Gestion de la mémoire

Stéphanie Moreaud

Département d'informatique
IUT Bordeaux 1

Plan

1. Introduction
2. Partage de la mémoire
3. Adresses logiques
4. Gestion de l'allocation contiguë
5. Swapping
6. Segmentation
7. Espace mémoire paginé
8. Mémoire virtuelle paginée
9. Algorithmes de remplacement de page
10. Mémoire virtuelle segmentée paginée

Bibliographie

Tanenbaum (A.), *Systèmes d'exploitation*. Pearson Education.

Billaud (M.), *Cours de ASR2-Système*.

`http://www.labri.fr/perso/billaud/`.

Remerciements à Michel Billaud.

Plan

1 Introduction
- Systèmes et mémoire

2 Partage de la mémoire

3 Adresses logiques

4 Gestion de l'allocation contiguë

5 Swapping

6 Segmentation

7 Espace mémoire paginé

8 Mémoire virtuelle paginée

9 Algorithmes de remplacement de page

10 Mémoire virtuelle segmentée paginée

Ressource mémoire

Mémoire : ressource essentielle au fonctionnement d'un ordinateur qui permet le stockage des informations.

L'évolution des ordinateurs s'accompagne de l'augmentation de la puissance de calcul et de la mémoire disponible...
... mais plus il y a de ressources disponibles, plus les programmes sont gourmands.

- les besoins en mémoire augmentent aussi vite que la taille de celle-ci...

➔ la mémoire doit être gérée avec soin.

Objectif du cours : comprendre comment les systèmes d'exploitation gèrent la mémoire.

Ressource mémoire

Les programmeurs voudraient une mémoire :

- infiniment grande
- extrêmement rapide
- non volatile
- bon marché

➔ de telles mémoires n'existent pas...

On utilise des compromis

- plusieurs types de mémoire, différentes caractéristiques
  - très rapide / moins rapide
  - volatile / non volatile
  - chère / bon marché
- organisation hiérarchique

Hiérarchie mémoire

L'information est stockée dans plusieurs niveaux de mémoire

- Registres
  - interne au CPU, capacité $<$ 1Ko, gérés par le programme
- Mémoire cache
  - très rapide, très chère, capacité de quelques Mo.
- Mémoire principale (*Random Access Memory*)
  - assez rapide, chère, capacité allant jusqu'à quelques dizaines de Go pour les gros calculateurs
- Disque dur
  - lent, peu chère, grande capacité (centaines de Go), non volatile
- Bande magnétique
  - très grande capacité, très peu chère, accès très lent, durée de vie longue.
  - utilisé pour l'archivage, concurrencé par les disques

Systèmes monotâches

Système monotâche ➔ 1 seul programme en mémoire

Utilisation de cartes perforées

- lecteur de carte remplissait la mémoire
- machine dans une salle entière
- fastidieux

Apparition des bandes magnétiques

- programme chargé en mémoire exécute les calculs et fait des opérations d'E/S

➔ chargement de données à l'exécution

- lecture sur la bande magnétique très lente

Systèmes multitâches

Système multitâche ➔ plusieurs programmes chargés en mémoire

La cohabitation dans l'espace mémoire introduit de nouvelles problématiques

- Comment partager la mémoire entre les différents programmes ? Éviter le gaspillage ?
- Comment protéger les programmes les uns des autres ?
- Comment assurer le fonctionnement des programmes sans compliquer la programmation ?

Plan

1 Introduction

2 Partage de la mémoire
- Allocation contiguë

3 Adresses logiques

4 Gestion de l'allocation contiguë

5 Swapping

6 Segmentation

7 Espace mémoire paginé

8 Mémoire virtuelle paginée

9 Algorithmes de remplacement de page

10 Mémoire virtuelle segmentée paginée

Partage de la mémoire

Où mettre les programmes ?

Idée : partitionner la mémoire

- mémoire physique divisée statiquement en partitions fixes
- programme chargé dans dans une zone suffisamment grande
- facile à implémenter, peu coûteux.
- gaspillage de mémoire, nombre de programmes limités

Partage de la mémoire

Où mettre les programmes ?

Idée : partitionner la mémoire

- mémoire physique divisée statiquement en partitions fixes
- programme chargé dans dans une zone suffisamment grande
- facile à implémenter, peu coûteux.
- gaspillage de mémoire, nombre de programmes limités

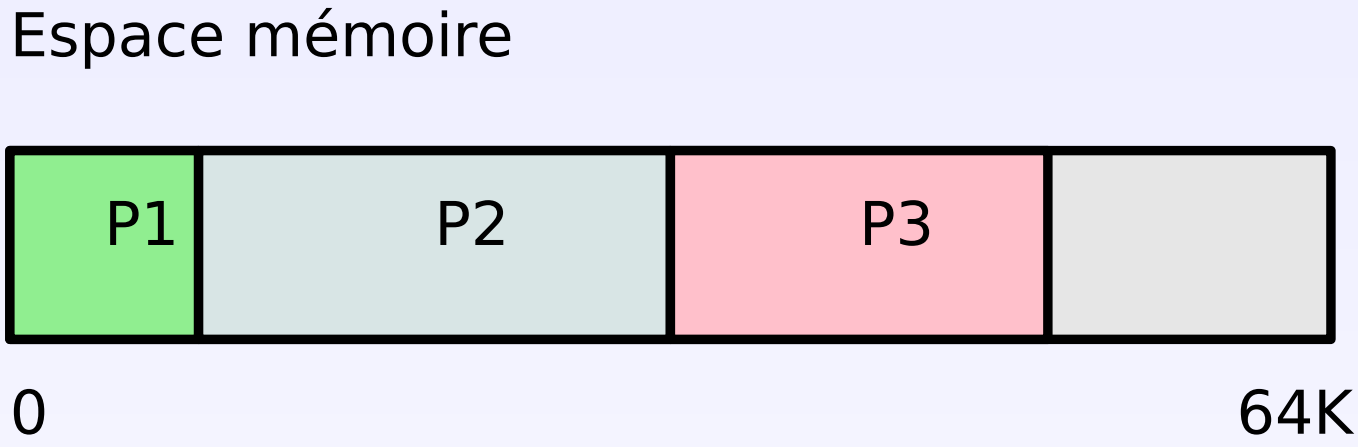
Autre idée : les uns à la suite des autres

- ➔ allocation contiguë
- partition de taille dynamique
- création à la volée en fonction de la mémoire requise, taille optimale

Allocation contiguë

Pour chaque processus, un espace contigu en mémoire physique

➔ une plage d'adresses



L'allocation contiguë

Problèmes :

- Emplacement du code connu au chargement

➔ Le code doit être indépendant de sa position physique.

- Les espaces alloués sont
  - de tailles diverses
  - libérés dans un ordre imprévisible

➔ Comment gérer les allocations/restitutions des espaces ?

Plan

1 Introduction

2 Partage de la mémoire

3 Adresses logiques
- Indépendance code/position physique
- Conversion adresse logique/adresse Physique
- Protection inter-processus
- Bilan

6 Segmentation

7 Espace mémoire paginé

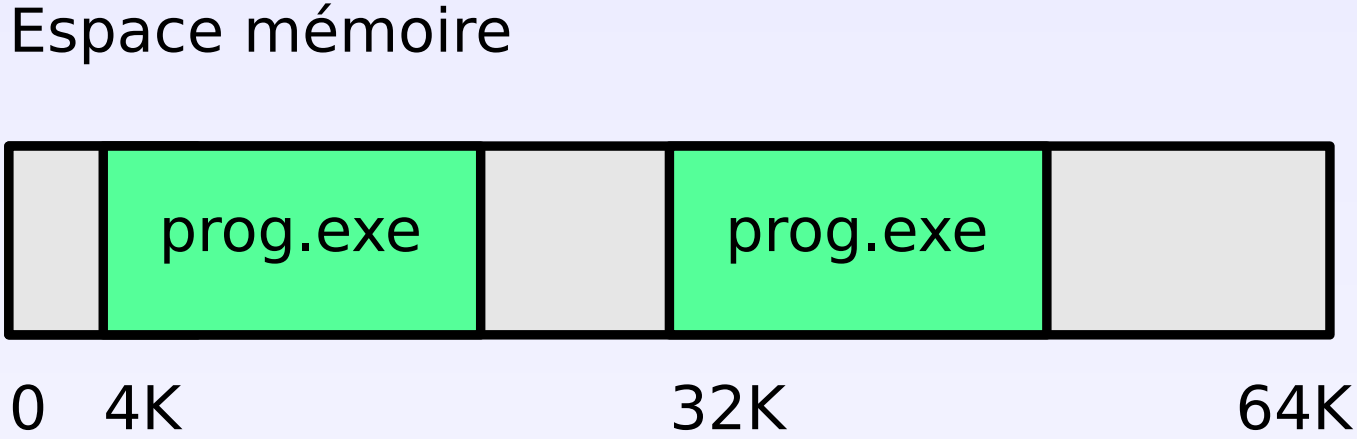
8 Mémoire virtuelle paginée

9 Algorithmes de remplacement de page

10 Mémoire virtuelle segmentée paginée

Indépendance code /position

Pb : Un exécutable peut être chargé à des emplacements différents (éventuellement simultanément)



Il doit fonctionner sans modification.

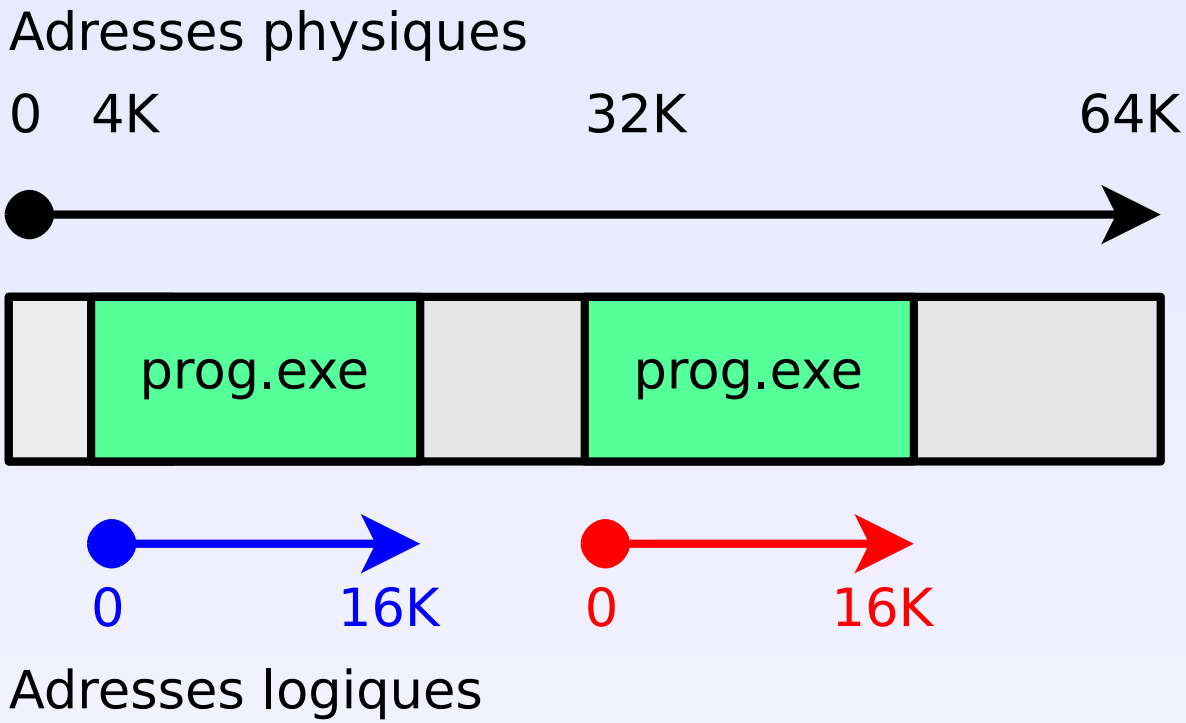
Indépendance code/position physique

Chaque processus utilise des adresses logiques.
Le compilateur génère du code relatif au début du programme

- toutes les adresses sont exprimées comme si le programme était chargé à l'adresse 0

➔ l'espace mémoire logique d'un processus commence à son adresse zéro

Indépendance code/position physique



Il faut alors traduire l'adresse logique utilisée dans le code en l'adresse physique correspondante dans la mémoire.

$$adresse\ physique = adresse\ de\ base + adresse\ logique$$

Conversion adresse logique/adresse physique

Première solution : conversion au chargement

- Le compilateur maintient une liste de tous les endroits du où l'on effectue une manipulation d'adresse.
- Au chargement, le système doit corriger les adresses en fonction de l'adresse de chargement du programme en suivant les indications du compilateur.

→ faisable mais coûteux

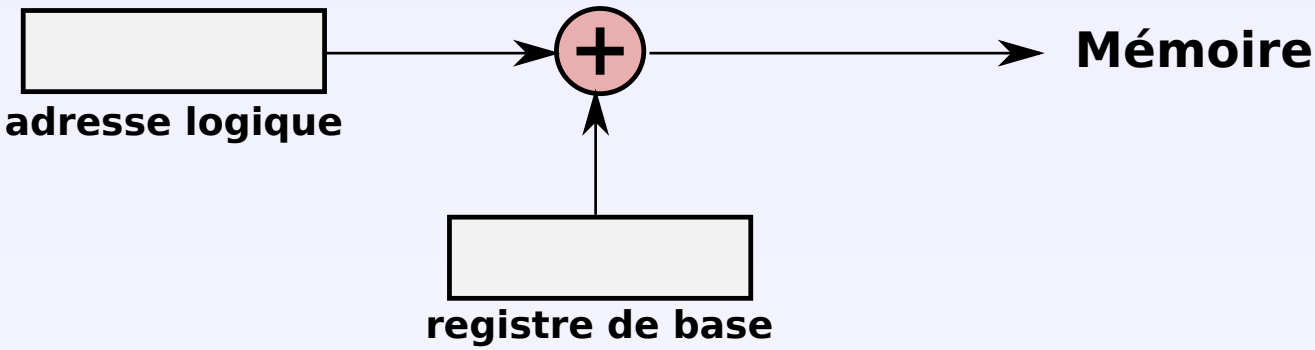
→ aucune protection mémoire

Modification du matériel

Deuxième solution : conversion à l'exécution

➔ Alternative matérielle

- le registre de base contient l'adresse de chargement du programme en mémoire.
- un additionneur ajoute à l'adresse logique la valeur du registre de base.



Protection inter-processus

Comment éviter qu'un processus n'accède à l'espace mémoire d'un autre ?

Protection inter-processus

Comment éviter qu'un processus n'accède à l'espace mémoire d'un autre ?

Une adresse logique valide est comprise entre

- 0
- la taille de l'espace logique (-1)

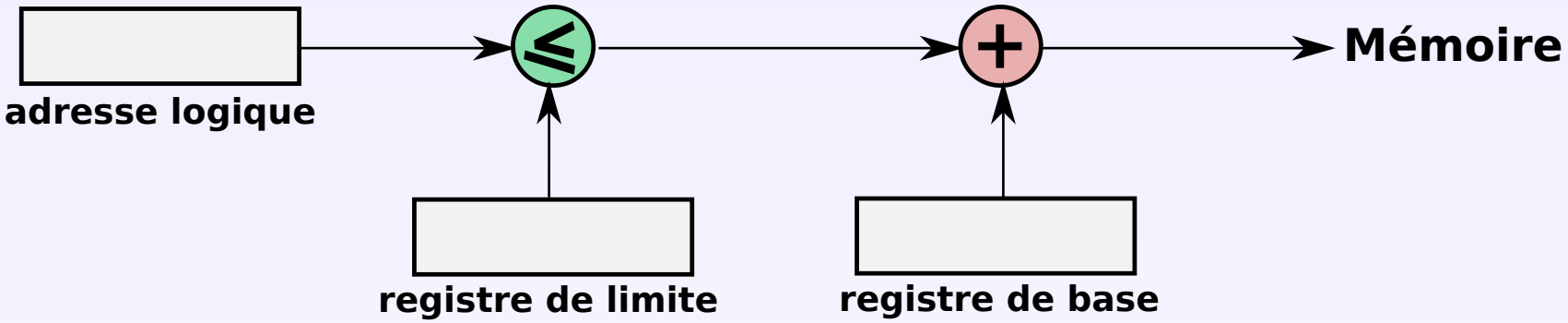
➔ comparer l'adresse logique avec la taille de l'espace logique.

Protection inter-processus

Modification du matériel : vérification des adresses logiques

- le registre de limite contient la taille de l'espace logique du processus en cours
- un comparateur effectue la comparaison entre l'adresse logique et la valeur du registre de limite

➔ génère une interruption si dépassement



Adresses logiques : Bilan

Introduction des adresses logiques

- code indépendant de sa position en mémoire
- simplifie le chargement des programmes
- facilite leur protection

Nécessite la conversion en adresse physique (réelle)

- réalisée par le matériel
- simple (utilisation de registres, additionneur, comparateur)
- à faible coût

Plan

1 Introduction

2 Partage de la mémoire

3 Adresses logiques

4 Gestion de l'allocation contiguë
- Fragmentation
- Solutions
- Bilan

5 Swapping

6 Segmentation

7 Espace mémoire paginé

8 Mémoire virtuelle paginée

9 Algorithmes de remplacement de page

10 Mémoire virtuelle segmentée paginée

Gestion de l'allocation contiguë

Initialement : un espace physique libre en mémoire

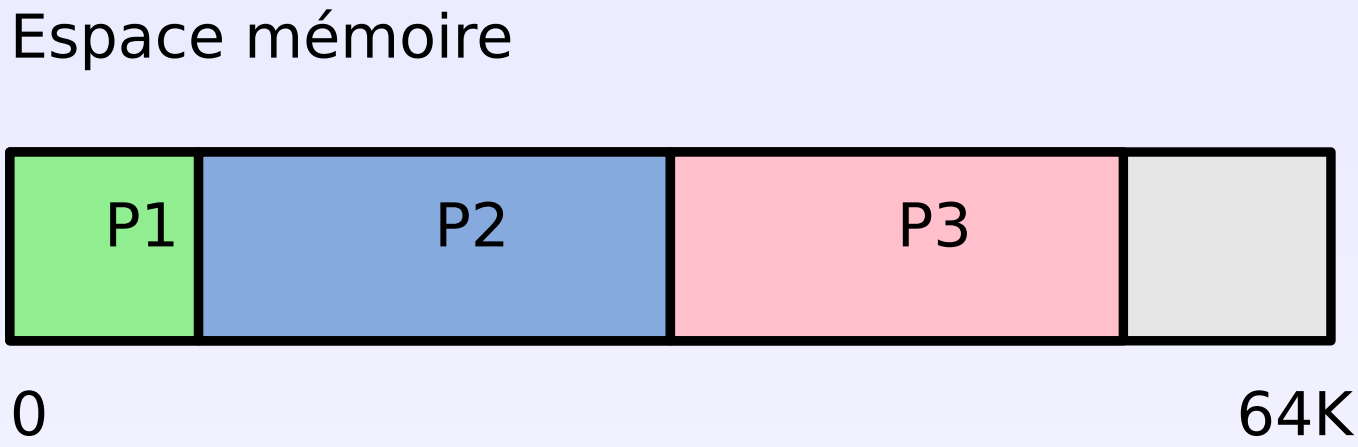
Évolution du traitement des programmes

- allocations de zones (prises dans l'espace libre)
- libération de zones allouées

➔ problème de fragmentation

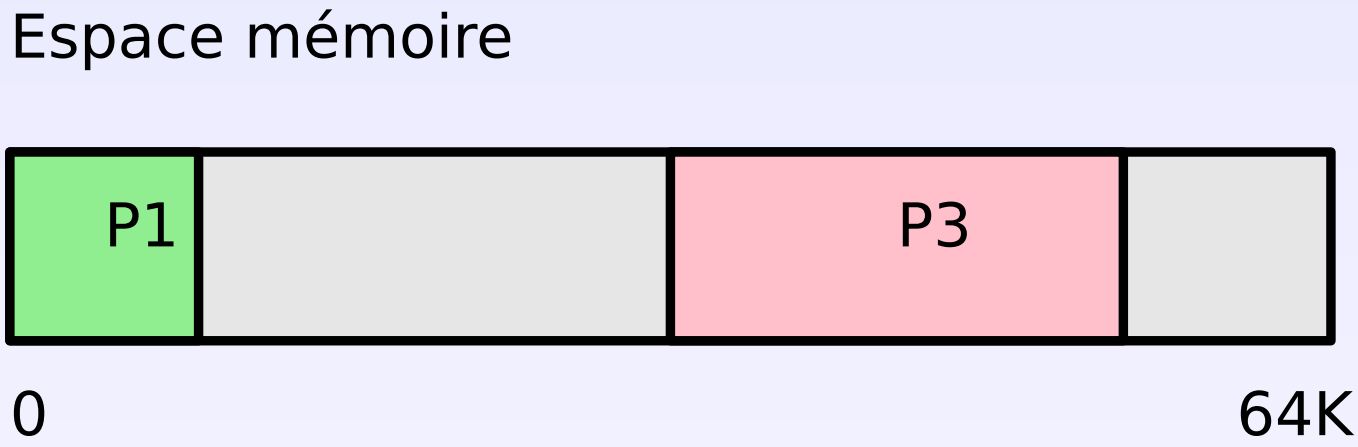
- allocations/libérations faites dans un ordre imprévisible
- tendance à laisser des trous
- les trous trop petits sont inexploitables

Fragmentation



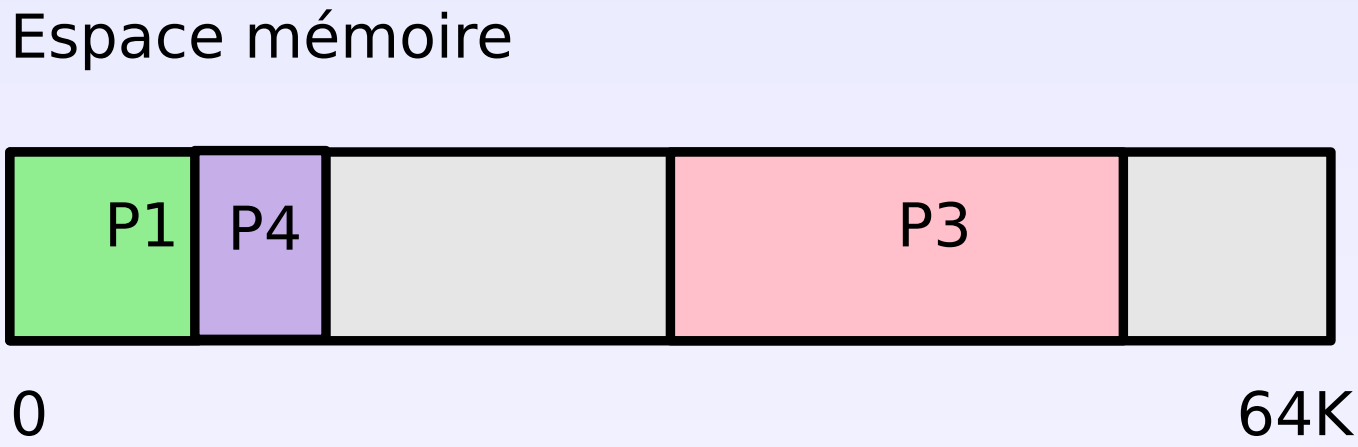
(0) 3 processus sont présents, P2 se termine ...

Fragmentation



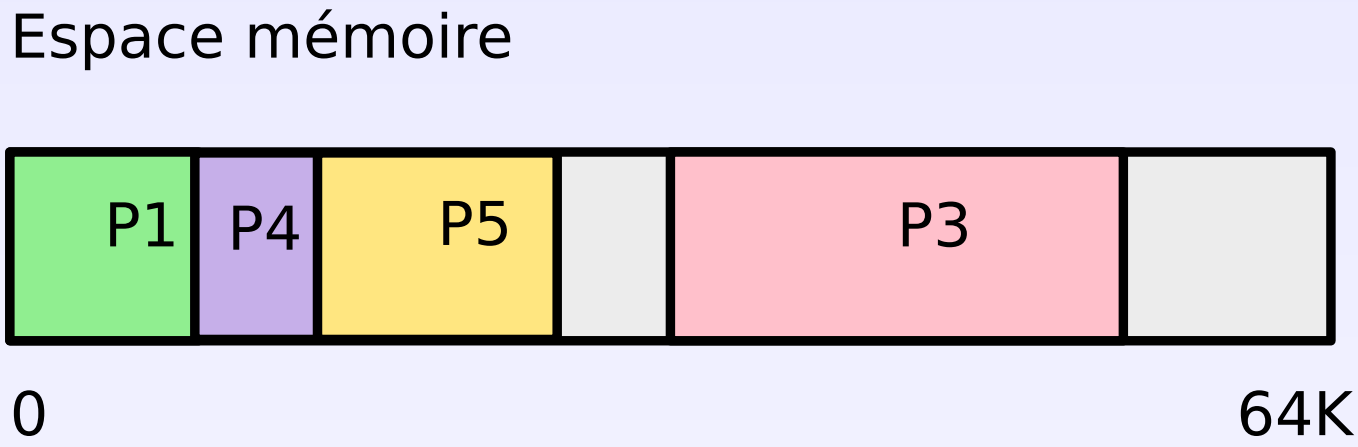
(1) allocation de P4 ...

Fragmentation



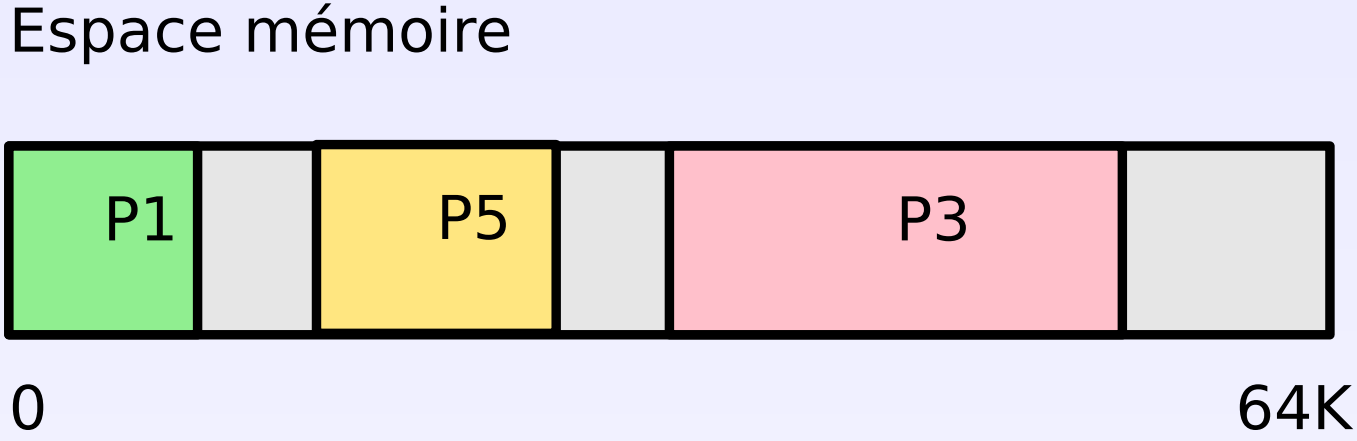
(3) allocation de P5 ...

Fragmentation



(4) libération de P4 ...

Fragmentation



(5) l'espace libre est fragmenté

Remèdes à la fragmentation

Solution curative : compactage (relocation)

- translater les zones allouées pour reconstituer une grande zone libre
- nécessite de lire toute la mémoire ➔ terriblement coûteux

Technique préventive : choix de la stratégie d'allocation

- first-fit, best-fit, worst-fit ...

Solutions

- first-fit : allocation dans le premier bloc assez grand pour contenir le programme
  - le reste devient une zone libre plus petite
- best-fit : allocation dans le plus petit bloc qui soit assez grand
  - méthode est plus coûteuse, évite de couper inutilement une grande zone, génère plus vite de très petites zones
- worst-fit allocation dans le plus grand bloc
  - limite les trop petites zones, pénalisant pour les gros programmes
- autres stratégies (buddy system, quick-fit)

Bilan

L'allocation de zones contiguës de tailles diverses conduit à la fragmentation

- problème retrouvé ailleurs (gestion des disques, `new`, `delete` en programmation, ...)
- les solutions préventives et curatives sont plus ou moins satisfaisantes

Plan

1 Introduction

2 Partage de la mémoire

3 Adresses logiques

4 Gestion de l'allocation contiguë

5 Swapping

6 Segmentation

7 Espace mémoire paginé

8 Mémoire virtuelle paginée

9 Algorithmes de remplacement de page

10 Mémoire virtuelle segmentée paginée

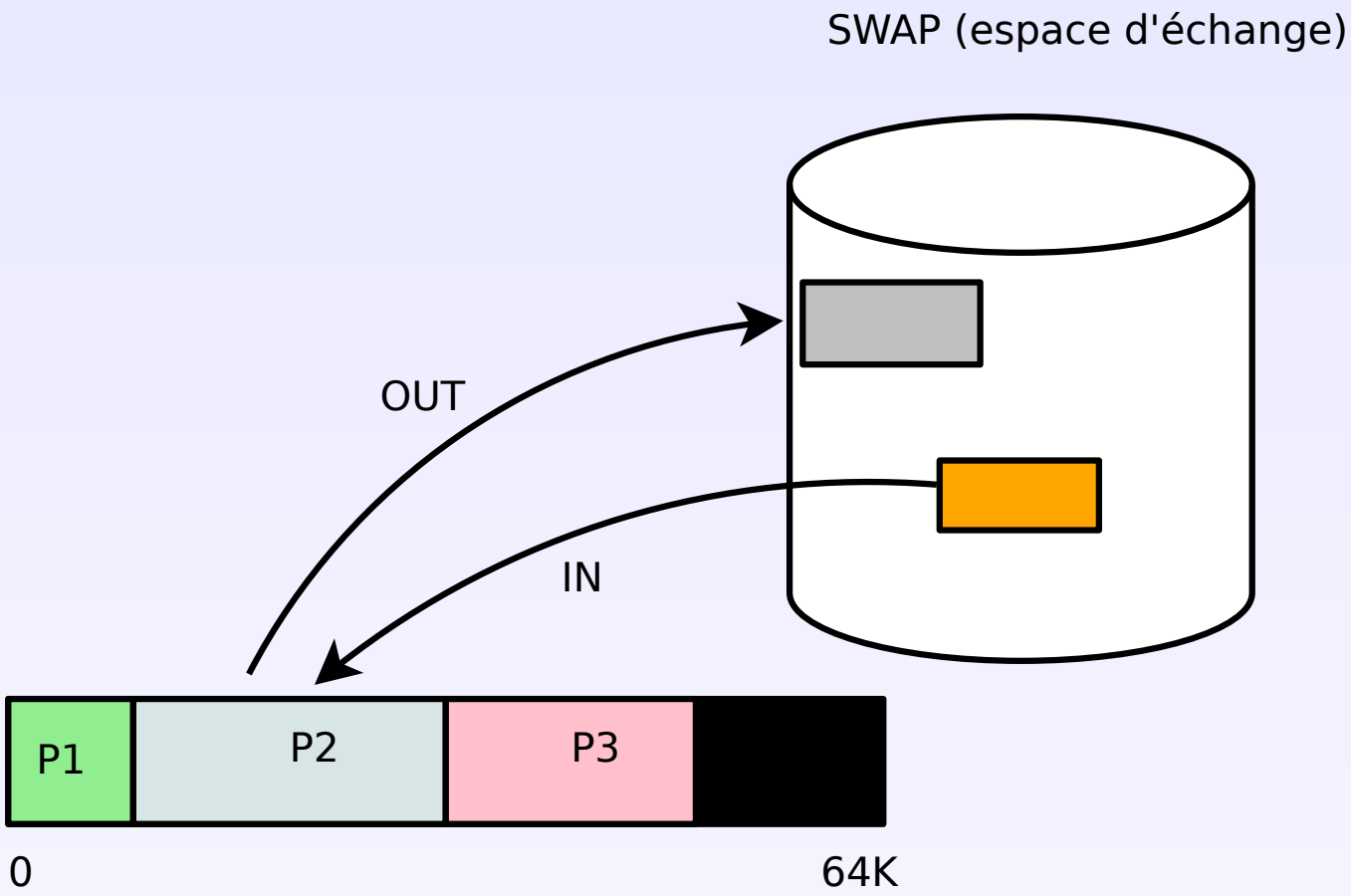
Swapping (va-et-vient)

La mémoire principale est une ressource limitée
➔ peut être insuffisante pour maintenir tous les processus courants (système à temps partagé)

Idée : conserver les processus supplémentaires sur le disque et les recharger pour les exécuter.

- transférer sur disque les processus inactifs
- les ramener le moment voulu

Swapping



Échanges mémoire ↔ disque

Avantages

On peut charger plus de processus en mémoire virtuelle que la mémoire centrale ne peut en contenir

Permet d'attendre des taux de charge élevés (rentabilisation)

Viable pour les applications interactives, l'utilisateur est un périphérique très lent

Plan

1 Introduction

2 Partage de la mémoire

3 Adresses logiques

4 Gestion de l'allocation contiguë

5 Swapping

6 Segmentation
- Bilan

7 Espace mémoire paginé

8 Mémoire virtuelle paginée

9 Algorithmes de remplacement de page

10 Mémoire virtuelle segmentée paginée

Segmentation

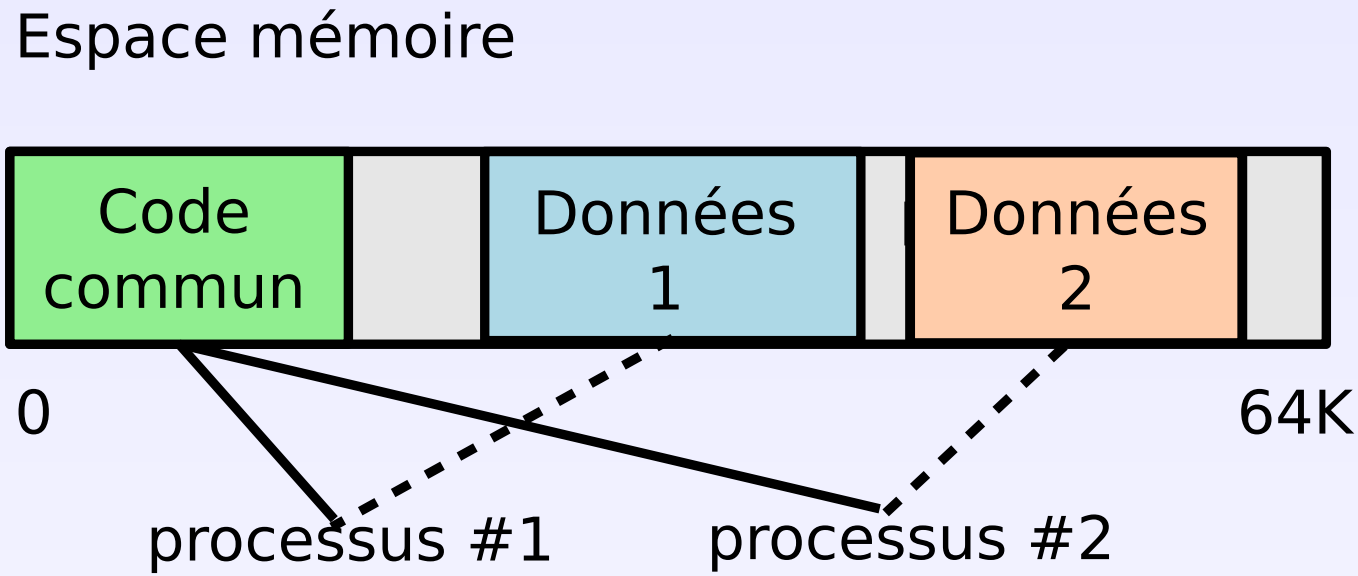
Principe : diviser l'espace mémoire en espaces logiquement indépendants (segments)

L'espace mémoire d'un processus comporte au moins deux segments :

- le segment de code
  - commun aux différentes instances du programme
  - un seul instance du code en mémoire, segment partagé entre les processus
- le segment de données
  - différent pour chaque processus

Avantages : partage segment (code, bibliothèque), économie mémoire, droits d'accès différents, ...

Segmentation



Deux processus exécutent le même code

Segmentation

Plusieurs segments ➔ position différentes en mémoire.

Adresse logique contient le couple :

$$< num\_segment, offset >$$

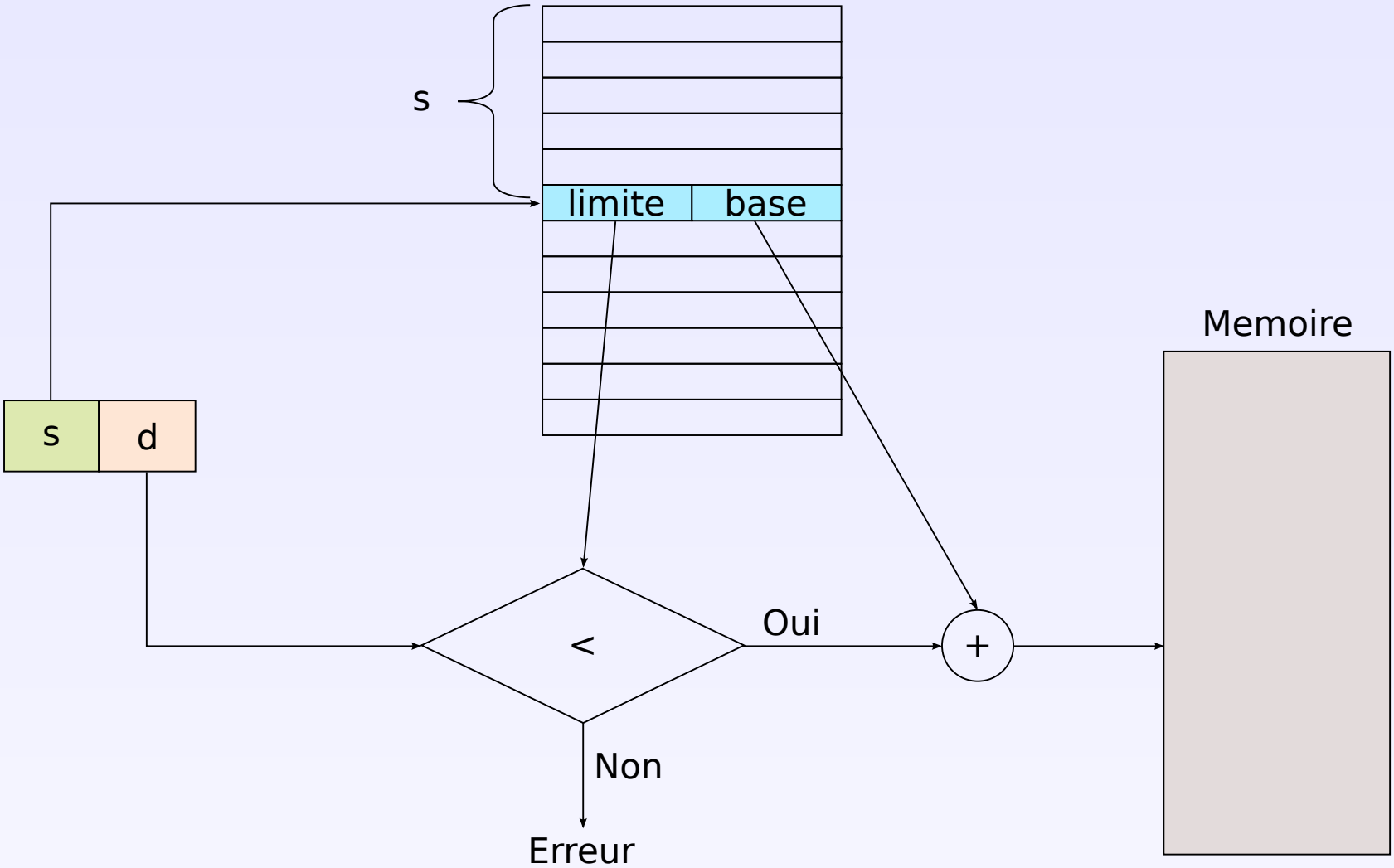
- *num_segment* = numéro de segment
- *offset* = position (déplacement) dans le segment

La table des segments permet d'effectuer la translation entre adresse logique et adresse physique

Chaque entrée de la table contient :

- l'adresse de base du segment,
- la taille (longueur) du segment.

Tables des segments



source : Wikipedia

Remarque : numéros de segments

On distingue la numérotation de segment

- locale : propre à chaque processus
- globale : commune

Deux processus peuvent utiliser le même segment réel avec des numéros locaux (éventuellement) différents.
Remarque :

- La table locale des segments (TLS) d'un processus fournit le numéro de segment global à partir du numéro de segment local
- La table globale de segments (TGS) contient la description de chaque segment (longueur, pages, protections ...)

Bilan

La segmentation permet le découpage mémoire d'un programme en plusieurs segments logiques.

- adresse logique représentée par un numéro de segment et une position
- utilisation d'une table des segments
- permet un gain de place important :
  - évite la duplication de code, segments utilisés pour les bibliothèques communes.

Plan

1. Introduction
2. Partage de la mémoire
3. Adresses logiques
4. Gestion de l'allocation contiguë
5. Swapping
6. Segmentation
7. Espace mémoire paginé
8. Mémoire virtuelle paginée
9. Algorithmes de remplacement de page
10. Mémoire virtuelle segmentée paginée

Espace mémoire paginé

Contexte : Partage de la mémoire physique par des processus
Problème : fragmentation causée par l'allocation et la libération d'espaces de tailles diverses ➔ gaspillage de mémoire.
Idée : déporter le problème à l'intérieur des processus

- espace découpé en pages de mêmes tailles
- les pages d'un même espace ne sont pas forcément consécutives
- la table des pages indique la position en mémoire des pages d'un processus

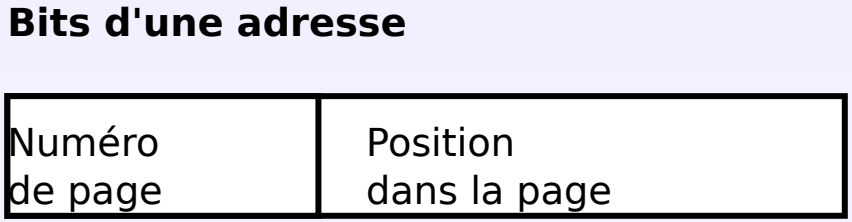
➔ Pagination

Pagination

Espaces logiques découpés en pages logiques : souvent 4Ko.

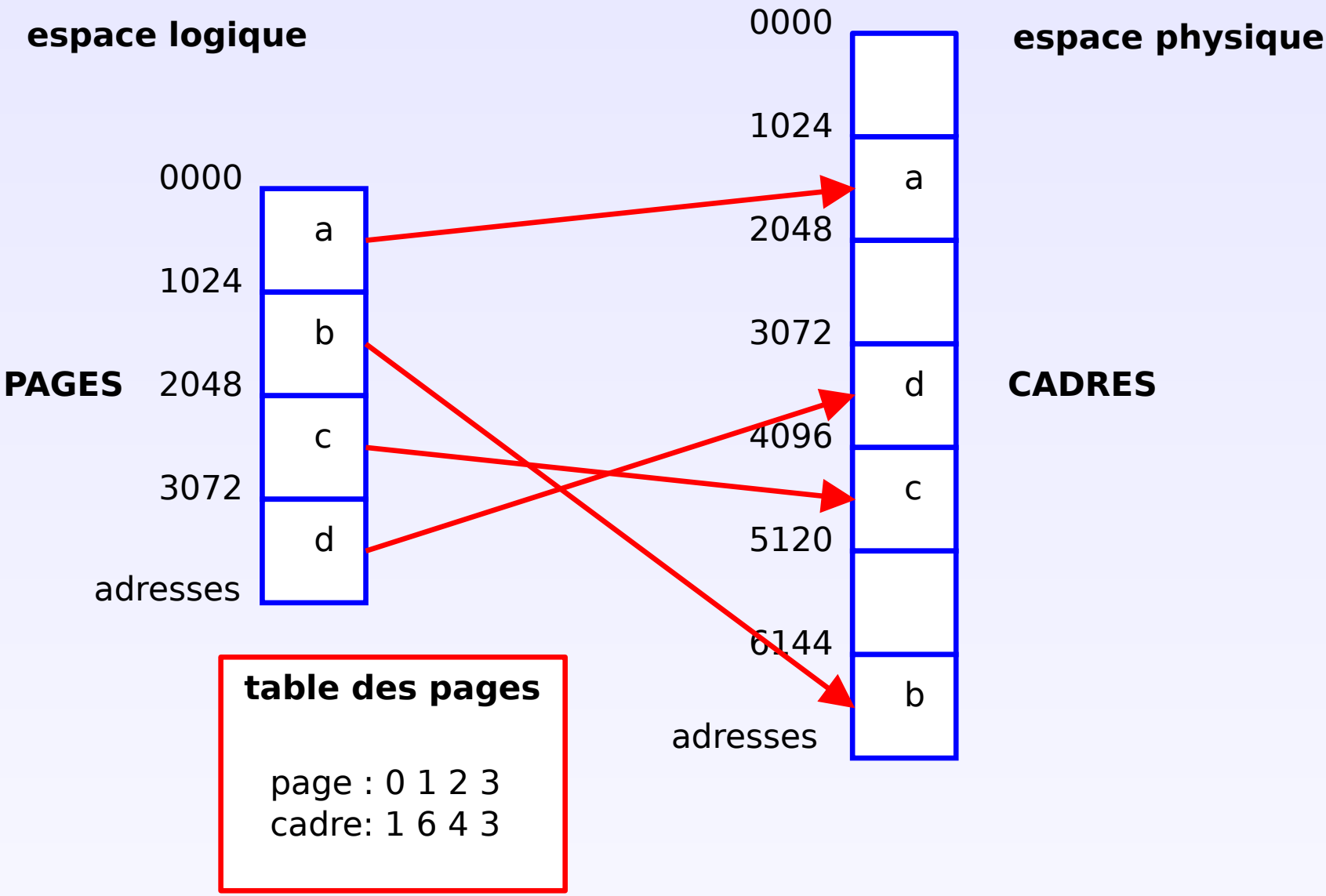
La mémoire physique est découpée en cadres de pages de la même taille.

Adresse logique comprend : le numéro de page et la position dans la page.



Pour chaque processus, la table des pages permet de retrouver la correspondance avec l'adresse physique.

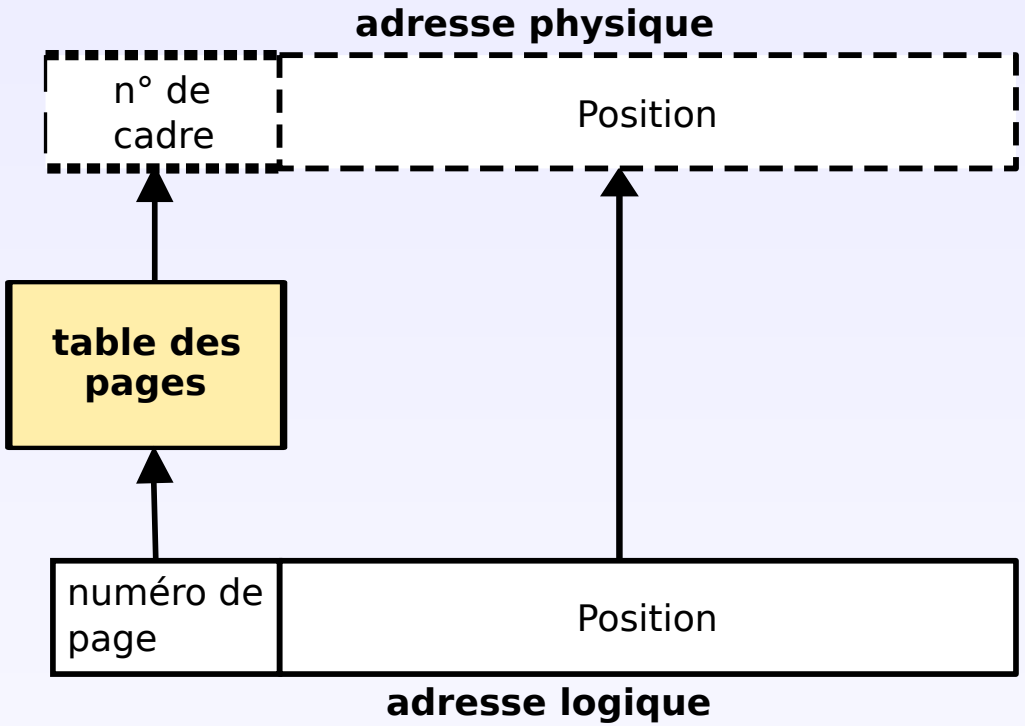
Espace paginé



Génération d'adresses

Le numéro de page est codé par les bits de poids forts
- il est converti en numéro de cadre de page

Les bits de poids faible donnent la position dans la page
- cadre de page et page logique de même taille
  ➔ déplacement équivalent



Plan

1 Introduction

2 Partage de la mémoire

3 Adresses logiques

4 Gestion de l'allocation contiguë

5 Swapping

6 Segmentation

7 Espace mémoire paginé

8 Mémoire virtuelle paginée
- Unité de Gestion Mémoire (MMU)

9 Algorithmes de remplacement de page

10 Mémoire virtuelle

Mémoire virtuelle paginée

Principe : combinaison de la pagination et du swapping.

- la table des pages connaît les pages présentes
- interruption en cas de défaut de page (la page référencée n'est pas en mémoire)

Unité de Gestion Mémoire (MMU)

Circuit spécialisé Memory Management Unit
En charge de :

- la translation d'adresses logiques (virtuelles) en adresses physique
- La protection mémoire
- contrôle de cache
- l'arbitrage du bus

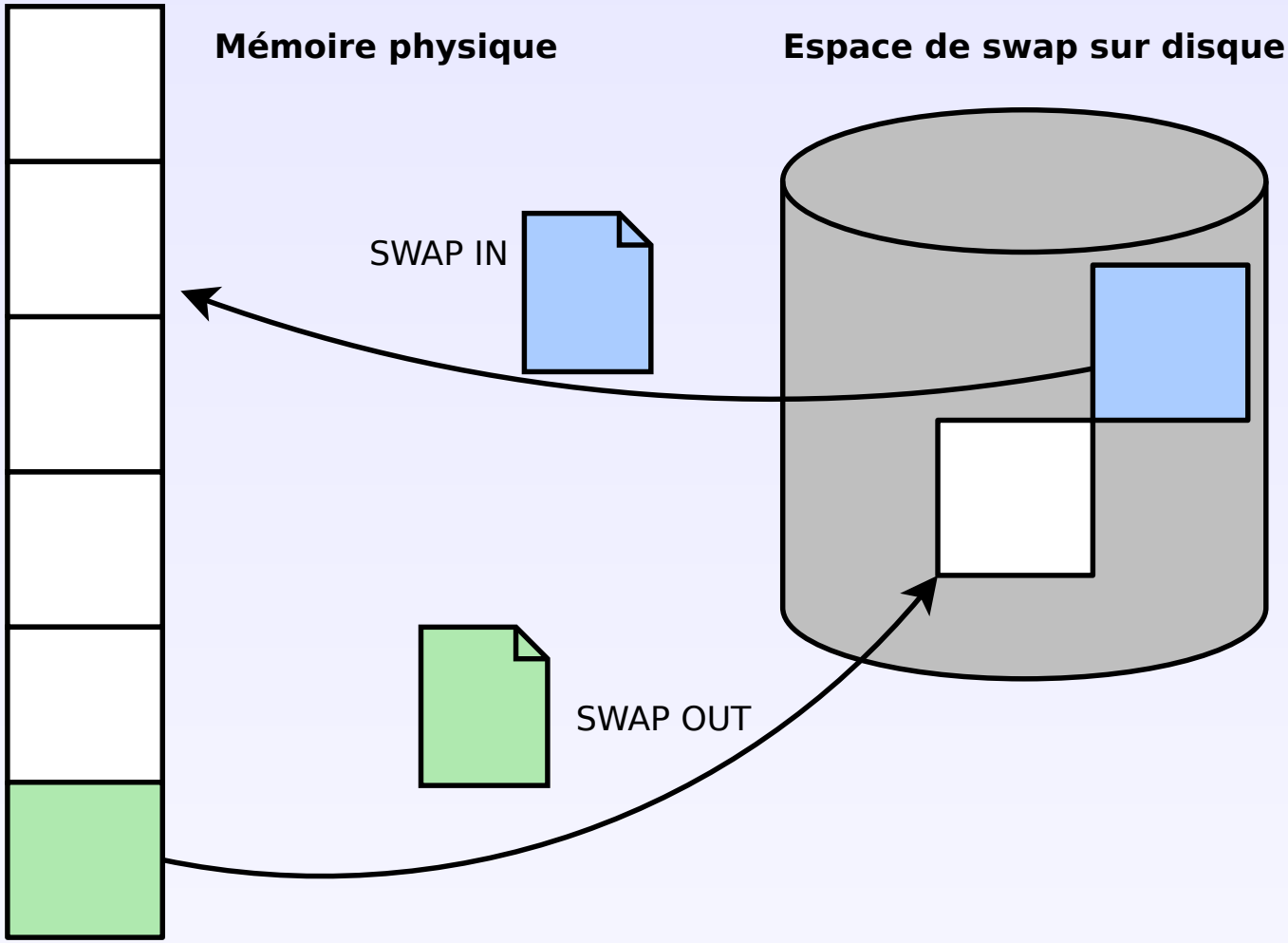
En cas de défaut de page

Le système d'exploitation reprend la main

- S'il y a des cadres vides, il ramène les pages demandées.
- Sinon il évacue une page sur disque pour libérer un emplacement

- il installe la page manquante,
- met à jour la MMU
- et relance l'instruction interrompue.

Swapping



Plan

1. Introduction
2. Partage de la mémoire
3. Adresses logiques
4. Gestion de l'allocation contiguë
5. Swapping
6. Segmentation
7. Espace mémoire paginé
8. Mémoire virtuelle paginée
9. Algorithmes de remplacement de page
10. Mémoire virtuelle segmentée paginée

Algorithmes de remplacement de page

Comment choisir la page à remplacer ? Idéal : choisir la page qui sera utilisée le plus tard possible (voir plus du tout). Pour choisir un emplacement à libérer, en général on choisit les pages *les moins utiles*

Plusieurs algorithmes :

- FIFO
- LRU (Least Recently Used)
- NRU : Not Recently Used

Plan

1. Introduction
2. Partage de la mémoire
3. Adresses logiques
4. Gestion de l'allocation contiguë
5. Swapping
6. Segmentation
7. Espace mémoire paginé
8. Mémoire virtuelle paginée
9. Algorithmes de remplacement de page
10. Mémoire virtuelle segmentée paginée

Mémoire virtuelle segmentée paginée

Une adresse virtuelle comporte un numéro de segment, et un déplacement.

- chaque processus possède ses numéros de segment.
- chaque segment est composé de pages.

Implémentation

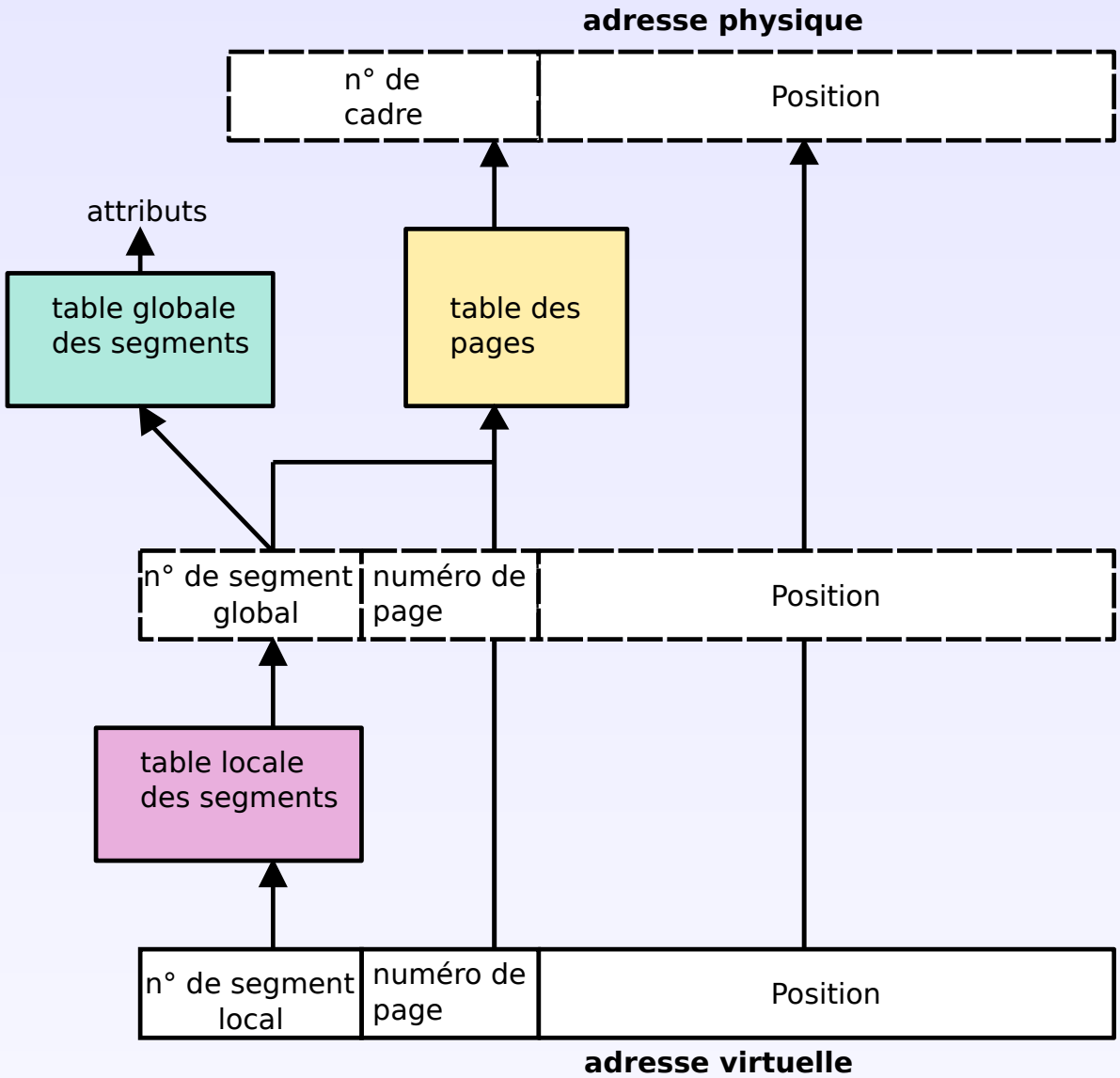
La table (TLS) traduit le numéro de segment local en numéro de segment global

Le couple (numéro de segment global, numéro de page) permet de retrouver le numéro de cadre de page correspondant

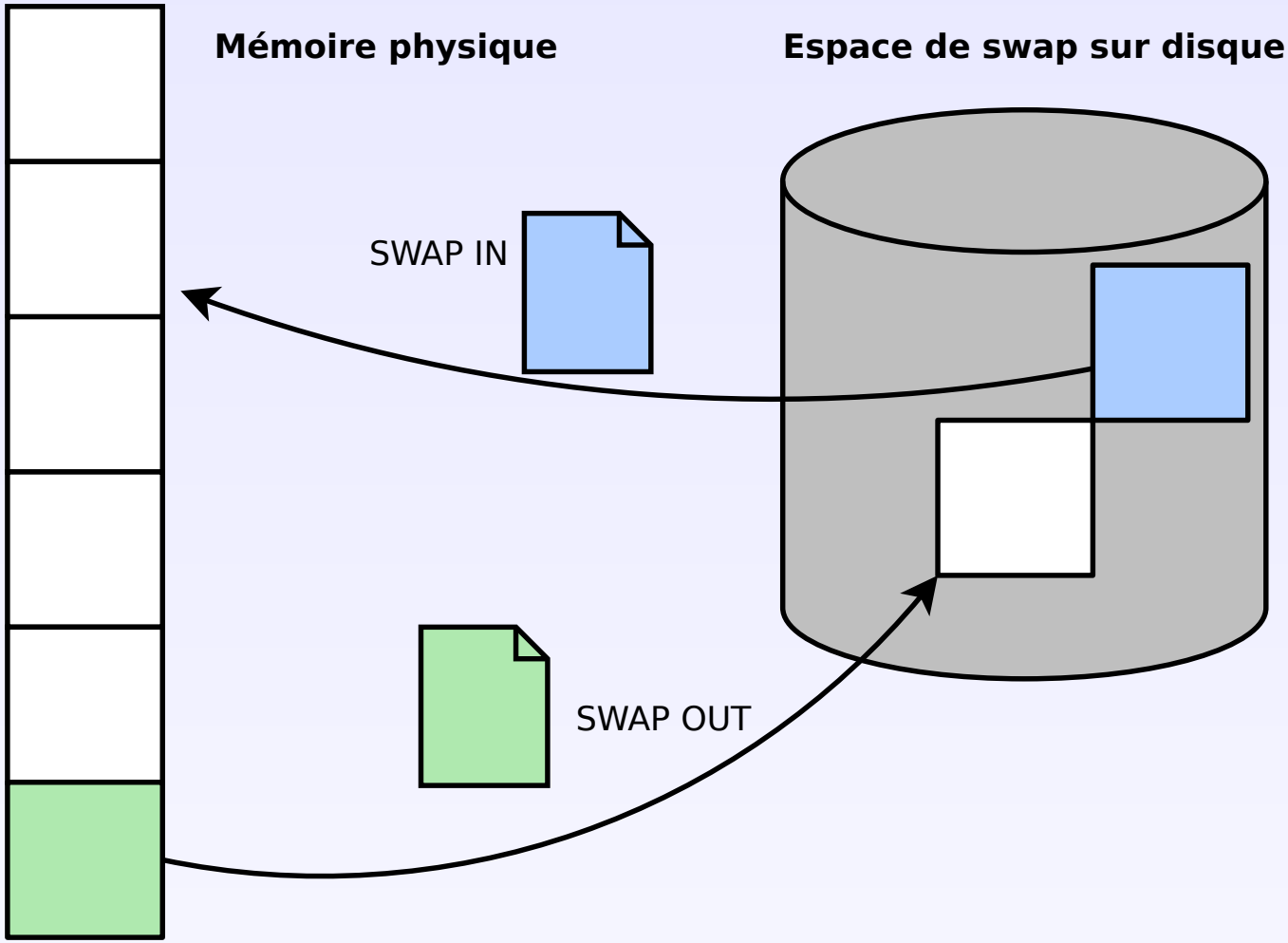
Le numéro de cadre de page et la position dans la page forment l'adresse physique.

La MMU utilise une mémoire associative pour mémoriser les correspondances
(numéro de segment global, num. de page → numéro de cadre de page)

Génération d'adresse



Combinaison avec le swapping



Bilan

Combinaison de :

- pagination,
- segmentation,
- swapping

Permet :

- la cohabitation entre les processus
- la protection
- les zones partagées
- l'économie mémoire

Technique largement adoptée depuis les années 70