Chapitre IV : La gestion de la mémoire

Eric.Leclercq@u-bourgogne.fr

Département IEM
http://ufrsciencestech.u-bourgogne.fr
http://ludique.u-bourgogne.fr/leclercq

January 17, 2017

Plan

La gestion des ressources

- Plusieurs dizaines de processus doivent se partager une mémoire commune : la RAM
- La capacité de la mémoire centrale est restreinte (même si elle peut sembler importante aujourd'hui)
- Si les processus sont nombreux, ils vont occuper plus de place que ne peut leur offrir la mémoire centrale
- La mémoire principale (RAM) et les registres sont les seuls éléments de stockage auxquels le processeur peut accéder directement

Quelques données quantitatives

- La mémoire secondaire (disque dur) est beaucoup plus importante (1To à 2To) que la mémoire centrale au moins un facteur 100
- Le SE propose un mécanisme de mémoire virtuelle
  - Le mécanisme de mémoire virtuelle consiste à utiliser le disque dur pour simuler de la mémoire réelle
  - Qu'en est il de la rapidité des accès aux éléments de la mémoire virtuelle ?
- Mais avant de proposer un tel mécanisme, le SE doit abstraire pour les uniformiser le fonctionnement et la gestion des différentes mémoires.

Quelques données quantitatives

Trois caractéristiques des mémoires sont importantes :

- Vitesse d'accès
- Capacité de stockage
- Coût

La capacité de stockage est d'autant plus réduite que la vitesse d'accès et les coûts sont élevés.

Types de mémoires

On distingue différentes catégories de mémoires

- **La mémoire volatile :** rapide, mais coût assez élevé
- Elle est considéré comme une ressource essentielle
- Pour qu'un programme soit exécuté il doit être placé en mémoire centrale RAM (donc volatile)

Exemples de mémoires volatiles : les registres du CPU

- Transfert des informations de la mémoire centrale vers les registres
- Réalise des opérations sur les registres
- Range les résultats dans la mémoire centrale
- Peu de registres, mais leur accès est très rapide

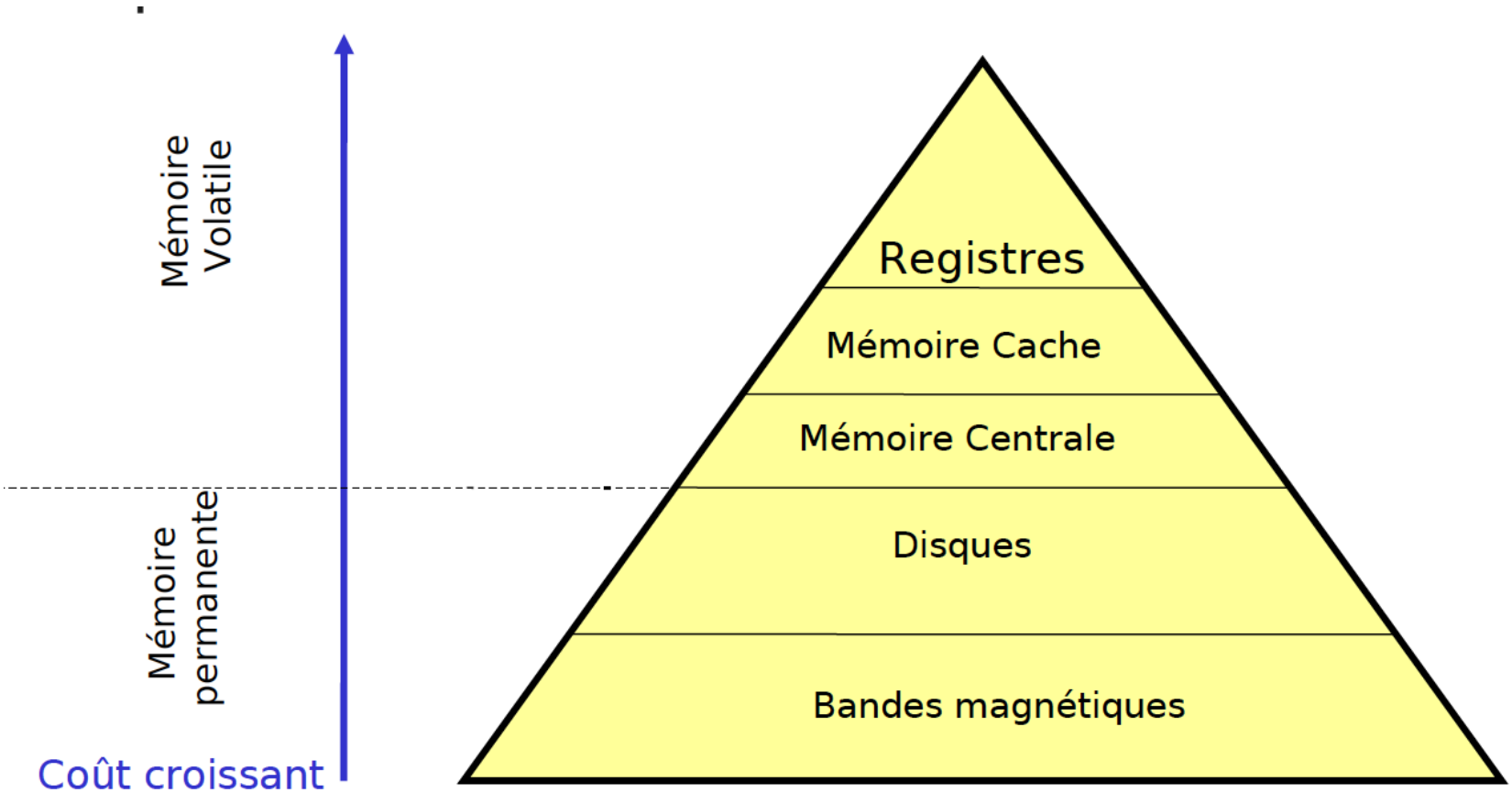
Types de mémoires

- La mémoire centrale et les registres du CPU ne travaillent pas à la même vitesse
- Pour servir d'intermédiaire on ajoute **une mémoire cache**
- Elle fonctionne comme un tampon (*buffer*) directement accessible par le processeur avec un temps d'accès très faible
- Les caches sont des mémoires associatives (cf TD) recherche en $O(1)$

Types de mémoires

- **Mémoire permanente :** supports magnétiques : disques, bandes, disques optiques
- Les temps d'accès sont beaucoup plus long
- Le coût est moins élevé
- On parle également de : mémoire de masse, mémoire secondaire ou auxiliaire

Types de mémoires



Coût croissant
Vitesse d'accès croissante
Capacité de stockage décroissante

La gestion de la mémoire dans le SE

Le mécanisme de gestion de la mémoire par le SE a pour objectifs :

- l'organisation de la mémoire en zones
- l'allocation, libération des zones
- la protection des processus
- la simulation d'une mémoire de grande taille (mémoire virtuelle sur disque)
- la transparence des accès (quelque soit le type de mémoire physique ou virtuelle)

Le SE utilise les notions de blocs, de pages et de zones.

Organisation de la mémoire

- L'accès doit être indépendant du support ;
- La mémoire est divisée en zones de taille fixe ou variable, il est impossible de la gérer octet par octet ;
- Une zone de taille N de la mémoire est un sous-ensemble des N mots (ayant des adresses consécutives) ;
- Le SE doit garantir l'intégrité de ces zones en interdisant les accès non autorisés ;
- Le contenu de certaines zones doit pouvoir être partagé (cf IPC ch3) ;

Organisation de la mémoire : stratégies simples

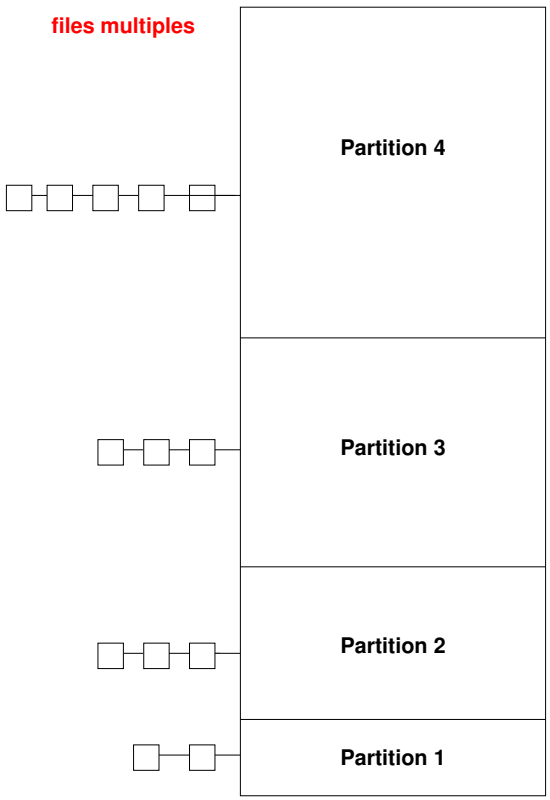
Mono-programmation :

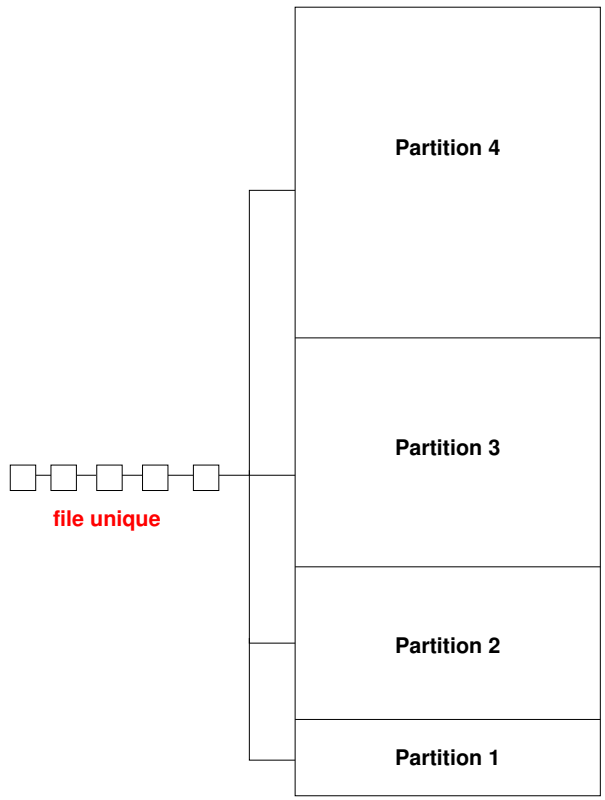
- Dans les systèmes mono-programme, un programme a accès à l'ensemble de la mémoire ;
- Il est possible de subdiviser la mémoire et de garder une partie du SE en RAM.

Multi-programmation : le SE contrôle différents processus en cours d'exécution

- Les processus peuvent être triés selon leurs besoin en mémoire (à déclarer au lancement ;
- Le SE propose k zones de différentes tailles mémoire ;
- Dans chaque zone, un seul processus est autorisé (**multi-programmation à partition fixe**) ;
- Le SE gère des files par zone ou une file globale (exemple OS/360 IBM).

Organisation de la mémoire : stratégies simples



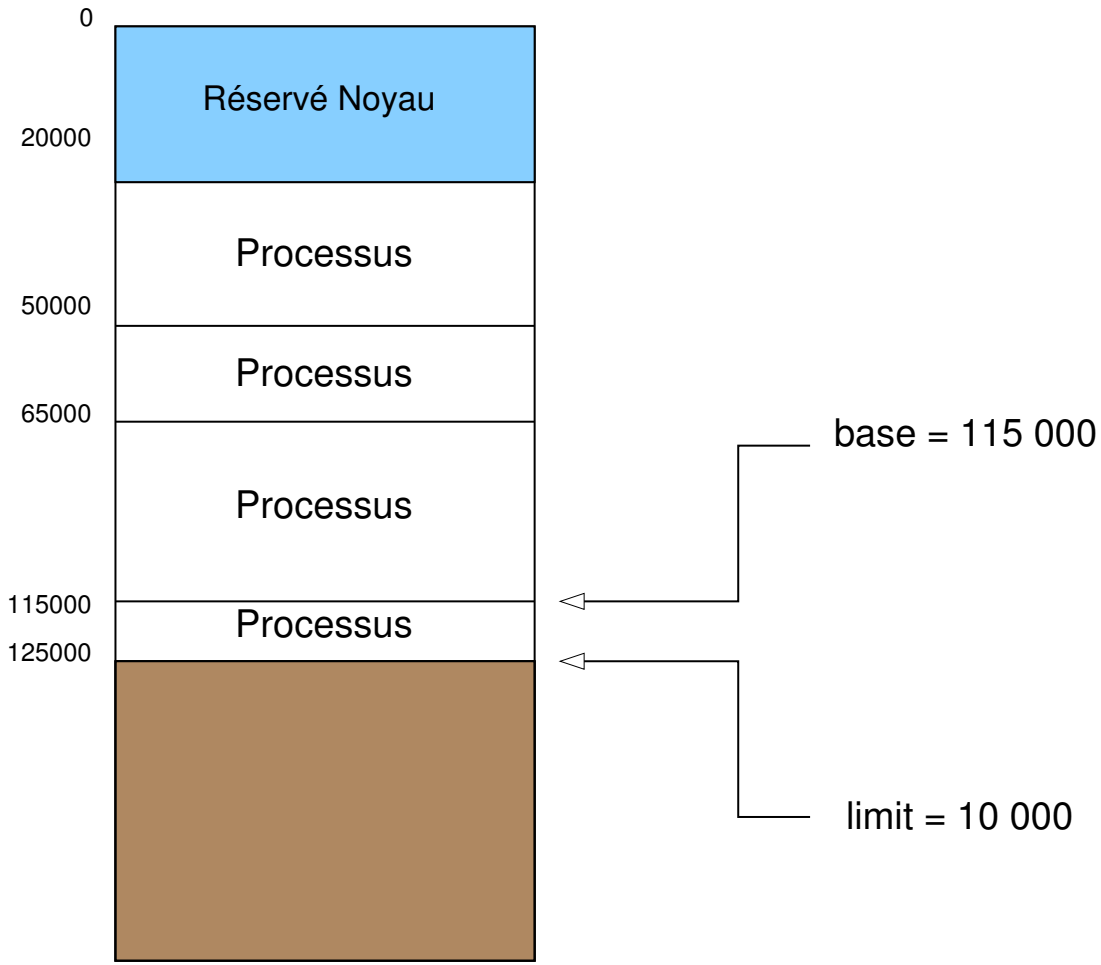


Stratégies d'allocation et de libération

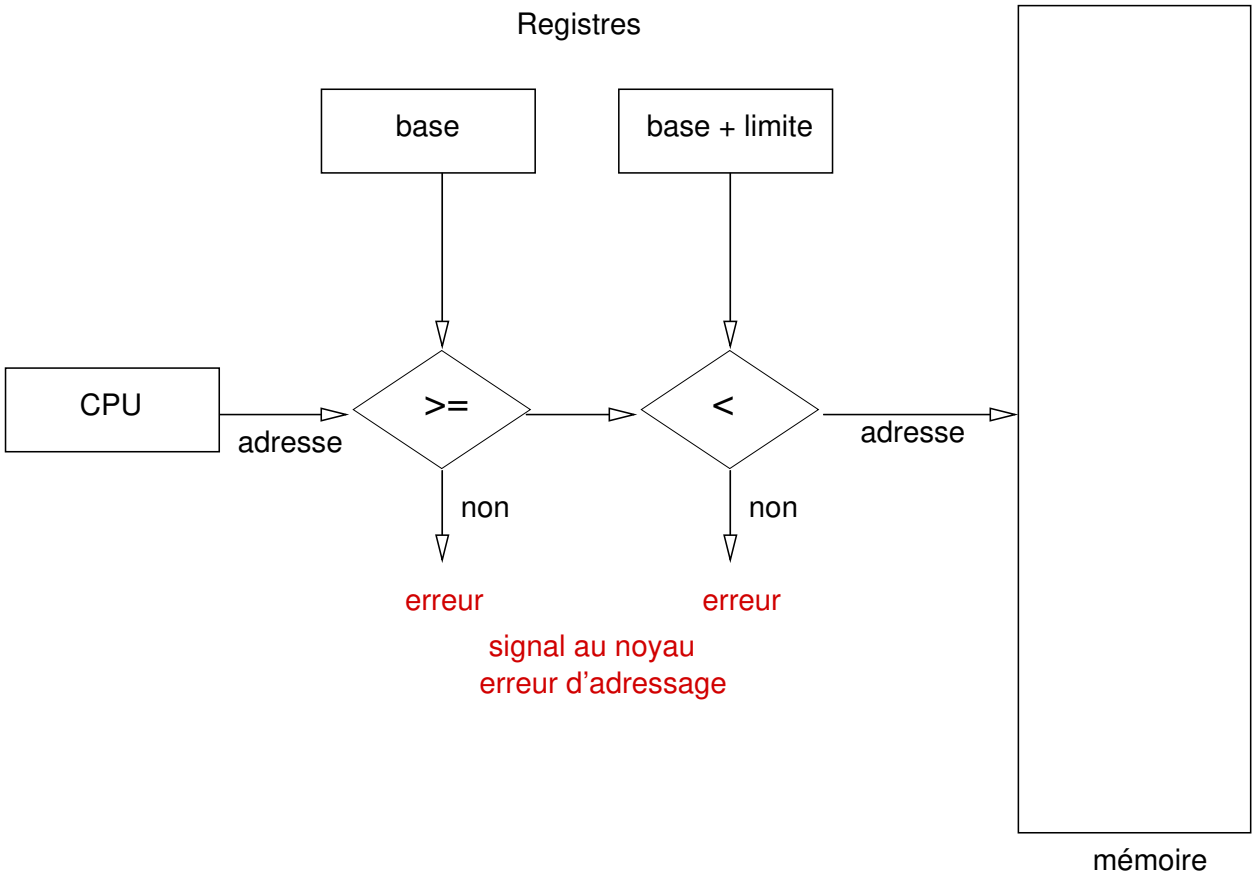
Les opérations associées à la mémoire sont :

- Opération sur les mots : lecture ou écriture
- Opérations sur les zones : allocations ou libération
- Dans certains cas on ajoute les opérations :
  - Extension ou diminution de zones
  - Division d'une zone (partitionnement)
- La gestion sur les mots (octets par exemple) est trop fine, celle sur les zones est trop grossière
- Le SE utilise principalement des blocs et des pages
- La MMU travaille à un niveau plus fin

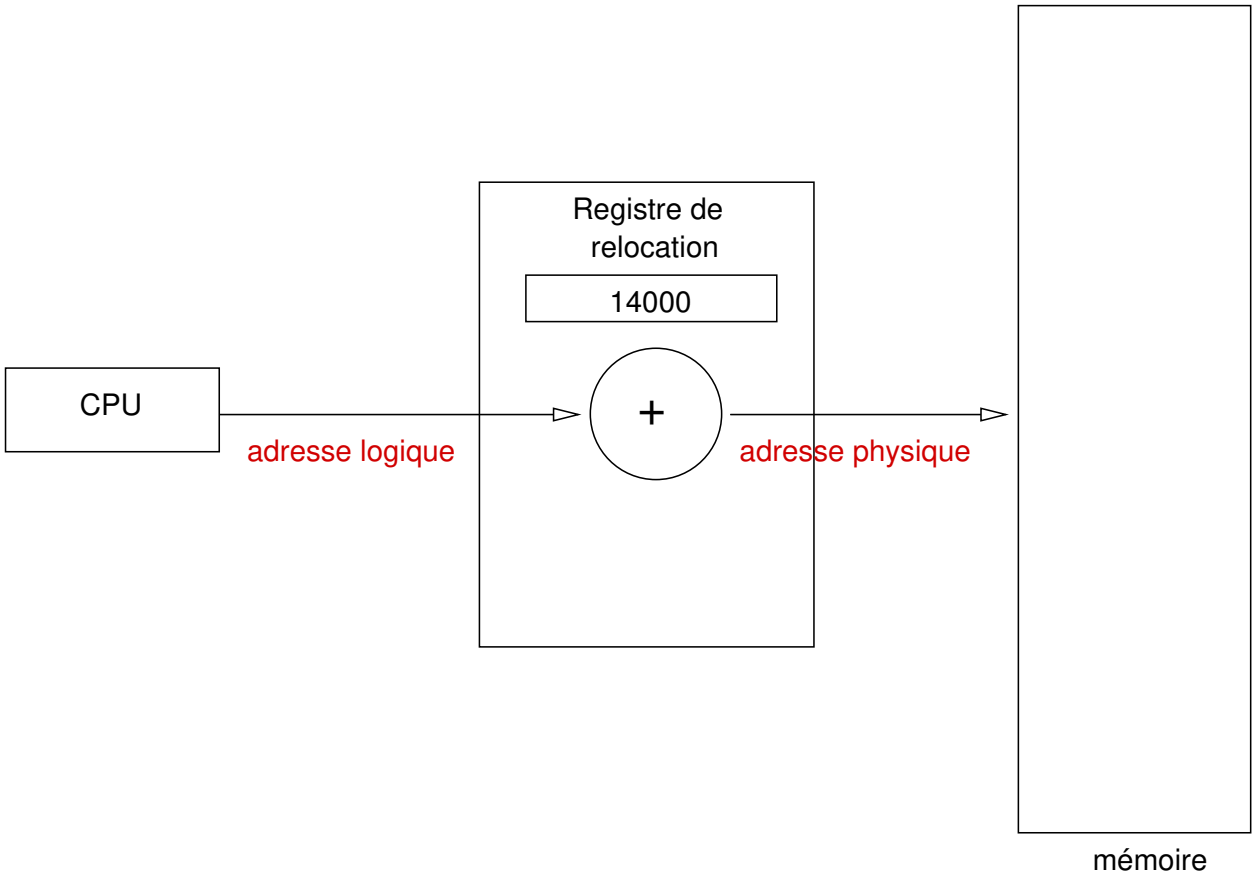
Memory Management Unit



Memory Management Unit



Memory Management Unit



Principe de l'allocation

Pour un espace donné on peut choisir deux modes d'allocation :

- **Allocation contiguë :** consiste à placer la totalité d'un programme à des adresses consécutives
- **Allocation non contiguë :** consiste à fractionner le programme et à placer les différents fragments à des adresses dispersées

On appelle **bloc** un groupe d'un nombre fixe de mots consécutifs (16,256,1024, 4096, etc.)

- La taille des blocs est fixée par les caractéristiques du matériel
- Les zones allouées comportent toujours un nombre entier de blocs

Allocation contiguë

L'espace mémoire est divisé en plusieurs zones de tailles variables qui sont soit :

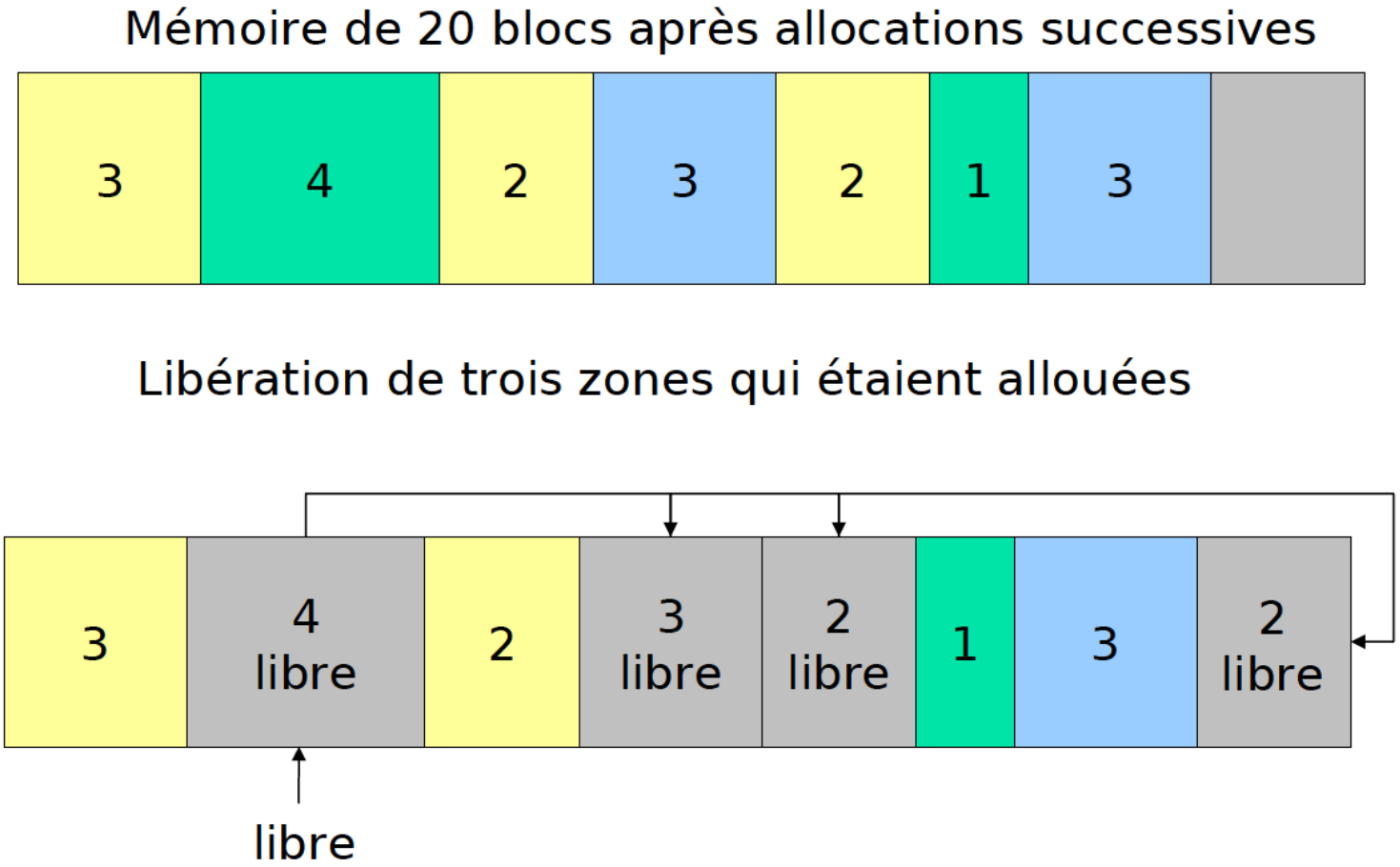
- Allouées pour un programme
- Vides (libre)

Le SE maintient en permanence la liste des zones vides (libres ou occupées) :

- soit avec une table de bits ;
- soit avec liste chaîne de zones (libres et occupées).

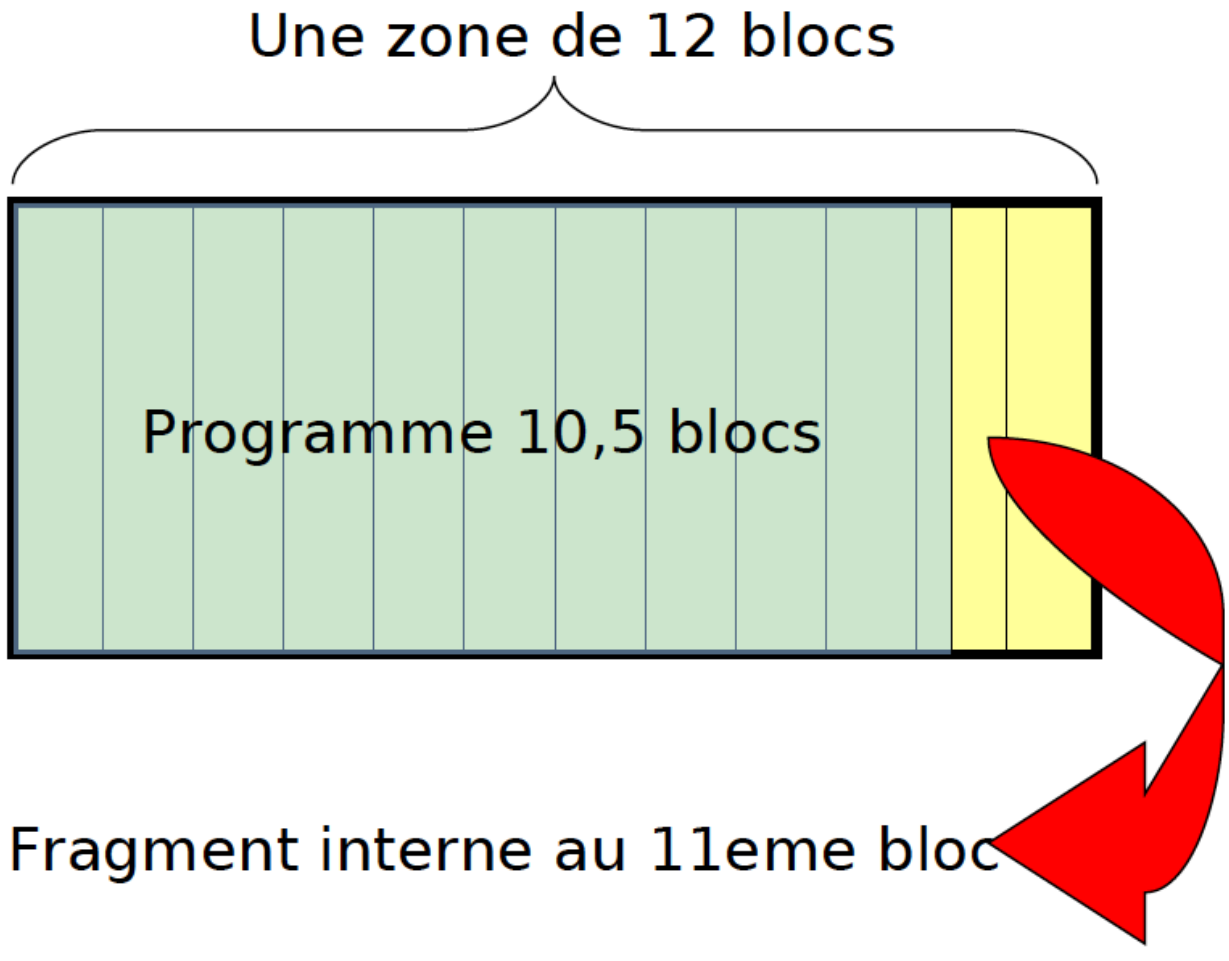
Quand une zone est allouée, elle est prélevée à l'extrémité d'une zone libre.

Allocation contiguë



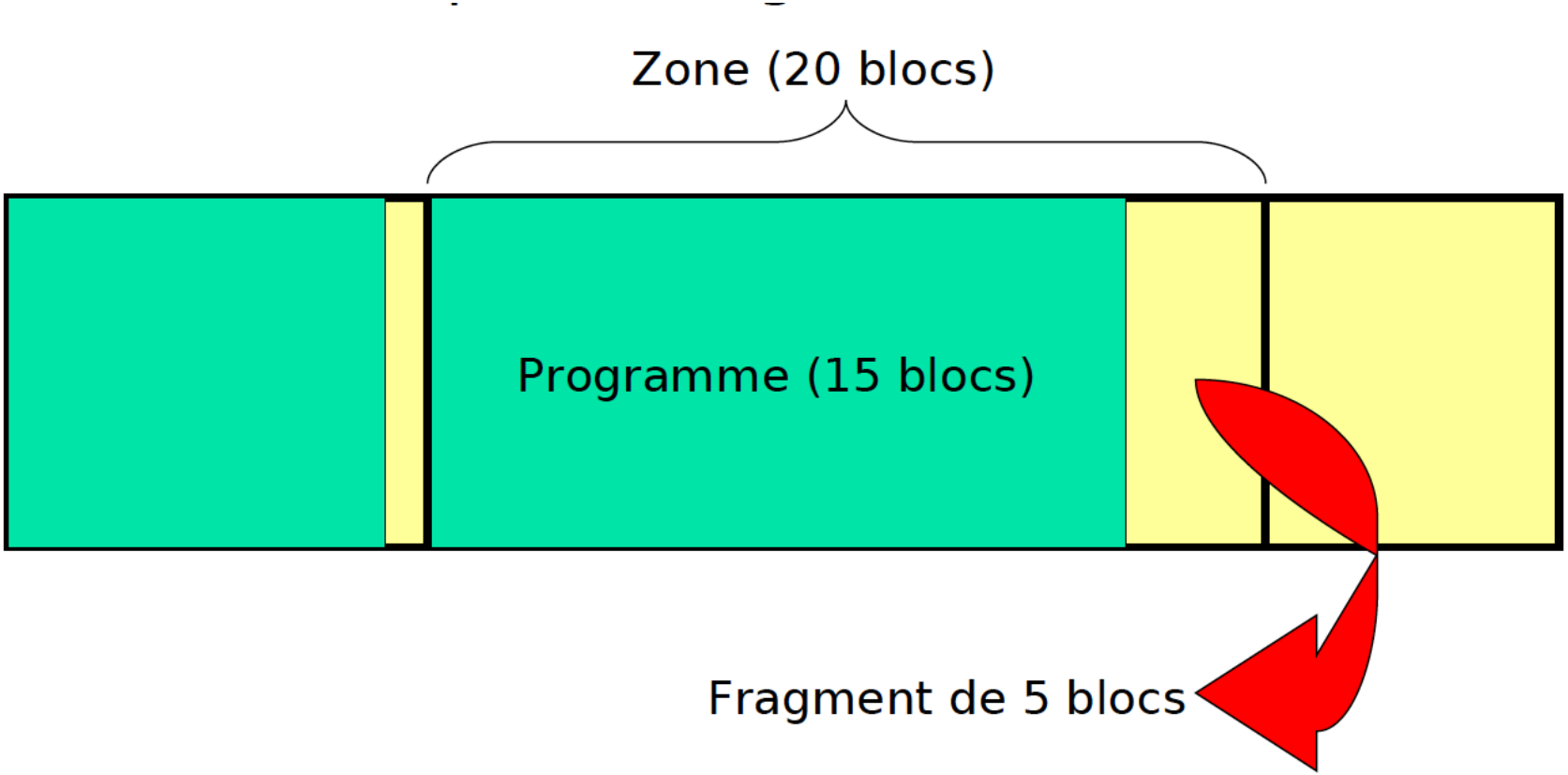
Notion de fragmentation

Fragmentation interne au bloc



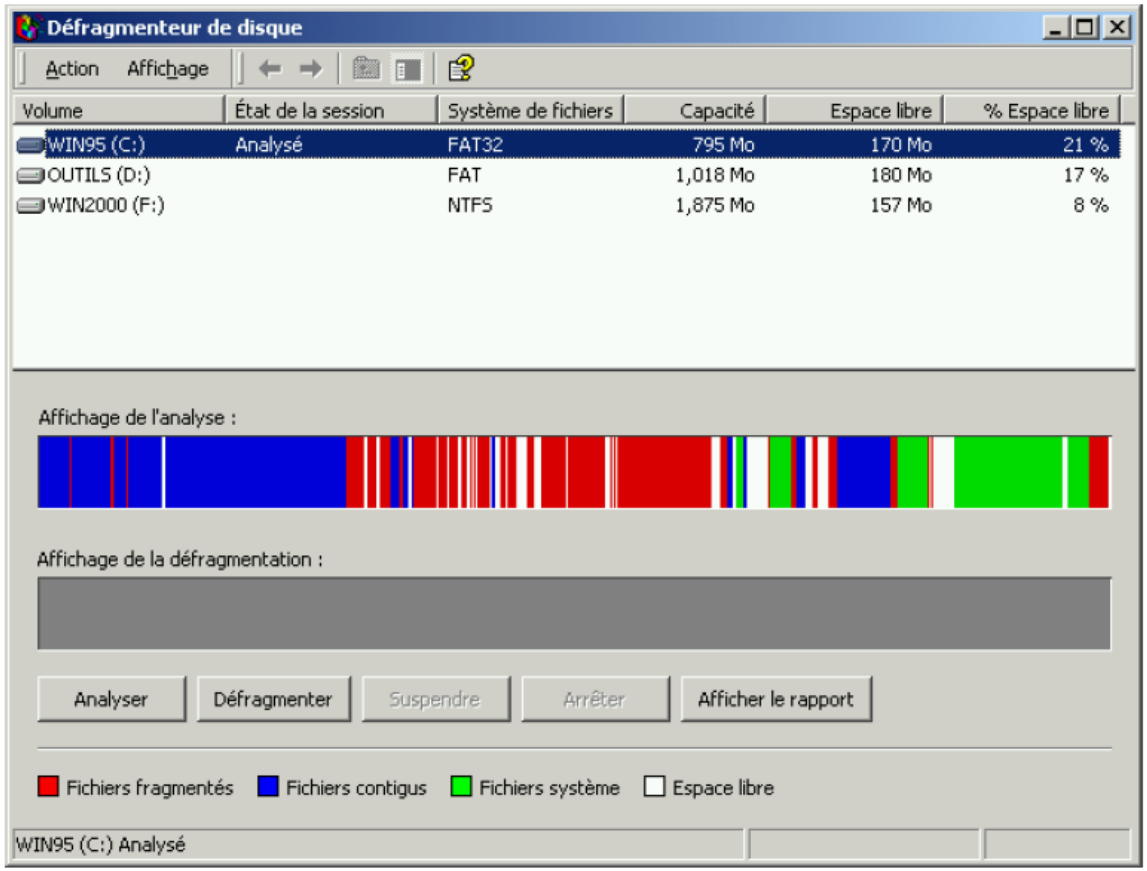
Notion de fragmentation

Fragmentation externe au bloc



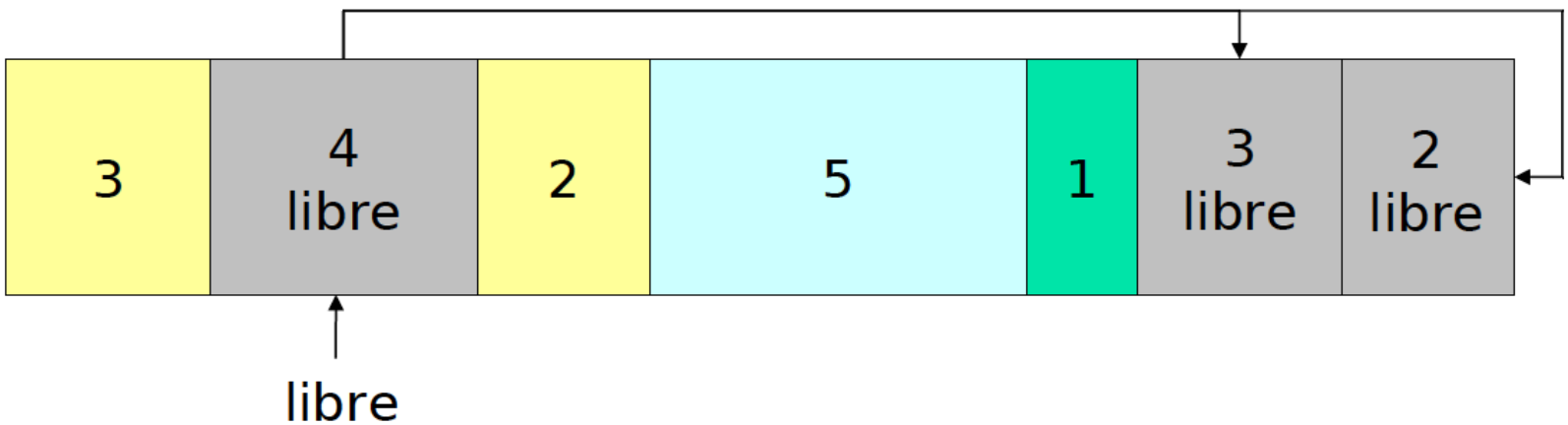
Fragmentation et fichiers

Ces deux types de fragmentations se retrouvent également sur les systèmes de fichiers (*file system*)



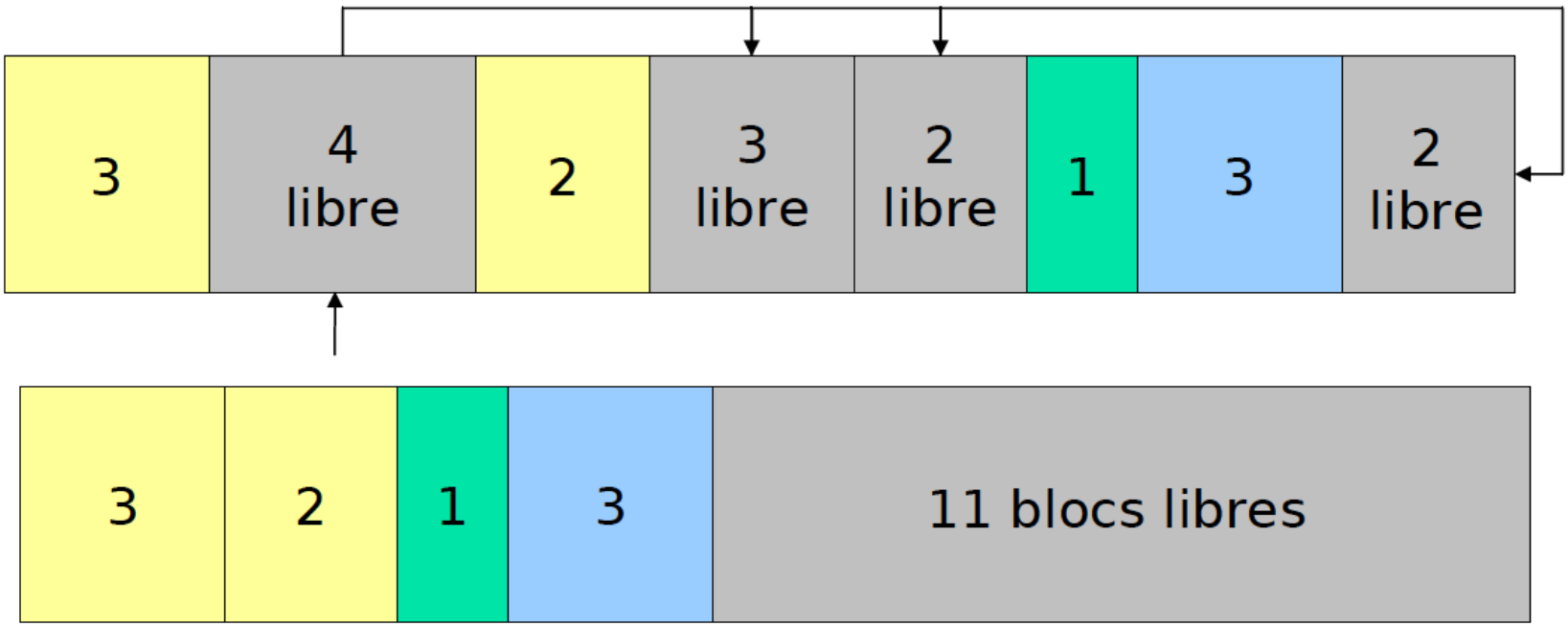
Allocation contiguë

Exemples et problèmes : demande de 5 blocs satisfaite en fusionnant 3+2



Allocation contiguë

Exemples et problèmes : demande de 11 blocs non satisfaite car il n'existe pas 11 blocs consécutifs => compactage



Allocation contiguë : compaction de la mémoire

Le compactage consiste à déplacer les zones allouées pour obtenir des zones libres contiguës Diverses stratégies :

- Conserver l'ordre
- Minimiser les déplacements

Le compactage a plusieurs inconvénients :

- Occupe l'unité centrale et diminue les performances (*overhead* = surcharge)
- Comment retrouver les zones déplacées ?

Allocation contiguë : stratégies de placement

Une stratégie de placement doit permettre de décider de l'endroit de la mémoire à allouer.

L'efficacité de différentes stratégies peut être mesurée au moyen des critères suivants :

- la rapidité du choix de l'emplacement
- La fragmentation externe qui en résulte (elle doit être minimale)
- L'utilisation globale de l'espace mémoire de la zone (elle doit être maximale)

Allocation contiguë : stratégies de placement

Les trois stratégies les plus utilisées sont :

- Première zone libre (first-fit) : exploration des zones, choix de la première de taille suffisante
- Le meilleur ajustement (best-fit) : la plus petite zone de taille suffisante (fragment minimal)
- Le plus grand résidu (worst-fit) : zone la plus grande choisie

Allocation non contiguë

Principe : fractionner un ensemble d'informations pour placer les différentes parties obtenues dans des blocs libres

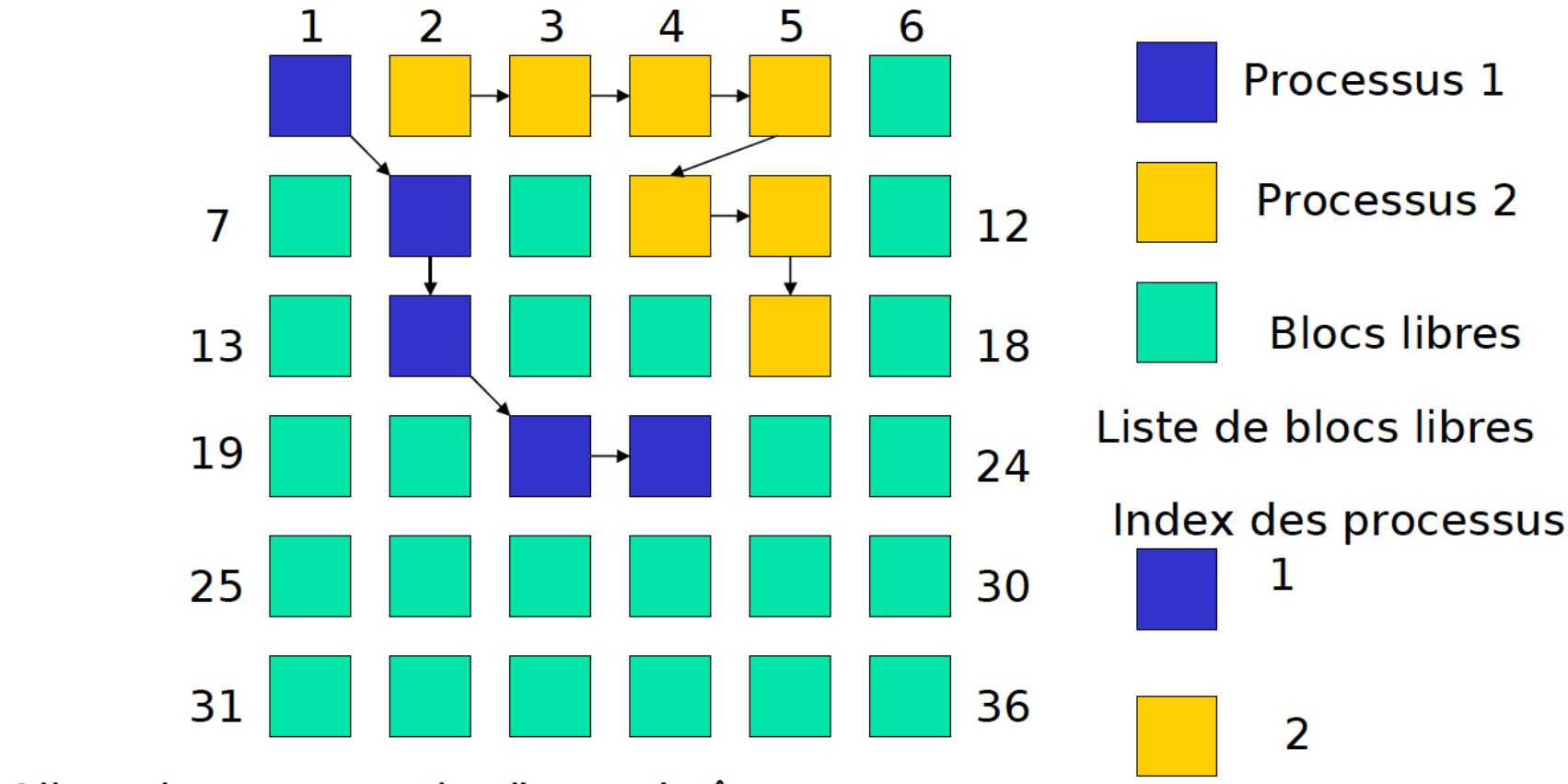
- Plus de fragmentation externe
- Évite une réorganisation de l'espace
- Pour retrouver l'ensemble des données stockées le SE doit conserver les informations d'allocation : chaînage et indexation

Allocation non contiguë : chaînage

Principe : chaque ensemble d'informations est composé d'une liste chaînées de blocs
La taille de l'ensemble peut varier par adjonction de blocs prélevés dans la liste des blocs libres
Inconvénients :

- Nécessite de parcourir la liste depuis le premier bloc
- Structure fragile : perte de lien dangereuse
- Perte d'espace dans chaque bloc du à la présence du lien

Allocation non-contiguë



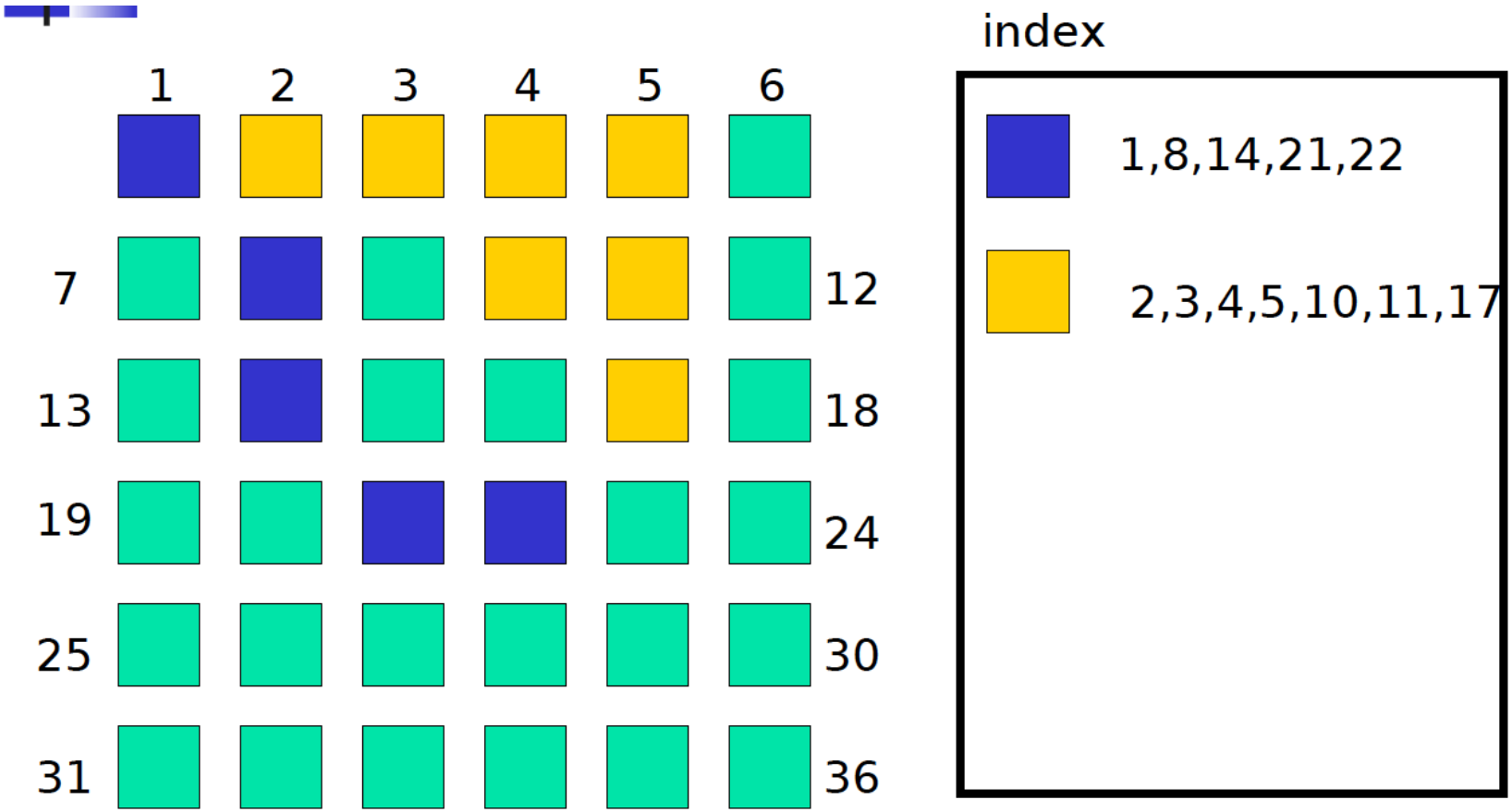
Allocation non contiguë par chaînage

Allocation non contiguë : indexation

Principe : conserver dans une table tous les numéros de blocs utilisés pour un ensemble (index)
On évite ainsi les liens
Accès direct à un bloc recherché en consultant l'index
Inconvénients :

- L'index peut être très grande
- Il est stocké lui aussi en mémoire (taille fixée)

Allocation non-contiguë



Allocation non contiguë par indexation

Principe général

Pour que la mémoire de la machine paraisse infinie (presque) on utilise la mémoire de masse

La simulation de mémoire repose sur un mécanisme double :

- un mécanisme d'adressage unique (ou uniforme) pour stocker les données en mémoire quelle soit réelle ou virtuelle
- un mécanisme d'échange (swap)

On peut développer deux hypothèses pour le swap :

- soit échanger le processus en entier de la mémoire réelle vers la mémoire virtuelle
- soit échanger seulement une partie de processus (solution retenue)

Problème de la gestion par blocs

La mémoire est gérée par blocs (quelques kilo-octets) et les blocs sont présents dans le SE en grand nombre :

- les blocs sont une unité de mémoire trop fine pour le mécanisme de swap
- solution : utiliser des groupes de blocs ou pages

Une page est un groupe de blocs consécutifs appartenant généralement au même processus. La machine dispose de n Mo de mémoire dont s sont simulés par le disque. Les processus peuvent adresser n Mo sans se soucier du type de mémoire (physique ou virtuelle).

Fonctionnement

La mémoire totale (physque+virtuelle) est découpée en pages de taille fixe.

Si l'adresse solicitée par un processus se trouve au delà de la limite de la mémoire physique c'est que le processus concerné possède des portion de code ou de données se trouvant sur la mémoire virtuelle : il faut aller chercher les données ou le code pour les mettre en mémoire centrale via le swap (échange)

le swap s'effectue en deux temps :

1. faire place : déchargement d'un certain nombre de pages de la mémoire centrale vers la mémoire de masse
2. chargement des pages du processus depuis le mémoire de virtuelle dans la mémoire centrale

Fonctionnement

Mise en fonction du swap :

- Le swapping s'effectue à la réception d'un signal de défaut de page, par exemple lorsque le processus demandé par l'ordonnanceur exécute du code qui ne se trouve pas en mémoire centrale
- La couche de SE chargée de gérer la mémoire centrale suspend l'exécution du programme
- Un signal est envoyé, puis transfert aux procédures de gestion d'interruption concernées
- Les procédures suspendent le processus, le font charger puis le débloquent

Mapping mémoire physique mémoire virtuelle

Mapping = correspondance adresse logique adresse réelle

- toutes les pages sont numérotées de 0 à n et notées P_i
- une adresse globale ou logique = un couple page, offset
- la mémoire physique contient $n - s$ emplacement de pages, numérotées de 0 à $n - s - 1$, notées EP_{mpi}
- une page peut se trouver en mémoire physique ou virtuelle donc un page P_i peut se trouver dans un des emplacements EP_{mpi}
- pour trouver l'adresse réelle correspondante à une adresse logique on consulte la table des pages (5,2) et page logique 5 en emplacement 1 donnera (1,2)

Stratégie pour décharger une page

- FIFO : on décharge la page la plus ancienne (Pb si on est dans une itération)
- LRU (least recently used)
- LFU (least frequently used)
- NRU (not recently used)