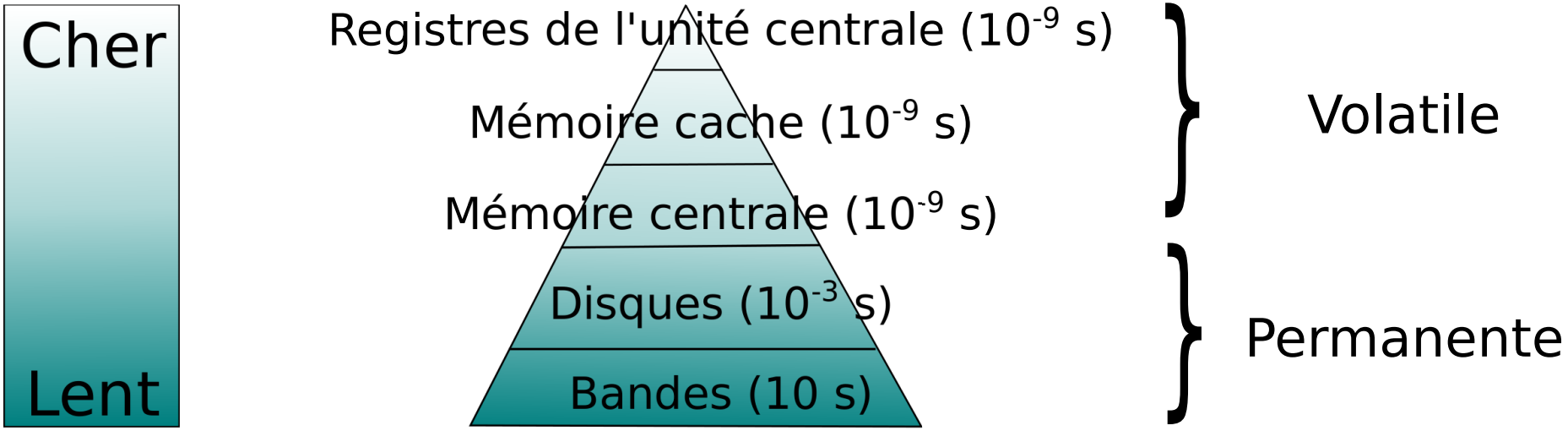
Chapitre 5

La mémoire

Types de mémoire

- Ressource cruciale pour les performances d'un système
- Impact en termes de sécurité
- Complexe en cas de multitâche
- Types, vitesses, coût...

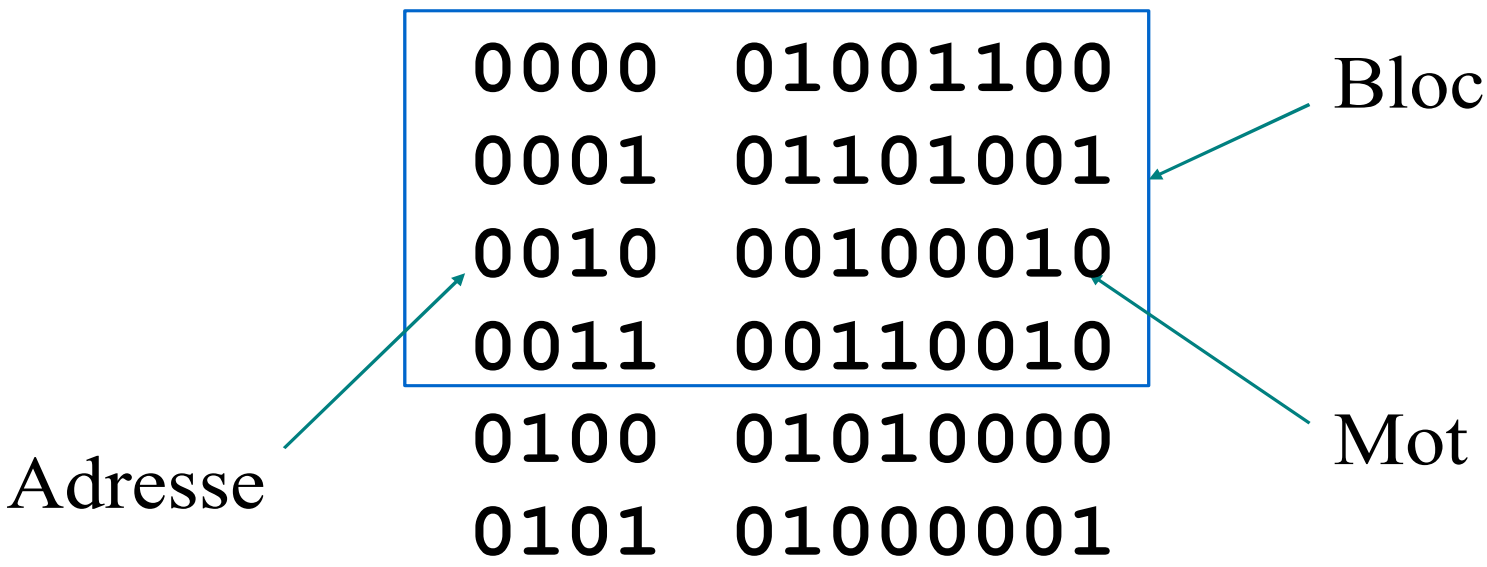


Partager la mémoire

- Partage entre les processus (droits d'accès)
- Différentes zones par processus (sécurité, efficacité)
- Connaître les zones libres et occupées
- Éviter le gaspillage de mémoire
- Gérer la mémoire dynamiquement
- Utilisation de mémoire virtuelle
- Stratégies de chargement/déchargement

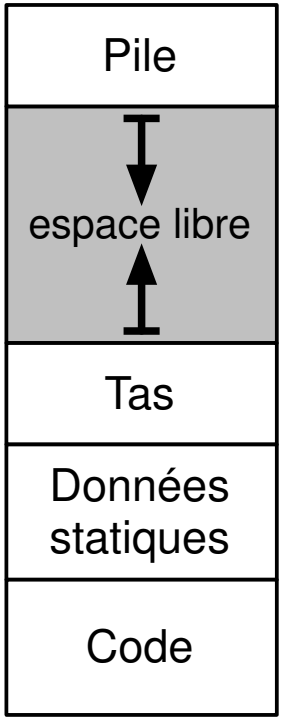
Organisation

- Mémoire considérée comme *linéaire* :
- L'espace mémoire est constitué d'un ensemble de *mots* ayant chacun sa propre adresse.
- La mémoire est généralement manipulée par *blocs* constitués de plusieurs mots.
- Un mot mémoire = 1 Octet (ou la plus petite zone de mémoire manipulable)



Segmentation de la mémoire pour un processus

- Découpage de la mémoire suivant l'usage qu'on en fait
- Vérification des accès à ces zones pas l'OS
- Processus : Code, Données, Tas, Pile
- Le tas peut référencer des pages hors de la zone du processus.
- Tas et pile se partagent le même espace.



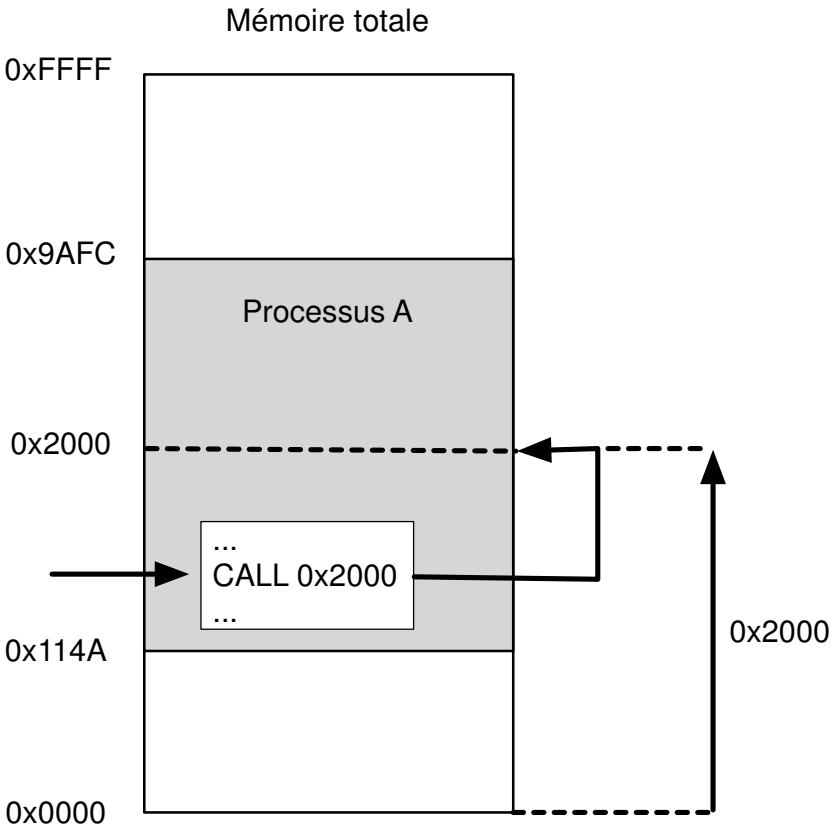
Translation d'adresse

- Un exécutable fait référence à des adresses mémoires.
- Adressage absolu inadapté : processus toujours chargés au même endroit

$\Rightarrow$ Utilisation d'adresses *relatives* :

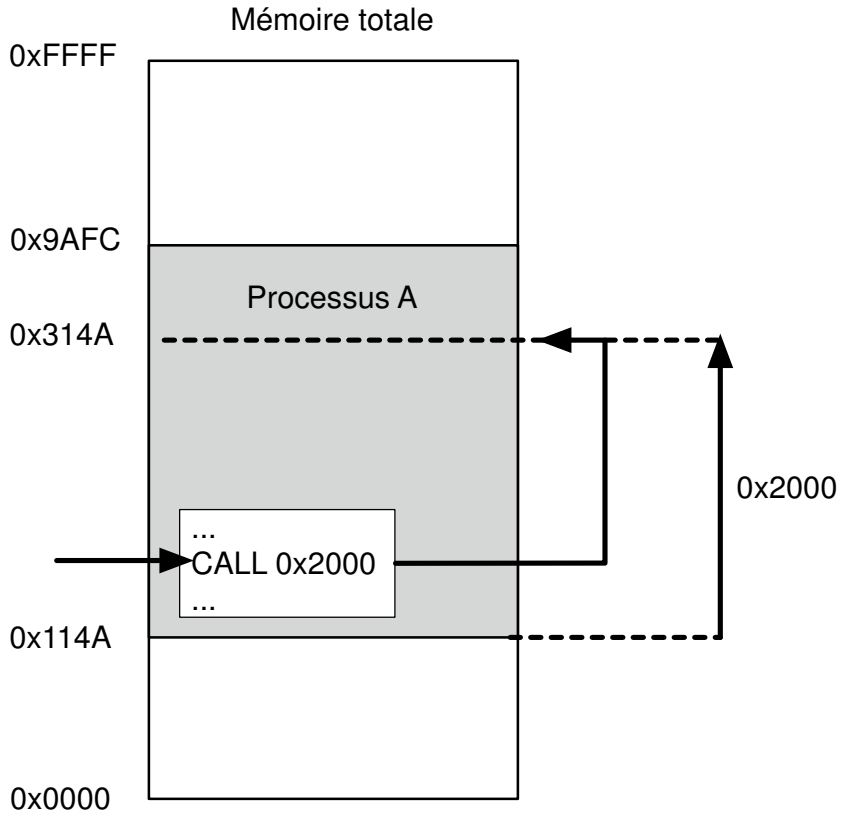
- Décalage d'un décalage (*offset)* relativement à une *base*
- Adresses de début des segments stockés en registres (CS, DS..)

Adressage *absolu* vs Adressage *relatif*



Adressage **absolu**

Appel du code à l'adresse 0x2000



Adressage **relatif**

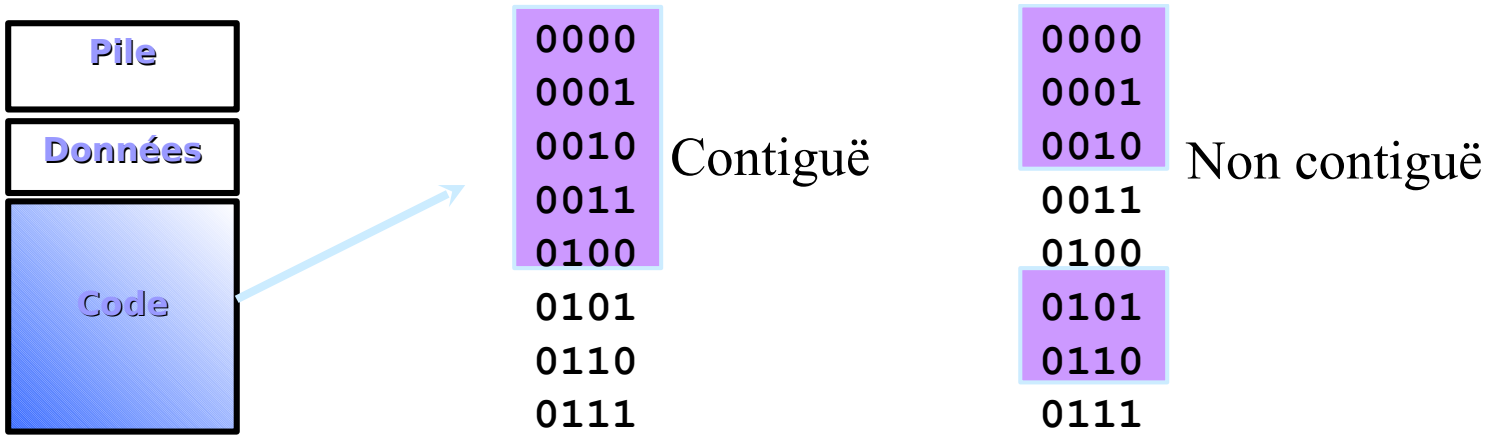
Appel du code au déplacement 0x2000 par rapport au processus courant

Adresses relative

- Code repositionnable
- Registre (Data Segment, Base Segment) contient l'adresse de base.
- Limitation du décalage pour rester dans le même segment.
- Translation d'adresse faite à la demande.

Méthode d'allocation

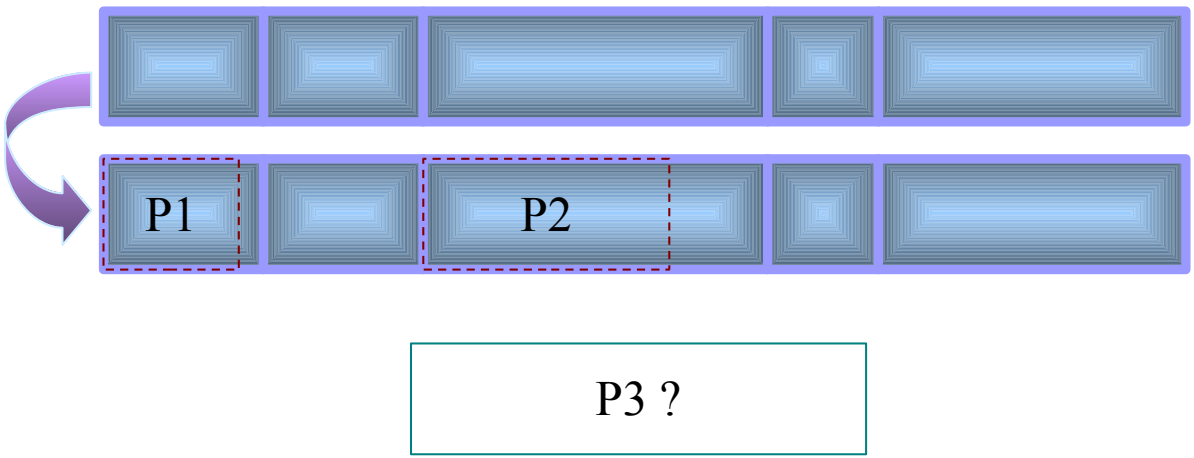
- Un processus est constitué de plusieurs "Zones" (Les segments de code, pile et données en sont trois exemples).



  - Contiguë : Une zone est constituée d'adresses consécutives.
  - Non contiguë : Une zone peut être fragmentée en blocs épars.

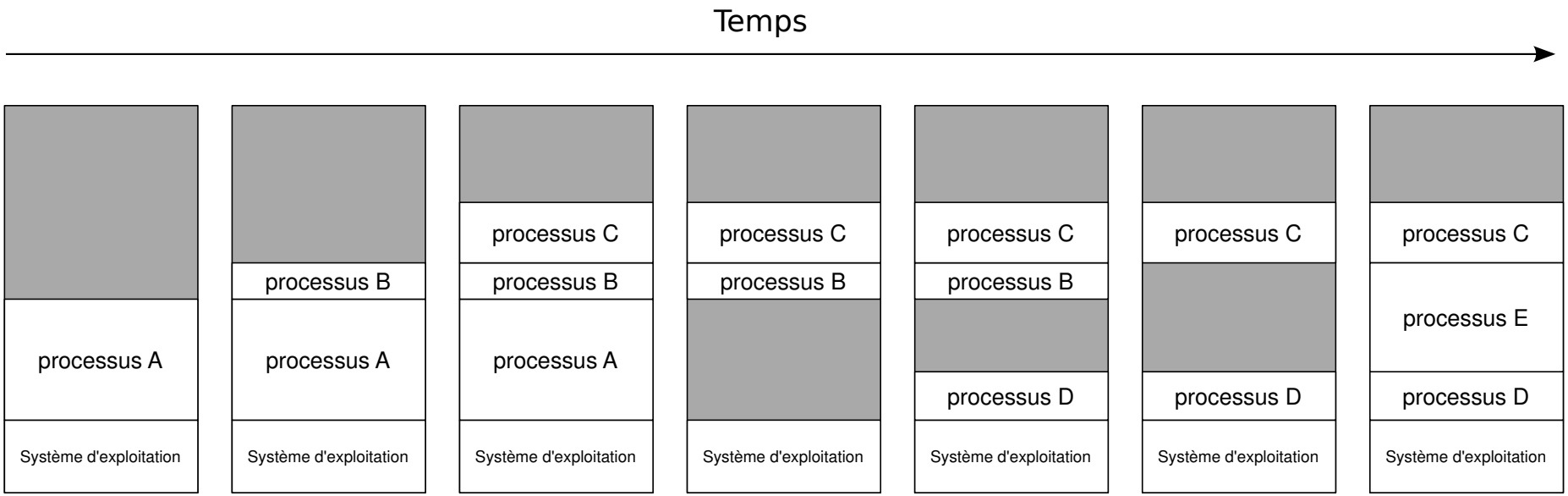
Allocation Statique

- Partitions statiques



Allocation Dynamique

- La taille des partitions s'adapte à la demande
- La fragmentation persiste.

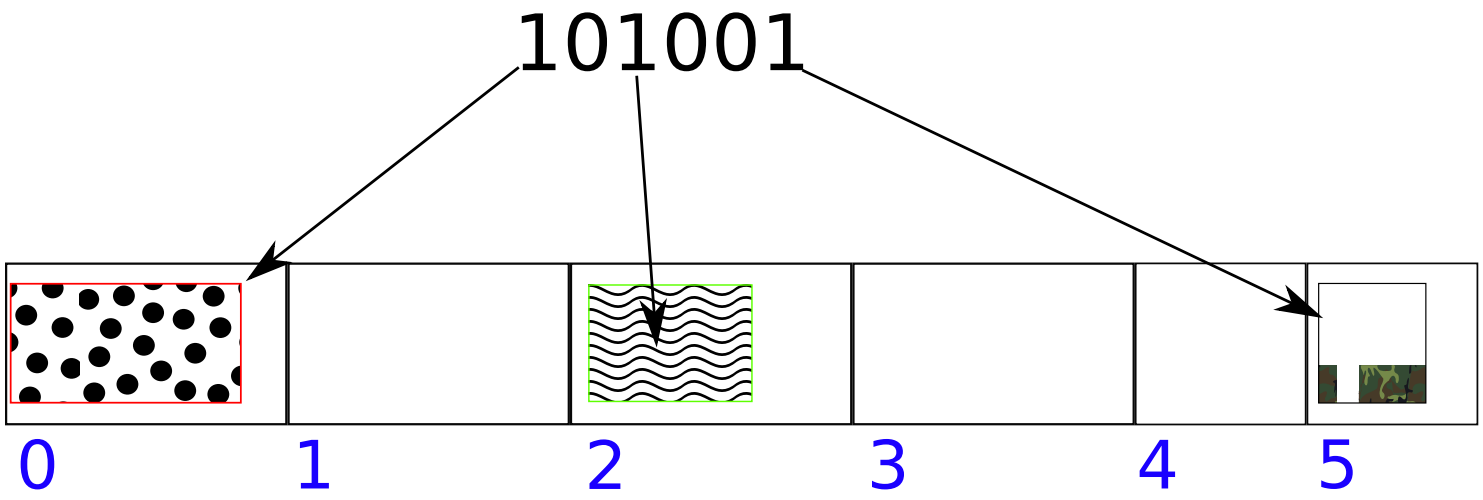


Stratégies de placement

- First Fit : le premier emplacement possible
  - Avantage :
  - Inconvénient :
- Best fit : l'emplacement dont la taille est la plus proche
  - Avantage :
  - Inconvénient :
- Worst fit : l'emplacement le plus gros
  - Avantage :
  - Inconvénient :

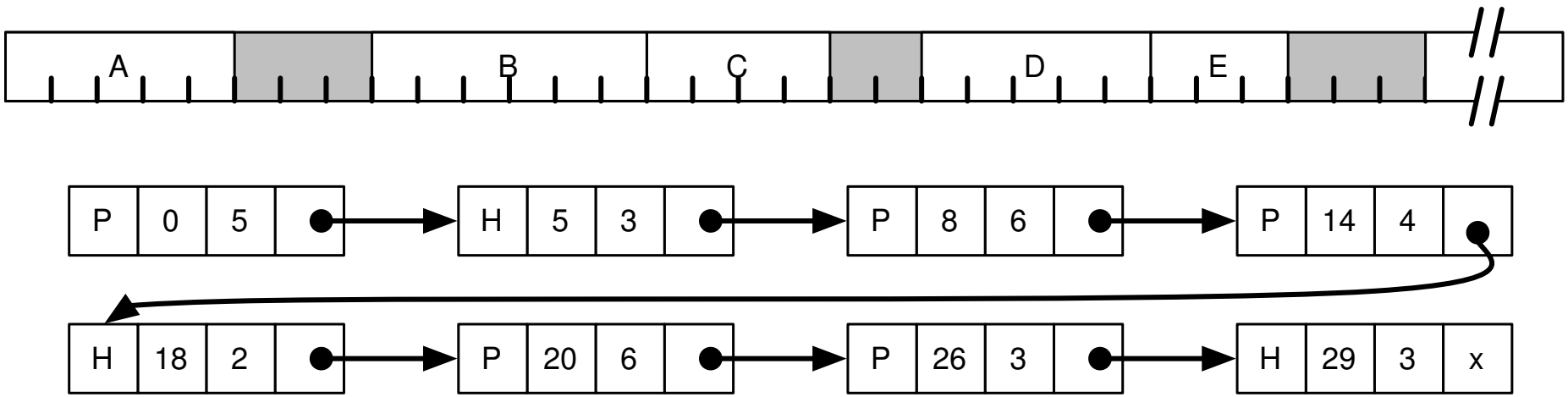
Mémorisation de l'occupation : table de bits

- Table de bits : Principe
  - Tableau avec un bit par cadre
  - 0 : libre, 1 : occupé
- Table de bits : Inconvénients
  - Recherche de *n* bits libres successifs lente
  - Taille du tableau ?



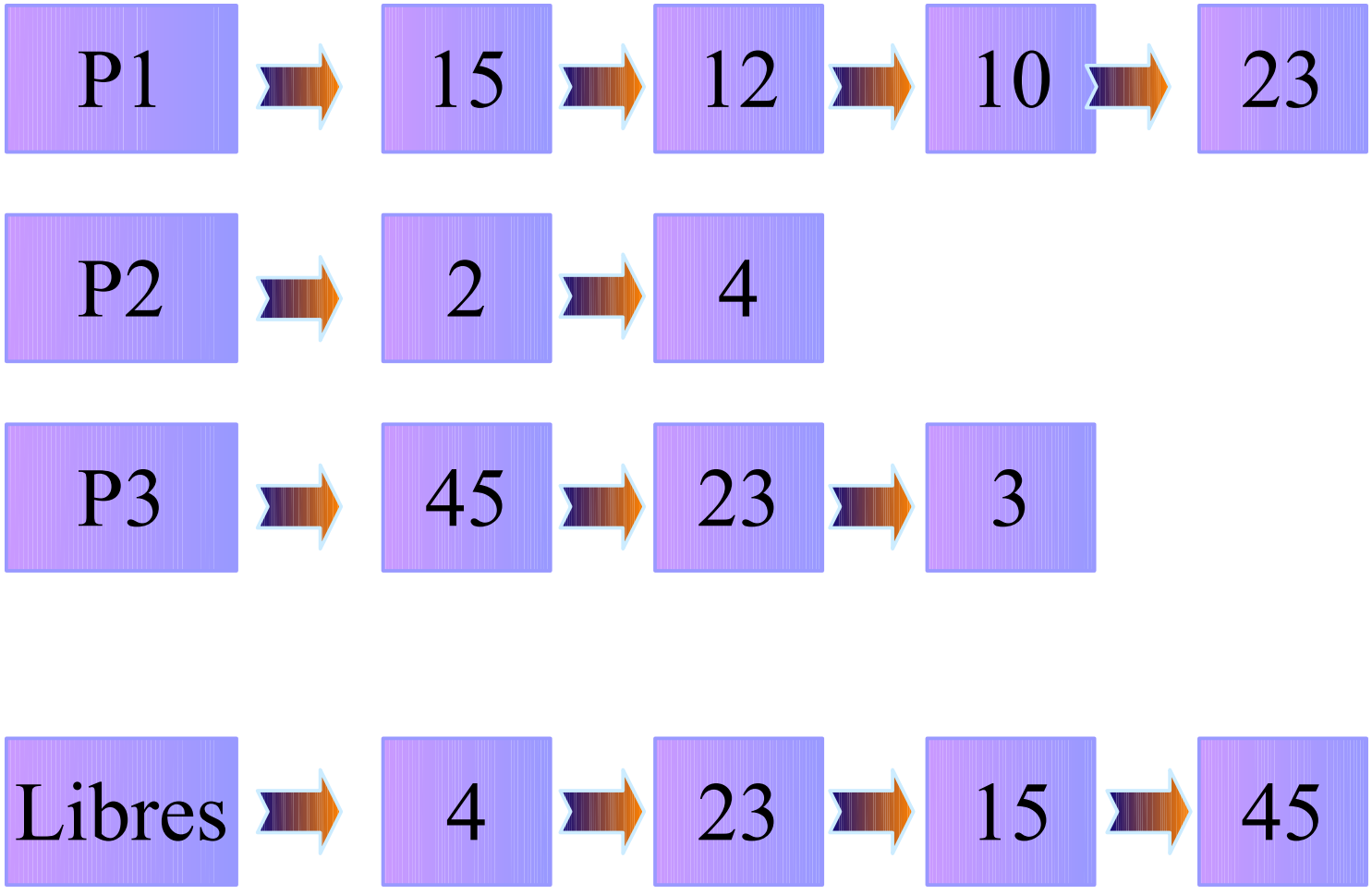
Mémorisation de l'occupation : liste chaînée

- Liste chaînée de blocs indiquant le début et la longueur du bloc, ainsi que si il est libre (H=Hole) ou occupé (P=processus)
- Recherche rapide mais coût élevé de MAJ.



Mémorisation : Unix

- Une liste par processus, une liste pour l'espace libre.

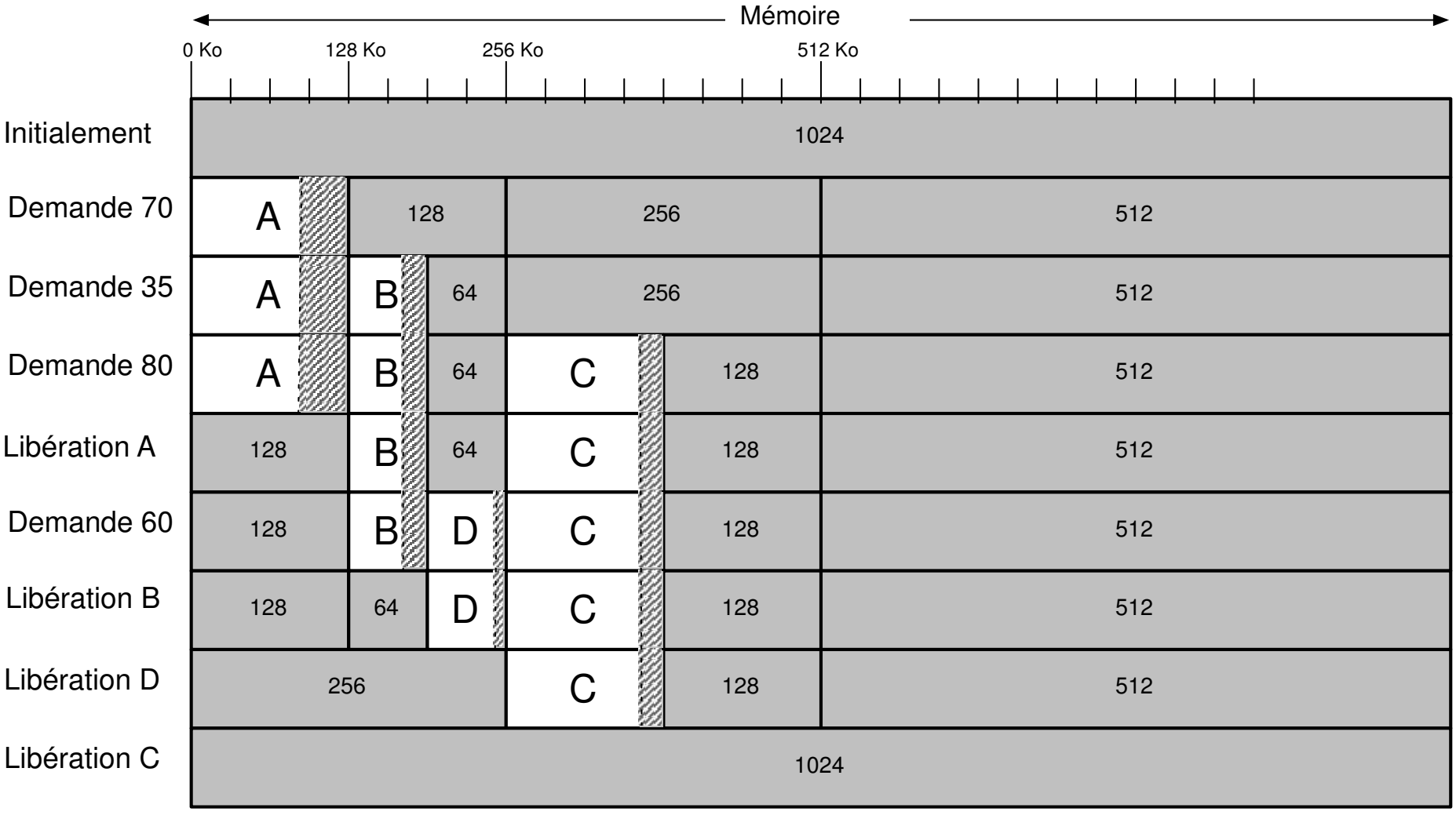


Mémorisation : Buddy Memory

- Idée : couper chaque bloc de mémoire en deux *buddy* (amis) qui seront très rapides à fusionner
- Principe : Ne gérer que des blocs d'une taille d'une puissance de 2 (1,2,4,8,16,...128,...1024... unités).
- Une liste ordonnée de blocs par taille.
- Pour utiliser un bloc libre, on le découpe en deux jusqu'à obtenir une bloc de taille minimum qui convienne.
- Recherche, division et fusion très rapides.
- Beaucoup de fragmentation.

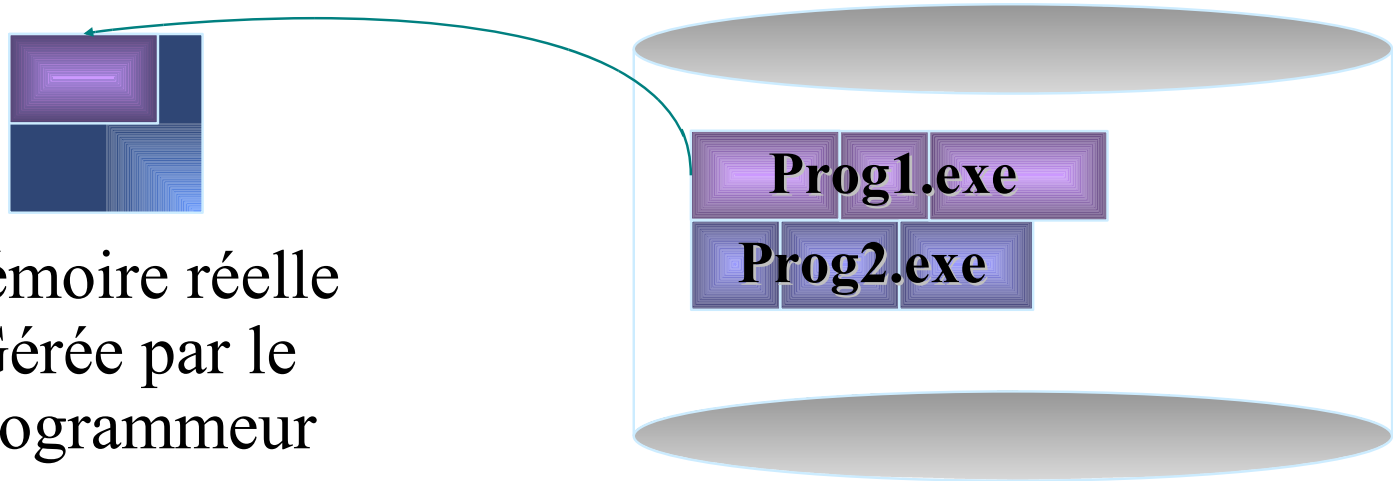
Mémorisation : Buddy Memory

- Exemple de fonctionnement de Buddy Memory (division)



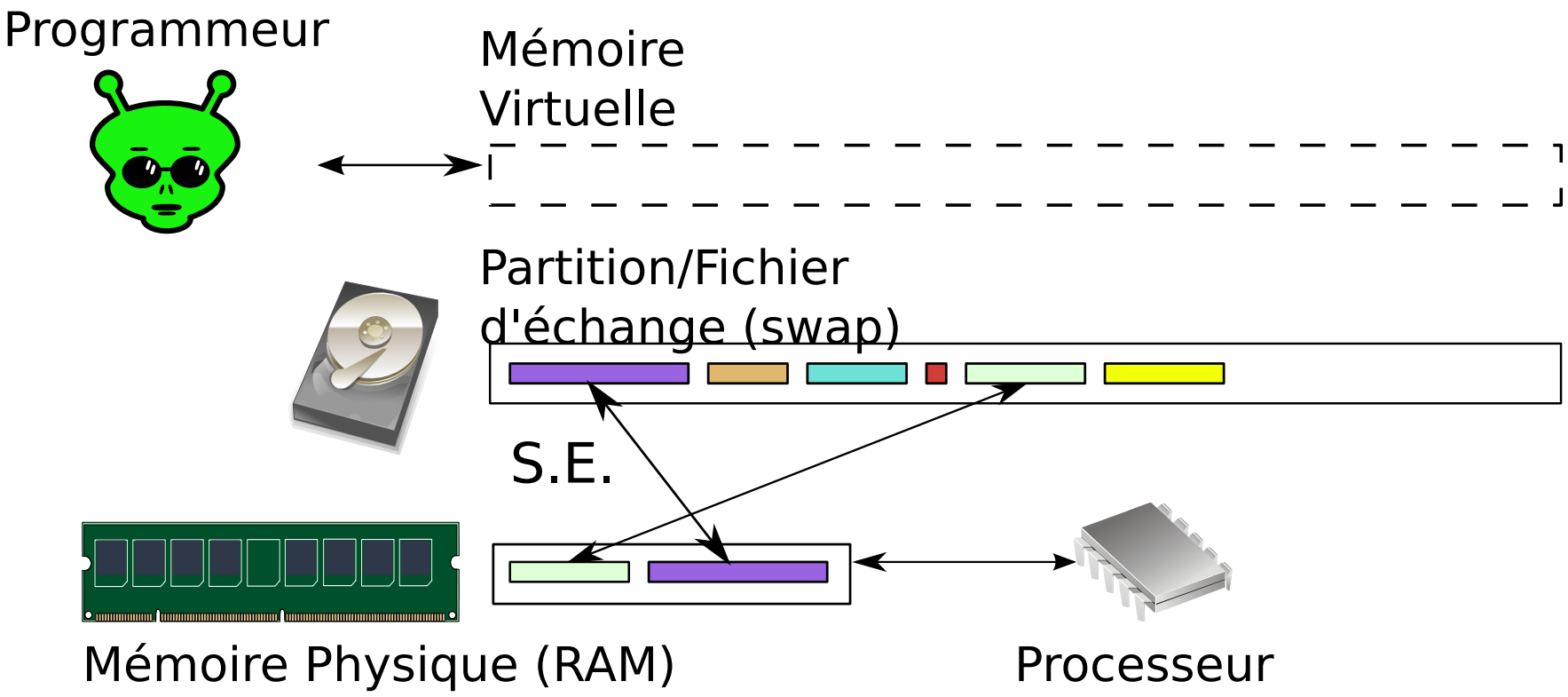
L'ancêtre de la mémoire virtuelle : les Overlays

- But : exécuter un programme plus grand que la mémoire
- Idée : laisser sur le disque ce qui n'a pas besoin d'être en mémoire
- Overlay : Zones de programme (dé)chargées au besoin par le programmeur
- PC/Dos (puis MS/Dos). Dépassé.



Mémoire virtuelle

- Utilise le disque dur pour compenser le manque de mémoire physique.
- Le système d'exploitation gère la mémoire...
- ... pas le programmeur.
- On parle parfois de *mémoire secondaire*.

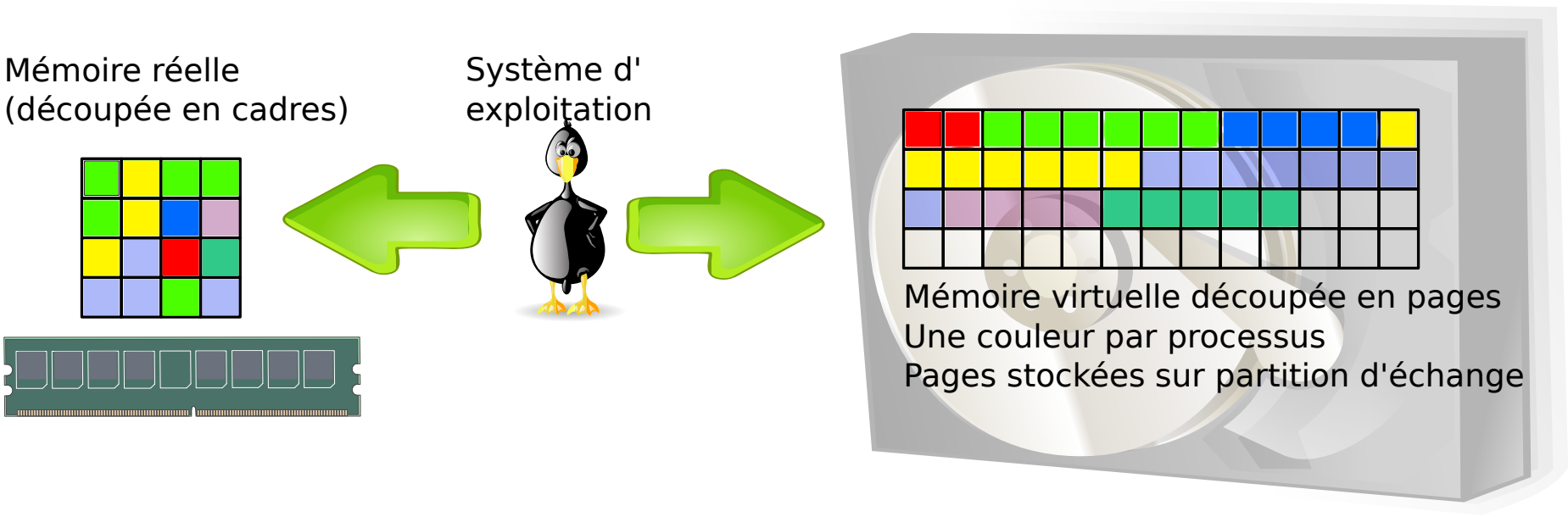


Mémoire virtuelle : implémentations

- Sous Unix (*swap*) :
  - Partition indépendante (mount)
  - et/ou fichier d'échange (swapon, swapoff)
  - Utilisables *à chaud*
  - Créés par mkswap, taille fixe
- Windows :
  - fichier d'échange pagefile.sys (taille variable)

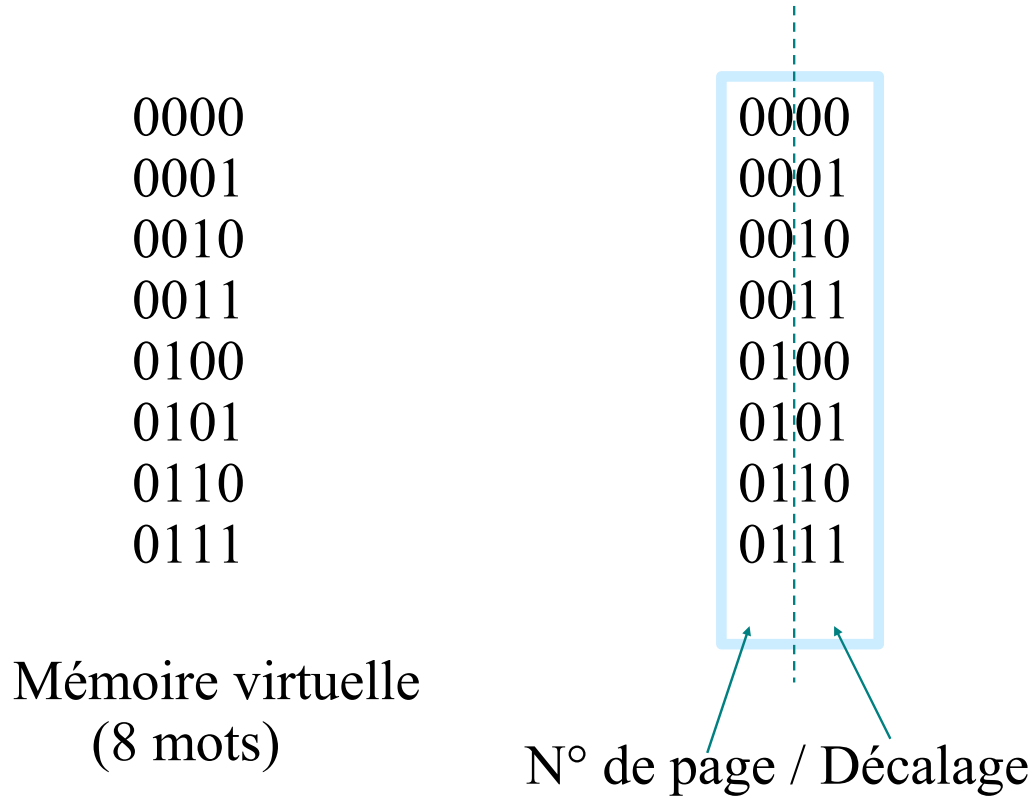
La pagination

- Mémoire *réelle* découpée en cadres
- Mémoire *virtuelle* découpée en pages
- Pour permettre le chargement : *taille des cadres = taille des pages*
- Page partiellement utilisée $\Rightarrow$ Fragmentation interne
- Accès à une zone de mémoire non chargé $\rightarrow$ *défaut de page* $\rightarrow$ *remplacement de page*



Les pages : de l'adresse au numéro de page

- Comment trouver le numéro d'une page en fonction d'une adresse mémoire ?
- Exemple suivant : mémoire initiale de 8 mots
- Découpage en pages de 4 (2^2) mots

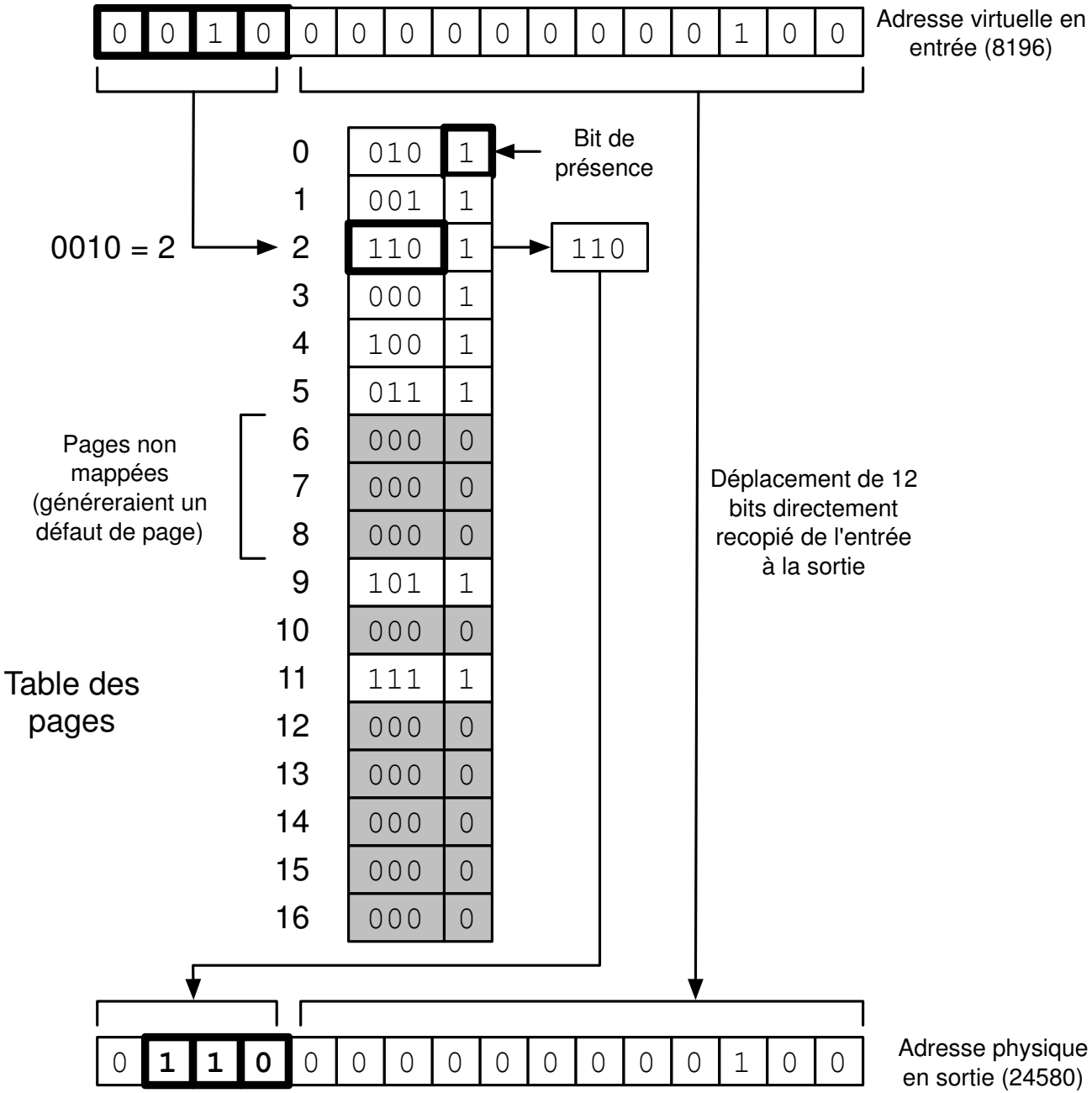


- À partir de maintenant, on ne s'occupe que des numéros de page/cadre.

La table des pages

- Une table des pages par processus.
- Une case par pages.
- La case de rang *page* contient :
  - Un bit indiquant la présence ou non de la page en mémoire.
  - Si oui, le numéro du *cadre* dans laquelle elle est chargée.
  - Un bit indiquant si la page a été modifiée ou non

La table des pages



Accélération matérielle de la pagination

- La MMU (Memory Management Unit)
  - Gestion de la mémoire paginée par un composant *matériel*
  - Plus rapide.
  - Transforme adresses virtuelles en adresses réelles.
  - Génère un *défaut de page* si la page demandée n'existe pas.
  - Présent sur la plupart des grosses architectures
  - Utilisation fréquente d'un TLB
- Le TLB (Translation Lookaside Buffer)
  - Cache limité à quelques entrées (64 typiquement. Sur Nehalem, 64 TLB1 + 512 TLB2).
  - Contient les numéro de cadre des pages les plus utilisées
  - Recherche *simultanée* sur toutes les entrées
  - Si on ne trouve pas dans le TLB, retour à la table des pages normale.

Algorithmes de remplacement de page

- En cas de défaut de page, on doit charger une nouvelle page

→ *Quelle page décharger pour faire de la place ?*

- Choix déterminant pour ne pas générer un écroulement sous les défauts de page !
- Plusieurs algos possibles...

Algo NFU (Not Frequently Used)

- On abandonne la page qui sert le moins en moyenne
- Coût très levé de mise à jour
- Quid de la page très demandée un moment et qu'on utilise plus ?

Algo LRU (Last Recently Used)

- On abandonne la page qui n'a pas servi depuis le plus longtemps
- Coût élevé de mise à jour
- Quid de la page demandée très régulièrement mais à des intervalles de temps assez grands ?

Algo de la seconde chance (forme de l'horloge)

- Les pages sont chargées dans une FIFO (les vieux d'abord !)
- Chaque fois qu'on accède à une page, on met son bit *R* à 1.
- Quand on cherche une page à décharger, on les examine une à une :
  - Si Bit *R* à 1 : on met *R* à 0 et on passe à la page suivante.
  - Sinon, on décharge la page, et on garde la position actuelle pour le prochain défaut de page.
- Algo de l'horloge : version avec une *liste circulaire* pour éviter test de retour de la FIFO.