préliminaire

- ROM: Read-Only Memory
  - Mémoire morte
  - Mémoire non volatile
    - Les données ne sont pas perdues lorsque l'alimentation électrique est coupée
  - Stockage des informations vitales d'un ordinateur:
    - Software lié au hardware: BIOS, instructions de démarrage
    - De plus en plus, le disque est préféré au ROM pour stocker ce type de software: l'accès est moins rapide mais la mise à jour plus aisée

Préliminaire



II. Mémoire réelle

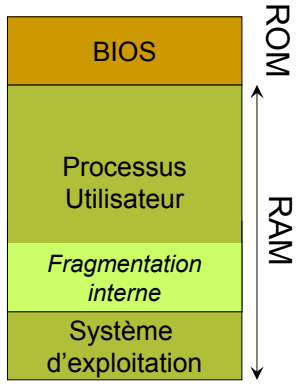
Gestion de la mémoire

Monoprogrammation

- Gestion de la mémoire la plus simple possible
  - Jusqu'en 1960
  - IBM PC

    Monotâche
    Mono-utilisateur

  - La RAM est divisée entre l'OS et un processus utilisateur unique



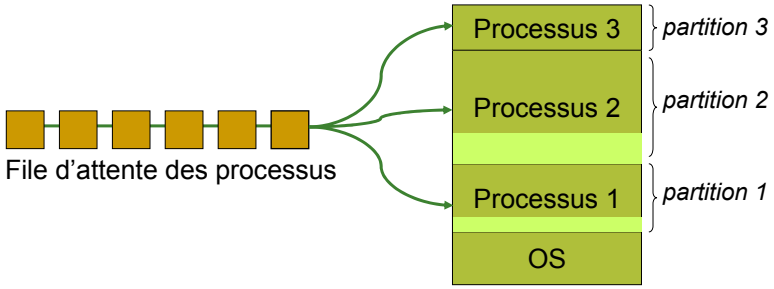
Multiprogrammation

- Multiprogrammation
  - Programmer plus facilement une application en la découpant en plusieurs processus
  - Les gros ordinateurs offrent des services interactifs à différents utilisateurs en même temps
  - La plupart des processus passent une bonne partie de leur temps en attente d'I/O sur disque
  - La CPU est davantage utilisée, qui est sinon oisive

Multiprogrammation

- Multiprogrammation avec partitions fixes (1)
  - Le nombre et les tailles des partitions sont fixés au démarrage du système

    IBM OS/360 ou OS/MFT: Multiprogramming with a Fixed number of Tasks



Multiprogrammation

- Multiprogrammation avec partitions fixes (2)
  - Algorithme de sélection de processus
    - Choix du processus à charger dans une partition libre en fonction de la taille
  - Relocation
    - Transformer les adresses absolues en adresses relatives à la partition dans laquelle un processus s'exécute
      - Si la première instruction est un appel à une procédure à l'adresse 100 dans le fichier binaire, et que la partition commence à 200K, il faut aller chercher la procédure à l'adresse 200K+100
  - Protection
    - Eviter qu'un processus écrive sur la partition d'un autre

Multiprogrammation

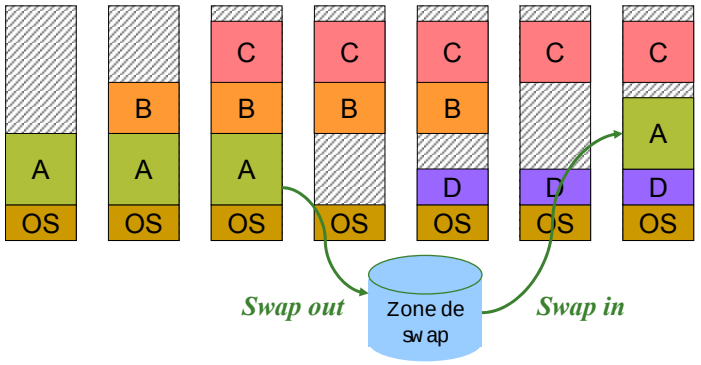
- Multiprogrammation avec partitions fixes (3)
  - Relocation
    - Modifier les adresses dans les instructions au moment où le programme est chargé en mémoire (OS/MFT)
    - Traduire une instruction (sans la modifier) à l'aide d'un registre de base contenant l'adresse du début de la partition (IBM PC)
  - Protection
    - Diviser la mémoire centrale en blocks et assigner à chacun des codes de protection (sur 4 bits) (OS /MFT)
    - A l'aide d'un registre de limite, vérifier que les adresses dans les instructions d'un programme n'accèdent pas à une autre partition

Batch processing versus Timesharing

- Batch processing : Traitement par lot
  - Exécution séquentielle d'un ensemble de programmes sur un ordinateur
  - Mutiprogrammation avec partitions fixes est *LA* solution
- Timesharing : Temps partagé
  - Il y a à chaque instant davantage de processus utilisateurs que de mémoire suffisante pour les traiter
  - Swapping: déplacement de processus de la mémoire centrale au disque (mise en attente) et inversement
  - Partitions de tailles variables pour optimiser l'utilisation de la mémoire centrale

Partitions de tailles variables

- Le nombre, la localisation et la taille des partitions varient dynamiquement

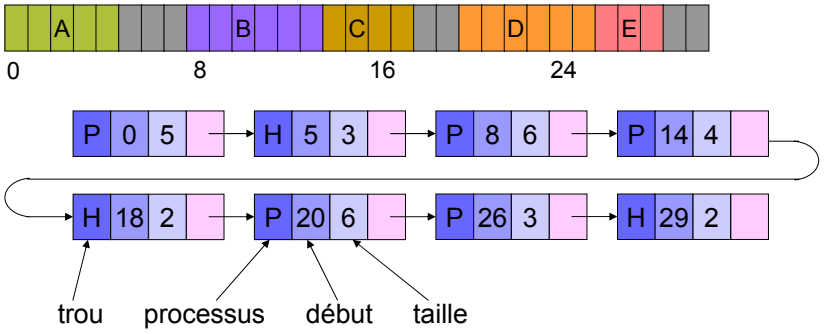


Partitions de tailles variables

- Les processus peuvent grossir
  - Allocation de partition de taille supérieure en prévision
  - En cas de dépassement
    - Allocation d'un espace libre adjacent
    - Allocation d'une nouvelle partition plus grande
    - Swap sur le disque, dans l'attente d'une partition de taille suffisante
    - Mort du processus si le disque est plein

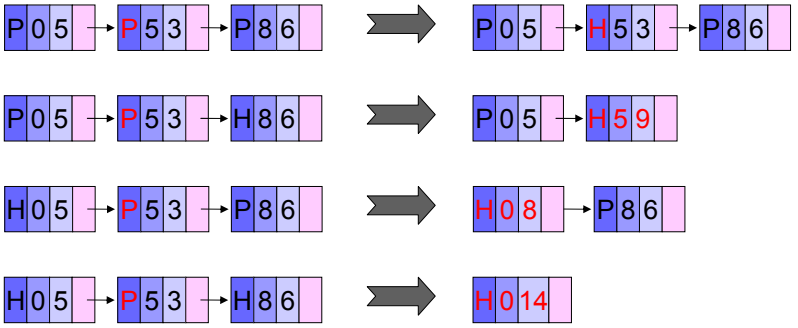
Gestion de mémoire avec liste chaînée

- Liste chaînée des segments de mémoire alloués et libres



Gestion de mémoire avec liste chaînée

- Si la liste est triée par adresses croissantes, sa mise à jour lorsqu'un processus termine ou est swappé est simple :



Gestion de mémoire avec liste chaînée

- Algorithmes d'allocation de mémoire à un processus
  - « First fit » on cherche le premier espace libre de taille suffisante
  - « Best fit » on cherche l'espace libre de taille la plus proche de celle nécessaire au processus
    - Pas mieux car générateur de fragmentation externe
      Occurrence de petits espaces libres non utilisés entre processus
  - « Worst fit » on cherche chaque fois le plus grand espace libre
    - Pas mieux
  - Optimisation listant séparément processus et trous
    - Le tri de la liste des espaces libres selon leurs tailles permet d'optimiser la recherche « First fit » et « Best fit »
    - La contrepartie est le coût de la mise à jour des listes lors de la désallocation

Gestion de l'espace disque

- Allocation d'espace disque pour le swap
- Selon les systèmes,
  - Un espace disque de swap est réservé à la création d'un processus, qui est libéré lorsqu'il termine
  - Ou bien un nouvel espace est attribué au processus chaque fois qu'un swap est nécessaire
- Les algorithmes d'allocation d'espace disque sont les mêmes que ceux d'allocation de la mémoire

III. Mémoire virtuelle

Gestion de la mémoire

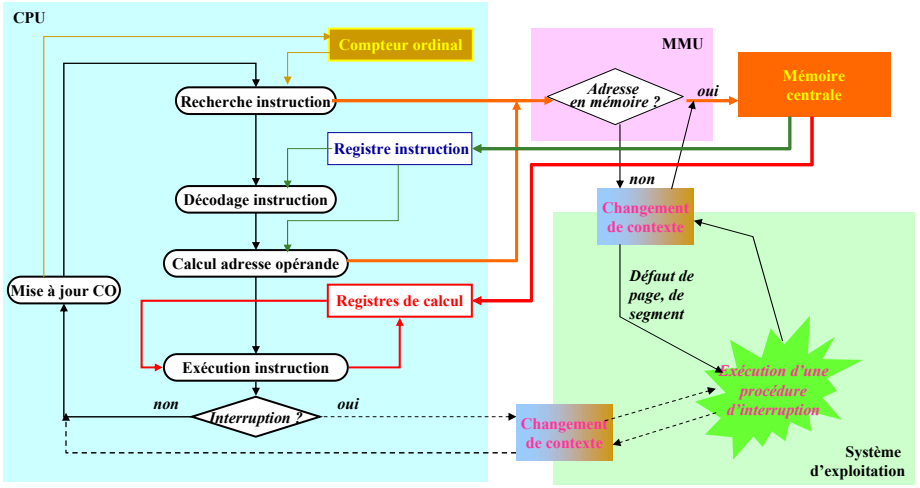
Mémoire virtuelle

- Utilisation d'une mémoire de masse pour pallier un manque de mémoire centrale pour exécuter des programmes de grande taille
  - L'OS ne conserve en mémoire que les parties utiles et swape le reste sur le disque
  - La mémoire physique sert de cache à la mémoire virtuelle
    - Un programme de 1M peut être exécuté sur une machine de 256K en choisissant à chaque instant les bons 256K à garder en mémoire
- Se combine très bien avec la multiprogrammation

Du virtuel au physique

- Les adresses générées par les programmes sont des adresses virtuelles
  - Sur des machines sans mémoire virtuelle, elles correspondent aux adresses physiques
  - Sur des machines avec mémoire virtuelle, la MMU *traduit* les adresses virtuelles en adresses physiques
- Si l'information adressée par l'instruction courante n'est pas en mémoire physique
  - l'instruction courante et le processus qui la contient sont suspendus
  - le système d'exploitation prend la main et fait monter l'information en mémoire physique
  - lorsque le processus est ordonnancé à nouveau, l'instruction suspendue est reprise à l'accès mémoire

Du virtuel au physique



Segmentation et pagination

- Segmentation
  - Espace d'adressage décomposé en segments de tailles variables
  - Les segments peuvent avoir une signification fonctionnelle (procédures, données d'un tableau…)
  - Le mécanisme de traduction doit vérifier la taille
  - L'allocation en mémoire physique est délicate
- Pagination
  - Espace d'adressage décomposé en pages de taille fixe
  - Le découpage est aveugle (pas de sens fonctionnel)
  - Les mécanismes de traduction et d'allocation sont simples
  - Le remplacement de pages est délicat

Pagination

- L'espace d'adressage virtuel est divisé en pages
- Les adresses de mémoire physique sont divisées en frames de page de même taille que les pages
- Pages et frames sont mis en correspondance dans une table de pages

Défaut de page

- Lorsqu'un processus veut utiliser une page qui n'est pas associée à un frame
- Le MMU fait appel à l'OS pour
  - choisir un frame peu utilisé,
  - copier son contenu sur le disque et
  - mettre à jour la table de correspondance pages/frames
    - en dissociant le frame choisi de la page à laquelle il était précédemment associé et
    - en l'associant à la page qui sera utilisée par le processus responsable du défaut de page

Table de pages

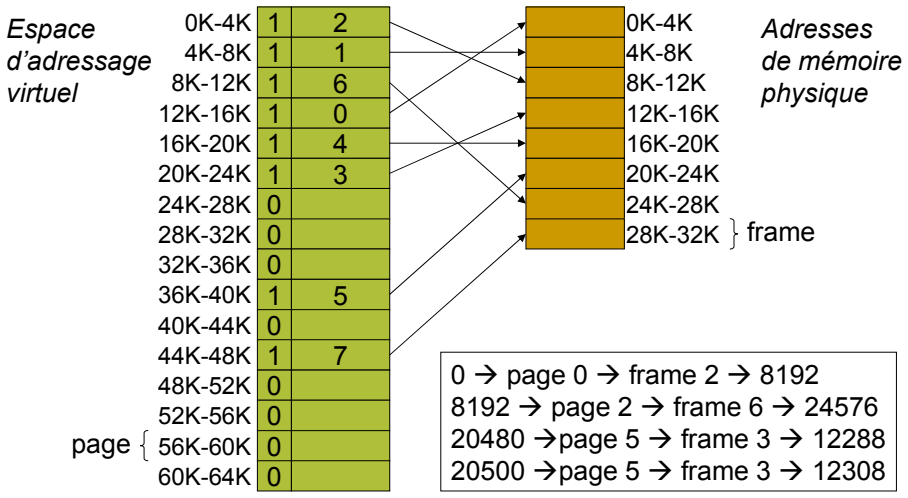


Table de pages

- Une adresse virtuelle sur n bits se compose
  - d'un numéro de pages sur x bits et
  - d'un déplacement (offset) par rapport au début de la page sur n x bits
- Dans la table de pages
  - Les numéros de page servent d'index
  - Les numéros de frames sont les valeurs
- L'adresse physique correspondant à une adresse virtuelle est composée en juxtaposant le numéro de frame correspondant au numéro de page dans la table des pages et le offset inchangé

Table de pages

- Exemple
  - Adresses virtuelles sur 16 bits ($2^{16} = 64*2^{10} = 64K$)
  - Codage des numéros de page sur 4 bits
    - La table stocke $2^4 = 16$ pages
    - Restent 12 bits pour le codage de l'offset
    - La longueur d'une page est de $2^{12} = 4*2^{10} = 4K$
  - Si la mémoire physique est 2 fois moindre que la mémoire virtuelle,
    - Adresses physiques sur 15 bits
    - Codage des numéros de frame sur 3 bits
    - La table stocke $2^3 = 8$ pages

Table de pages

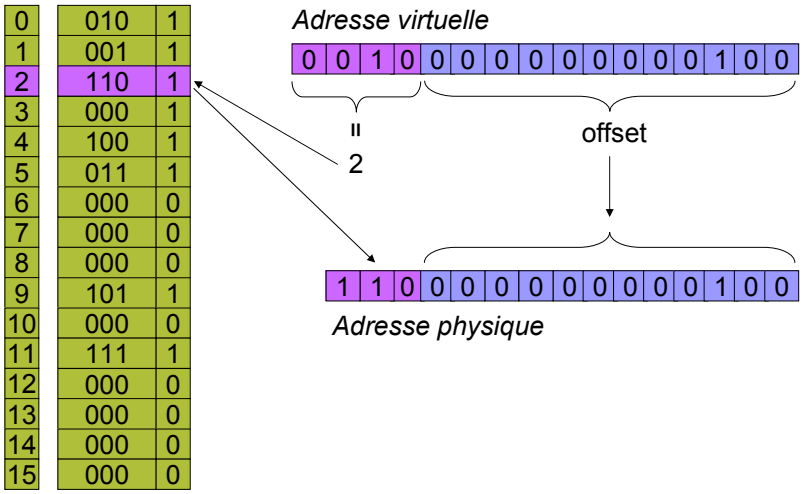
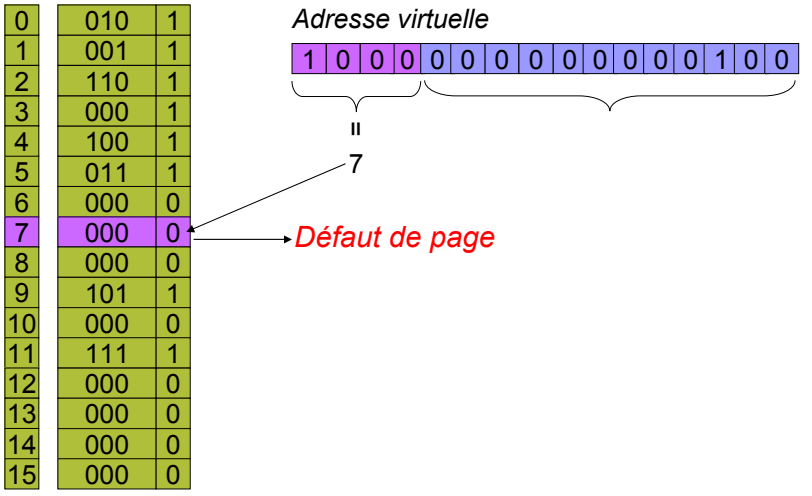


Table de pages



Taille de page

- Une grande taille de page est génératrice de fragmentation interne
- Une petite taille de page nécessite une table de pages de grande taille
- La taille de page optimale est celle qui minimise la somme de la taille de la table de page et de la mémoire perdue par fragmentation interne, calculées en fonction de la taille moyenne des processus
- En pratique entre 512 bytes et 8K

Mémoire associative

- Le temps d'accès aux adresses physiques dans une grande table de pages est trop long.
- Les entrées les plus souvent utilisées sont conservées dans une mémoire associative (mémoire cache) aussi appelée Translation Lookaside Buffer (TLB)
  - Chaque adresse virtuelle issue du processeur est d'abord cherchée dans le TLB
  - S'il y a correspondance, l'entrée dans le TLB est utilisée
  - Sinon une interruption est déclenchée, une recherche est faite dans la table des pages et le TLB devra être mis à jour avec l'entrée de la table stockée en mémoire avant que l'instruction fautive soit redémarrée
- Tous les microprocesseurs actuels possèdent un TLB

Remplacement de page

- Lors d'un défaut de page, l'OS choisit une page à retirer de la table en mémoire pour la remplacer par celle demandée
- Si la page à retirer a été modifiée pendant qu'elle était en mémoire, elle est réécrite sur le disque pour tenir celui-ci à jour
- Il s'agit donc de trouver un algorithme de remplacement de pages qui minimise les swaps et les mises à jour du disque

Trashing ou Effondrement

- Saturation du système quand des défauts de page sont générés trop souvent (forte diminution du rendement de l'ordinateur)
- Le phénomène de trashing apparaît
  - Quand la mémoire cache est trop petite
  - Quand le taux de multiprogrammation est trop élevé
- Il est possible de diminuer les effets de ce comportement
  - En rajoutant des ressources matérielles (ajouter de la mémoire)
  - En diminuant le taux de multiprogrammation
  - En modifiant la priorité des processus

Remplacement de page idéal

- Algorithme idéal
  Remplacer la page en mémoire qui ne sera pas utilisée pendant la durée la plus longue

  … Suppose de connaître l'avenir !

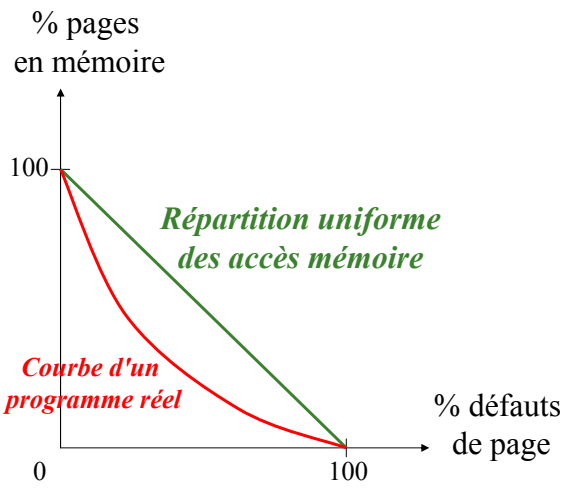
- Approximations liées au principe de localité

Principe de localité

- Le comportement d'un programme n'est pas chaotique
  - Programmation structurée (boucles)
  - Modularité (fonctions)
  - Structures de données (tableaux)

  ➔ Un processus passe beaucoup de temps à s'exécuter au sein de quelques régions
- C'est le *principe de localité*
  - Un processus accède fréquemment à un petit ensemble de pages, qui évolue lentement avec le temps

Principe de localité



Remplacement de page « NRU »

« Not Recently Used » (algorithme le plus utilisé)

- 2 bits d'état sont associés à chaque page :
  - R pour référencée (en écriture ou en lecture)
  - M pour modifiée (écrite).
  - Ces bits sont mis à jour à chaque accès à la mémoire.
- Quand un processus est lancé, les bits R et M de toutes les pages sont mis à 0 par l'OS
- Périodiquement, tous les bits R sont remis à 0 pour pouvoir distinguer les pages qui ont été référencées *récemment*
- Lorsqu'un défaut de page survient, une page est enlevée aléatoirement parmi celles dont le RM est minimum

Remplacement de page NRU

- RM
  - 00 : page non référencée, non modifiée
  - 01 : page non référencée, modifiée
  - 10 : page référencée, non modifiée
  - 11 : page référencée, modifiée
- Stratégie
  - Mieux vaut ne pas enlever une page récemment référencée car elle peut l'être à nouveau prochainement
  - Parmi les pages non référencées, mieux vaut swapper une page non modifiée pour ne pas devoir mettre à jour le disque

Remplacement de page FIFO

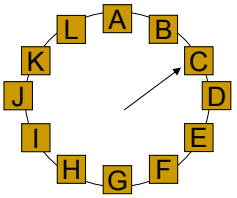
- L'OS maintient une file des pages en mémoire:
  - la page la plus ancienne en tête
  - la page la plus récente en queue
- Lorsqu'un défaut de page survient,
  - La page en tête de file est retirée et
  - Une nouvelle page est ajoutée en queue de file
- Problème de « l'anomalie de Belady »
  - Il est possible que plus de défauts de page soient générés alors qu'on augmente le nombre de frames (c'est-à-dire rajoute de la mémoire)

Remplacement de page FIFO

- Algorithme de « la seconde chance »
  - Variante de l'algorithme « FIFO »
  - Une page ancienne en mémoire peut cependant avoir été référencée récemment et est donc susceptible de l'être à nouveau bientôt
- Inspecter le bit d'état R de la page en tête de file
  - Si R=0, remplacer la page
  - Si R=1, remettre R à 0 et mettre la page en queue de file comme si elle venait juste d'arriver en mémoire

Remplacement de page FIFO

- Algorithme de « l'horloge »
  - Variante de « la seconde chance »
  - Liste circulaire des pages et
  - Pointeur sur la page la plus ancienne



- Cette implémentation évite de modifier la liste pour changer le rang d'une page

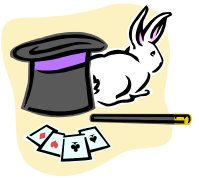
Remplacement de page LRU

« Least Recently Used »

- Lors d'un défaut de page, remplacer la page utilisée la moins récemment
- L'OS maintient une liste des pages en mémoire,
  - Celle utilisée le plus récemment en tête
  - Celle utilisée le moins récemment en queue
- La mise à jour de cette liste doit donc avoir lieu à *chaque* accès à la mémoire
  - Trop coûteux

Remplacement de page aléatoire

- Tirage aléatoire de la page à remplacer



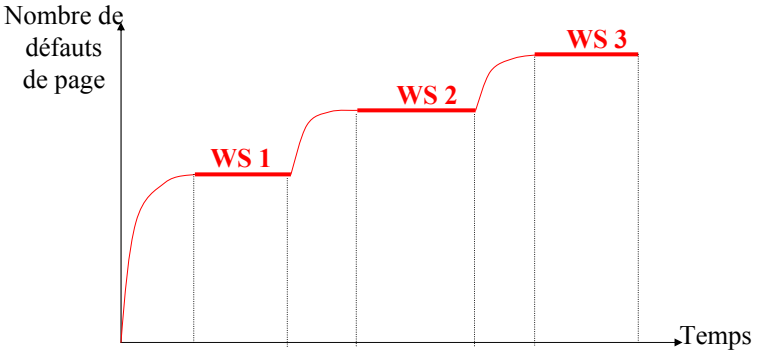
Remplacement de page

- Gestion globale de l'allocation de mémoire
  - Pagination à la demande
  - Algorithmes de remplacement de page NUR, FIFO, LRU
- Gestion locale, i.e. par processus
  - Prépagination: prédiction des pages utiles à un processus
  - Modèle du « Working Set »
    - Ensemble des pages qu'utilise un processus

    Chaque fois qu'un processus est envoyé en attente sur le disque, son working set est mémorisé et rechargé avant que le processus ne revienne en mémoire

Remplacement de page « working set »

- Modèle du « working set »



Remplacement de page « working set »

- Taille du working set Δ
  - Nombre de références aux pages accédées par chaque processus durant un certain laps de temps
  - Fixée expérimentalement : trop petite elle ne couvre pas l'espace de travail *effectif* du processus, trop grande elle inclut des pages inutiles
  - C'est à l'OS d'optimiser la valeur de Δ de sorte que le taux de multiprogrammation et l'utilisation de la CPU soient maximisés tout en évitant le trashing
  - Remarque: Si la somme des Δ de tous les processus est supérieure au nombre de frames disponibles, il y a forcément trashing

Allocation locale versus globale

- En général, une politique globale fonctionne mieux
  - En particulier pour des processus dont le Δ est susceptible de varier au cours de leurs durées de vie
  - Avec une politique locale, si Δ augmente, des défauts de page surviennent et si Δ diminue la mémoire est sous-utilisée
- Une politique locale efficace doit pouvoir maintenir le nombre de pages allouées à chaque processus
  - En deçà d'un seuil correspondant à une sous-utilisation de la mémoire
  - Au-dessus d'un seuil correspondant au trashing
    - Quand il devient impossible de maintenir les nombres de pages allouées au-dessus des seuil de trashing, des processus sont swappés sur le disque

Segmentation

- Découpage de la mémoire virtuelle en espaces d'adressage complètement indépendants
- Chaque segment est dédié à une utilisation particulière
  - le code source d'un programme,
  - la table des symboles, la table des constantes,
  - l'arbre contenant l'analyse syntaxique du programme,
  - la pile pour les appels de procédures dans le compilateur,
  - une procédure, un tableau, une pile, etc.
- Un segment est donc une entité logique

➢ Structuration de l'espace d'un processus

Segmentation

La segmentation simplifie …

- La gestion des structures de données qui grossissent ou diminuent
  - Les segments sont de tailles différentes et leurs tailles peuvent varier pendant l'exécution d'un programme
- L'édition de liens entre procédures compilées séparément
  - Si chacune occupe un segment séparé avec pour adresse de départ 0, que l'une d'elle doive être recompilée ne change rien

    (si les procédures occupaient un même espace, l'adresse de l'une serait relative à celle de l'autre)
- Le partage de procédures ou de données entre processus
  - Notamment pour les librairies partagées

Segmentation

- Table des segments
  - Dans un segment l'adressage se fait à l'aide d'un déplacement
  - Le couple (segment, déplacement) est traduit en une adresse mémoire à partir d'une table de segments dont les entrées contiennent deux champs, limite et base: les adresses de début et fin de segment
- Segmentation et pagination
  - Une différence essentielle: la taille des pages est fixe alors que celle des segments est variable
  - Utilisation conjointe des segments et pages: segments paginés