Gestion Mémoire Paginée

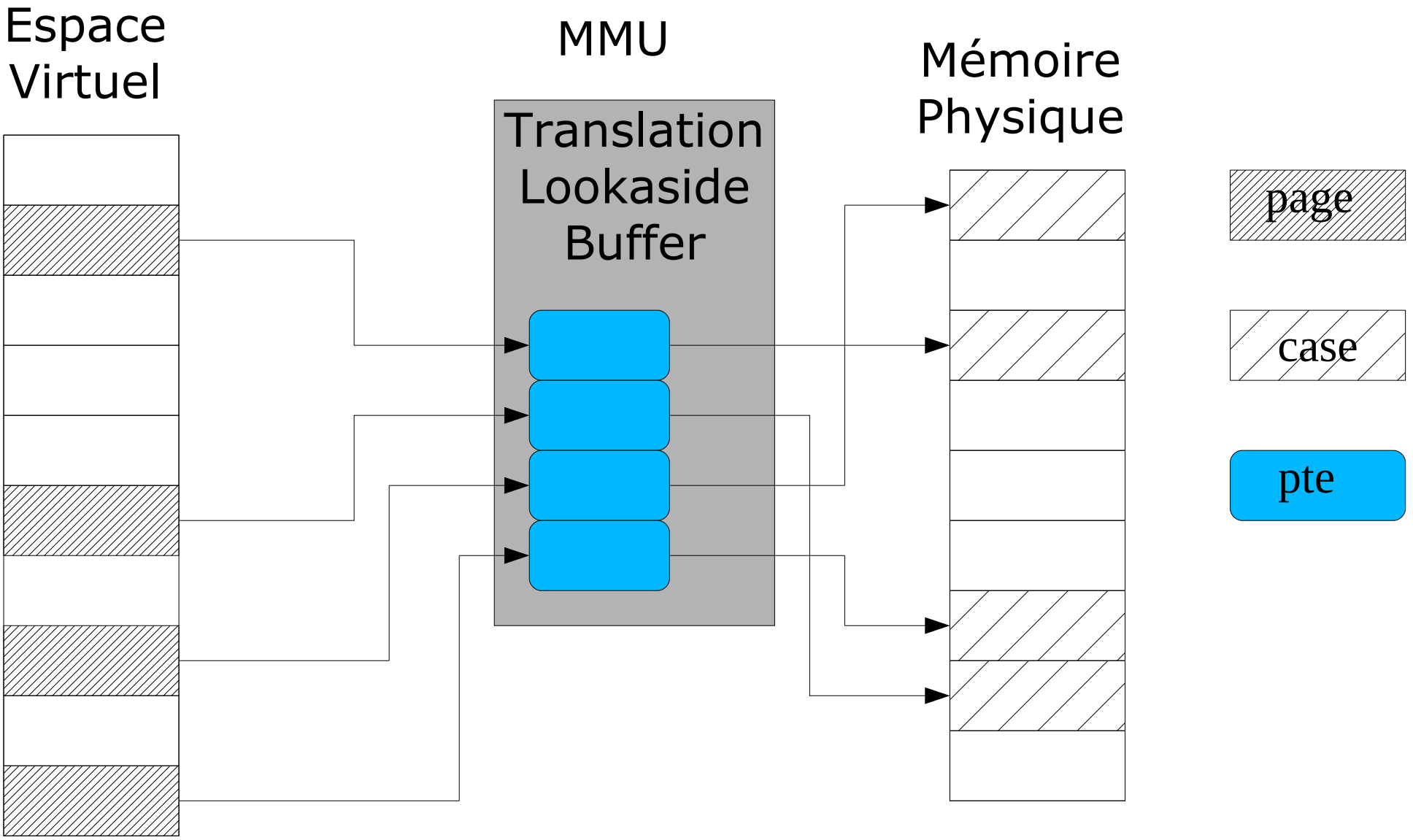
Plan

- Principe de la pagination
- Comportement des programmes
- Notion défaut de page
- Politiques de remplacement de pages

Espace Virtuel Paginé

- Mémoire centrale découpée en cases de taille fixe
- Espace virtuel divisé en pages de même taille
- Chaque page définie dans espace virtuel associée à une case de mémoire centrale
  - Opération d'association dynamique
  - 2 pages contigües ne sont pas nécessairement rangées dans des cases contigües
- Conversion adresse virtuelle => adresse physique
  - Réalisé par MMU (Memory Management Unit)

Espace Virtuel Paginé



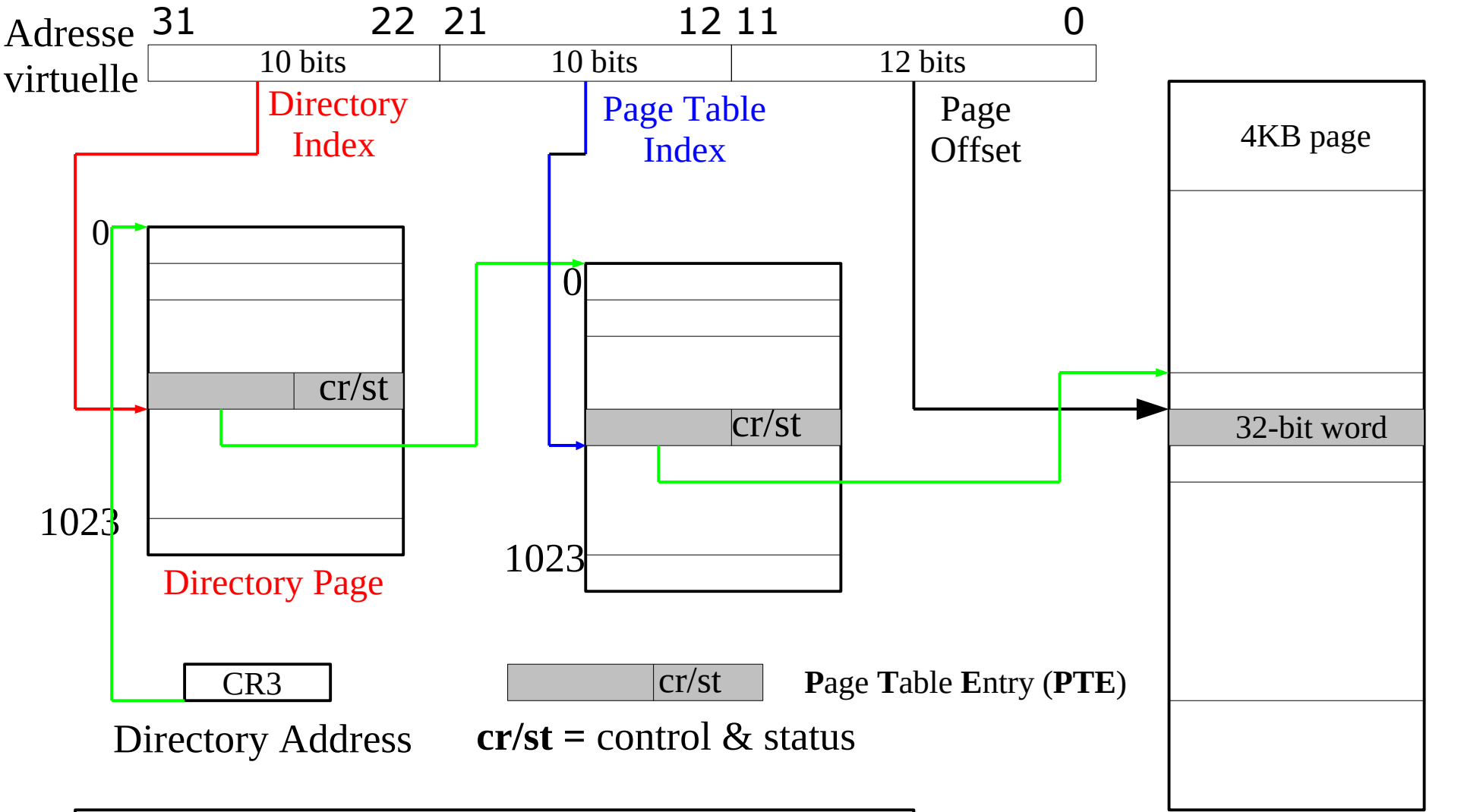
Memory Management Unit

- MMU décompose adresse virtuelle en 2 parties
  - Partie déplacement dans la page (de taille maximum égale à la taille de page)
  - Partie numéro de page interprétée par MMU pour obtenir adresse de case correspondante
  - Adresse physique = adresse case | déplacement
- Numéro de page
  - Index direct dans table de pages
  - structure "arborescente" multi niveaux
    - Intel : 2 niveaux (Page Directory + Page Tables)

Page Table Entry

- Page Table Entry (PTE) décrit une page d'un espace virtuel
- Adresse de la page physique associée, si valide
- Bits de contrôle
  - Page physique associée valide
  - Droits d'accès (lecture, écriture, exécution)
- Bits de status
  - Page accédée
  - Page modifiée

MMU Intel


Adresse virtuelle
31
22
21
12
11
0
10 bits
10 bits
12 bits
Directory Index
Page Table Index
Page Offset
4KB page
0
cr/st
1023
Directory Page
0
cr/st
1023
32-bit word
CR3
Directory Address
cr/st
Page Table Entry (PTE)
cr/st = control & status
Translation Lookaside Buffer (TLB) = cache for PTEs
Mémoire physique

Avantages & Inconvénients

- Résoud problème de fragmentation de la mémoire centrale et de l'espace de va-et-vient
  - Gestion dynamique fine et efficace
- Autres avantages
  - Partage de pages entre espaces virtuels
  - Permet de "mapper" des fichiers dans espace virtuel d'un processus
- Inconvénients
  - Mécanisme matériel complexe
  - Fragmentation interne : pages non complètement occupées

Plan

- Principe de la pagination
- Comportement des programmes
- Notion défaut de page
- Politiques de remplacement de pages

Comportement des Programmes

- Référence à une page
  - Accès à un emplacement de la page
  - Plusieurs références distinctes par instruction
    - page(s) contenant l'instruction elle-même
    - page(s) contenant les opérandes de l'instruction
- Comportement d'un programme
  - Temps virtuel dont l'écoulement est repéré par ensemble des instructions successives exécutées
  - Chaîne de références : séquence des pages référencées successivement durant son exécution

Propriétés Chaînes de Références

- Non-uniformité
  - Répartition du nombre de références à chaque page n'est pas uniforme
  - Fraction faible des pages cumule part importante du nombre total de références
- Localité
  - Répartition des références stable sur courte période
  - Séquences d'instructions d'adresses successives, parcours de boucles
  - Manipulent données « proches » (champs d'une structure, éléments successifs d'un tableau, etc...)

Phases & Transitions (1)

- Déroulement d'un programme défini par succession de *phases* séparées par *transitions*
- Phase P_i
  - Période de comportement stable et prévisible
  - Ensemble de pages S_i
  - Intervalle de temps T_i
- Transitions entre phases S_i et S_{i+1}
  - Période de comportement plus erratique durant laquelle les références sont dispersées

Phase & Transitions (2)

- Majeure partie temps virtuel total occupée par phases de [relative] longue durée
- Ensemble périodes de transition
  - fraction faible du temps virtuel total d'un programme
  - Importantes par dispersion des références
- Notion d'ensemble de travail (« *working set* »)
  - W(t, T) = pages référencées entre instants t et t – T
  - Pages de W(t, T) ont plus grande probabilité d'être référencées à l'instant t+1 si T « *bien choisie* »
  - T <= T_i quand programme est dans phase P_i

Plan

- Principe de la pagination
- Comportement des programmes
- Notion défaut de page
- Politiques de remplacement de pages

Notion de Défaut de Page

- Programmes partiellement chargés en mémoire
  - Chargement des pages « à la demande »
- Défaut de page
  - Référence à une page absente de la mémoire centrale
  - Page « fautive » doit être chargée en mémoire
  - Si pas de case disponible, **remplacement de page** : page chargée doit reprendre case allouée à une page déjà présente appellée « page victime »
- Durée de vie = intervalle temps moyen entre 2 défauts de page successifs

Observation des Défaut de Page

- Taille mémoire optimale pour un programme
- Lorsqu'on diminue la taille mémoire :
  - Nombre de défauts de page augmente lentement, puis explose au dessous d'une taille minimum
  - La durée de vie du programme diminue, puis chute rapidement en dessous de la taille minimum
- Phénomène d'écroulement (« trashing »)
  - Forte croissance du taux de remplacement de pages
  - Chute du taux d'utilisation du processeur

Traitement Défaut de Page

- Si PTE(page) présente dans TLB, MMU
  - Translate adresse virtuelle -> adresse physique
  - Met à jour bit d'accès et bit de modification (si écriture)
- Sinon, MMU provoque déroutement de défaut de page :

  a) Déterminer page « *fautive* » (type, localisation)

  b) Trouver case libre en mémoire centrale
  Si nécessaire, libérer une case occupée après avoir sauvegardé son contenu si modifié depuis chargement

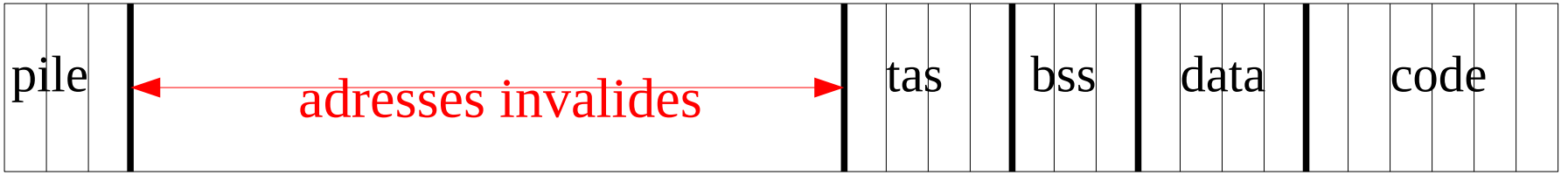
  c) Charger contenu de la page fautive dans la case précédemment allouée

  d) Mettre à jour PTE de la page

Localisation d'une page

- Ensemble des régions de l'espace d'adressage de chaque processus géré par OS
- Région associée à objet en mémoire secondaire
  - Code -> fichier binaire du programme exécuté
  - Données initialisées (avant modification) -> **idem**
  - Piles, données globales, tas -> espace de swap
- Espace de swap : partie d'une mémoire secondaire (disque)
  - Structuré en blocs de la taille d'une page
  - Géré comme extension mémoire centrale
  - Utilisé pour stockage temporaire de pages modifiées

Localisation page fautive



- Chercher région contenant la page fautive
  - Si pas trouvée, adresse invalide
- Sinon, déterminer localisation de la page
  - Code, données
    - Position relative page dans région -> position dans fichier binaire
  - Autres régions :
    - Si 1er accès : page sans contenu prédéfini
    - Sinon, page -> numéro de bloc dans espace de swap

Politiques allocation de page

- Partition fixe
  - Nombre fixe de cases attribué à chaque processus
  - Peut-être différent selon le nombre de processus
- Partition variable
  - Nombre de cases attribuées à chaque processus varie avec le temps
  - Seul le nombre de cases est significatif, par l'identité des cases allouées

Plan

- Principe de la pagination
- Comportement des programmes
- Notion défaut de page
- Politiques de remplacement de pages

Politiques remplacement de page

- Remplacement local
  - Page victime choisie dans pages déjà allouées au processus « fautif »
- Remplacement global
  - Page victime choisie parmi ensemble des pages chargées en mémoire centrale
- Choix de la page victime selon critéres de remplacement indépendants local/global

Critères de remplacement (1)

- Page « propre »
  - Non modifiée depuis son chargement
  - Possèdant une copie conforme en mémoire secondaire
  - => pas besoin d'être sauvegardée
- Page « sale »
  - page modifiée depuis son chargement
  - Doit être sauvegardée avant remplacement
- Utilise bit modification de PTE entretenu automatiquement par MMU

Critères de remplacement (2)

- Page partagées par plusieurs processus
  - Choisir page utilisée par un seul processus avant page partagée par plusieurs processus
  - Par généralisation, choisir page la moins partagée
- Pages « verrouillées » en mémoire centrale
  - Protégées contre remplacement
  - Pages utilisées comme tampons entrées/sorties (protection temporaire)
  - Pages du système (protection permanente)

Remplacement Optimal (MIN)

- Choisir victime parmi pages plus jamais référencées ultérieurement
- Par généralisation, choisir page qui fera l'objet de la référence la plus tardive
- Irréalisable car suppose connaissance de la suite de références à l'exécution
- Utilisé pour base de comparaison pour évaluer les autres algorithmes

Remplacement par ordre chronologique (FIFO)

- Choisir comme victime la page la plus anciennement chargée en mémoire centrale
- Très simple à réaliser, en entretenant une file des cases par ordre de chargement des pages
- Moment de la prochaine référence à une page est en général indépendant du moment (passé) du chargement de la page
- Par généralisation, probabilités de référence à des pages chargées convergent avec leur temps de présence

Remplacement LRU (1)

- Ordre chronologique d'utilisation (*Last Recently Used*)
- Choisir comme victime la page ayant fait l'objet de la référence la plus ancienne
- Basé sur propriété de localité des programmes : pages récemment utilisées ont une probabilité plus élevée d'être utilisée dans un futur proche
- Implique d'ordonner les cases par « dates » de référence aux pages qu'elles contiennent

Remplacement LRU (2)

- Datation d'une page par compteur de référence
- Compteur intégré dans PTE et incrémenté par MMU à chaque référence
  - Remis à zéro après avoir atteint valeur maximum
  - => solution approchée du LRU
- Mécanisme trop coûteux au niveau matériel par rapport aux avantages fournis

Remplacement LRU (3)

- Solution « de la seconde chance » ou FINUFO (« First In Not Used, First Out »)
- Approximation du LRU
- Basé sur bit de référence intégré dans PTE et mis à 1 par MMU à chaque référence
- Cases ordonnées dans une liste circulaire
- Tête de liste (*last_loaded)* = case de la dernière page chargée

Remplacement LRU (4)

```
case = suivant(last_loaded); /* dans liste */
tantque ( bit_référence(case) != 0 ) faire
   bit_référence(case) = 0;
   case = suivant(case); /* dans liste */
fin
victime = case; last_loaded = victime;
```

- Cherche première case non référencée
- Donne seconde chance aux pages référencées parcourues de ne pas être choisies comme victime
- Appellé aussi algorithme de l'horloge (« clock »)

Évaluation avec partitions fixes

- Évaluation des méthodes de remplacement basé sur nombre total de défauts de page
- Classement par performances décroissantes
  - MIN, LRU, FINUFO, FIFO
- Influence de taille mémoire largement supérieure à celle de méthode de remplacement
- => augmenter nombre de cases attribuées à un programme pour augmenter ses performances

Remplacement avec partitions variables

- Préférable d'adapter la mémoire allouée à un programme en fonction de son comportement
- => augmenter nombre de cases attribuées à un programme pour augmenter ses performances
- Allouer nombre de cases minimum compatible avec taux acceptable de défauts de pages
  - Méthode basée sur « working set »
  - Méthode basée sur mesure du taux de défauts de page (« *Page Fault Frequency* », ou PFF)

Méthode du Working Set

- OS gère working set par processus
- Page victime choisie
  - Parmi page de working set non présents en mémoire
  - Puis dans working set de processus les moins prioritaires
- Mémoire allouée à un processus si assez de cases libres pour son working set
- Complexe car impose de déterminer et maintenir working set de chaque processus

Méthode du PFF

- Mesure taux de défauts de page des processus
- Si taux dépasse un seuil supérieur, OS attribue case supplémentaire au processus
- Inversement, si taux tombe en dessous d'un seuil inférieur, OS retire une case
- Simple à réaliser avec bit de référence par page (pour déterminer case retirée)