diafor
organisation

Gestion de la mémoire centrale

Gestion de la mémoire centrale

Plan de la phase

Introduction
Chargement d'un programme en mémoire centrale
Mémoire physique et mémoire logique
Allocation de la mémoire physique
Mémoire virtuelle
Conclusion
Exercices

Gestion de la mémoire centrale

Introduction

Dans le cadre d'un système multiprogrammé, plusieurs programmes sont chargés dans la mémoire centrale, qui doit donc être partagée entre ceux-ci.

Ce partage engendre trois problèmes :

- Il faut définir un espace d'adressage séparé pour chaque processus.
- Allouer de la place en mémoire centrale pour le code et les données.
- Protéger cet espace d'adressage vis-à-vis des accès des autres processus.

Par ailleurs, le nombre de processus nécessaires à une occupation optimale du processeur va très vite rendre insuffisante la quantité de mémoire physique disponible sur la machine.
Une gestion de la mémoire ne visant à ne charger que les parties utiles à un instant donné de chaque processus est donc mise en place pour palier ce problème : c'est le concept de ***mémoire virtuelle***.

Définition d'un processus :
Un processus est un programme en cours d'exécution auquel est associé un environnement processeur (CO, PSW, RSP (registre haut de pile), registres généraux) et un environnement mémoire (zone de code, de données et de pile) appelés ***contexte du processeur***.
Un processeur est l'instance dynamique d'un programme et incarne le fil d'exécution de celui-ci dans un ***espace d'adressage protégé***.

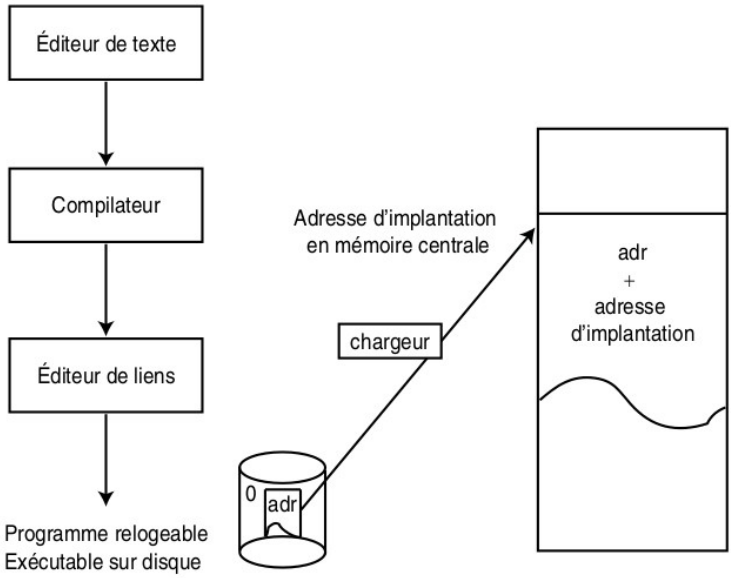
Gestion de la mémoire centrale

Chargement d'un programme en mémoire centrale

Le chargement constitue la dernière étape de la chaîne de production de programmes et met en œuvre l'outil *chargeur*. Le chargeur est appelé lorsque l'utilisateur souhaite exécuter son programme, il copie alors le programme exécutable depuis le disque dur vers la mémoire centrale.

Le fichier exécutable stocké sur le disque est qualifié de fichier ***relogeable*** : toutes les adresses des instructions est des données dans ce code exécutable sont calculées à partir de 0, i.e que le premier octet du code exécutable à une adresse égale à 0.

Lorsque le chargeur copie le code exécutable depuis le disque vers la mémoire centrale, il implante le code dans un espace libre de la mémoire centrale, dont le premier octet n'a pas forcement l'adresse 0, mais une adresse quelconque appelée ***adresse d'implantation mémoire***.



Toutes les adresses `adr` calculées dans le programme doivent être modifiées pour tenir compte de cette adresse d'implantation mémoire : c'est l'opération de translation des adresses qui consiste à ajouter à chaque `adr` apparaissant dans le programme exécutable du disque, la valeur de l'adresse d'implantation mémoire.

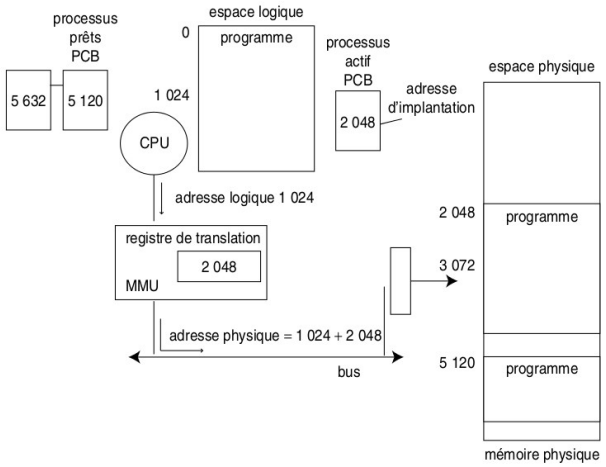
- Faire démo `/diapos4/note.txt`
- Lire note sur chargement et édition de liens dynamique/statique

Gestion de la mémoire centrale

Mémoire physique et mémoire logique

L'ensemble des adresses du programme exécutable, générées par le processeur au moment de l'exécution des instructions, est appelé ***espace d'adresses logiques ou virtuelles***.
L'ensemble des adresses physiques réellement occupées par le programme exécutable suite au chargement en mémoire centrale est appelé ***espace d'adresses physiques***.

Pour que le processeur puisse accéder à la mémoire centrale, il est nécessaire de *convertir* les adresses logiques générées par le processeur en adresse physiques.
Cette conversion est réalisée par un dispositif matériel, la MMU (*Memory Management Unit*).



Sur la figure, la MMU contient le registre de translation. Ce registre est toujours chargé avec l'adresse d'implantation du programme exécutable correspondant au processus actif.

Nous verrons qu'il existe d'autres schémas de conversion, notamment les schémas liés à la segmentation et à la pagination.

Note : En même temps que le chargement en mémoire centrale du programme exécutable, le système d'exploitation crée une structure de description du processus : le PCB *(Process Control Block*), qui contient les informations suivantes : PID, l'état courant du processus (élu, prêt, bloqué) **(note 1)**, le contexte du processus : la valeur du CO, la valeur des autres registres du processeur... Le PCB permet la sauvegarde et restauration du contexte mémoire et du contexte processeur lors des opérations de commutations de contexte.

Gestion de la mémoire centrale

Allocation de la mémoire physique

L'attribution de place mémoire aux programmes utilisateurs et la libération de celle-ci en fin d'exécution des processus s'appuient sur différentes stratégies d'allocation et de libération mémoire qui peuvent être divisées en deux grandes familles :

• Un programme est un ensemble de *mot contigus insécable*. L'espace d'adressage du processus est linéaire. On trouve ici les méthodes d'allocations en ***partitions variables***.

• Un programme est un ensemble de mots contigus sécable, i.e que le programme peut être divisé en plus petits morceaux, chaque morceaux étant lui-même un ensemble de mots contigus. Chaque morceaux peut alors être alloué de manière indépendante. On trouve ici les mécanisme de ***segmentation*** et de ***pagination***.

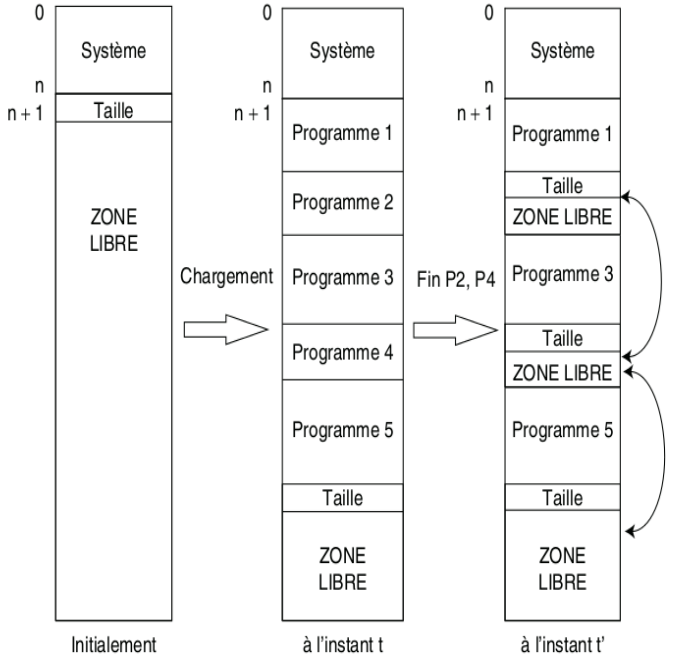
Gestion de la mémoire centrale

Allocation de la mémoire physique – allocation contiguë

Notion de partitions variable : le programme est considéré comme un espace d'adressage insécable. La mémoire physique est découpée en zones disjointes de taille variable, adaptables à la taille des programmes, qui sont dénommées *partitions*.

- Initialement, la zone réservée aux programmes est vide et constitue une unique *zone libre*.
- Au fur et à mesure des chargement de programmes, la zone libre se réduit et à l'instant t, elle n'occupe qu'une fraction de la mémoire centrale.
- Lorsque les exécution des processus se terminent, la mémoire est libérée : il se crée alors pour chaque zone libérée, une nouvelle zone libre.

Chaque zone libre (*ZL*) est caractérisée par une adresse d'implantation en mémoire centrale et une taille en octet.

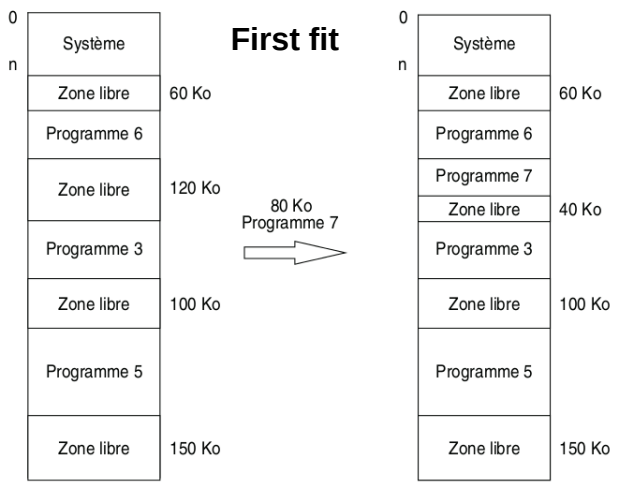


Gestion de la mémoire centrale

Allocation de la mémoire physique – allocation contiguë

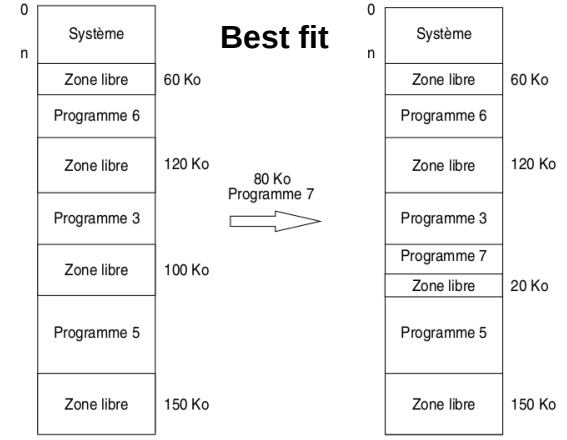
Méthode d'allocations d'une zone libre

Charger un nouveau programme consiste à trouver une zone libre suffisamment grande pour pourvoir y placer ce programme, tout en minimisant la place perdue en mémoire centrale et le temps de choix de la zone libre. Trois stratégie existent pour le choix de la zone libre :



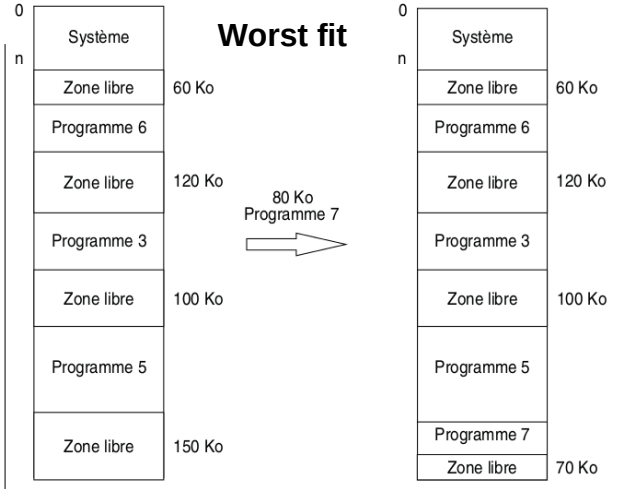
Stratégie de la première zone libre :
On prend la première zone libre suffisamment grande trouvée,
i.e la première zone libre ZL telle que taille(ZL) ≥ taille(programme à charger)

Avantage : rapidité du choix d'une zone.
Inconvénient : ne conduit pas à la meilleure utilisation possible de la mémoire centrale.



Stratégie de la meilleure zone libre :
On choisit la zone libre dont la taille est proche de celle du programme à allouer.

Avantage : optimise l'utilisation de la mémoire centrale.
Inconvénient : crée des zones libres résiduelle de tailles de plus en plus petit, qui ne seront peut-être pas utilisable pour une nouvelle allocation.
De plus moins rapide dans le choix de la zone libre (examen de l'ensemble des zones libres)



Stratégie de la plus mauvaise zone libre :

On prend la zone libre dont la taille est la plus éloignée de celle du programme à allouer.

L'avantage par rapport à la stratégie précédente est des zones résiduelles créées suite à une allocation, sont davantage utilisable pour de nouvelles allocations.

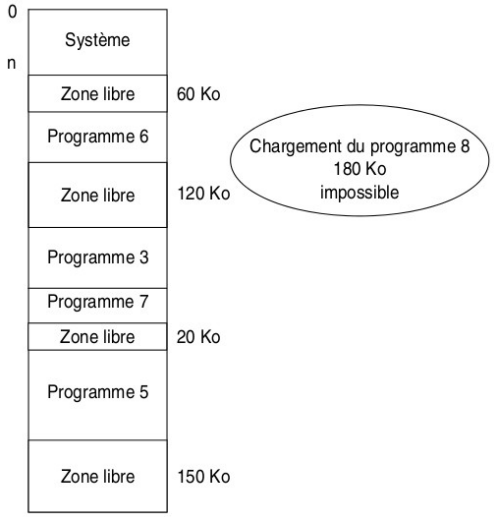
Gestion de la mémoire centrale

Allocation de la mémoire physique – allocation contiguë

Fragmentation externe

Au fur et à mesure des opérations d'allocation et de désallocation, la mémoire centrale devient composée d'un ensemble de zones occupées et de zones libres éparpillées.

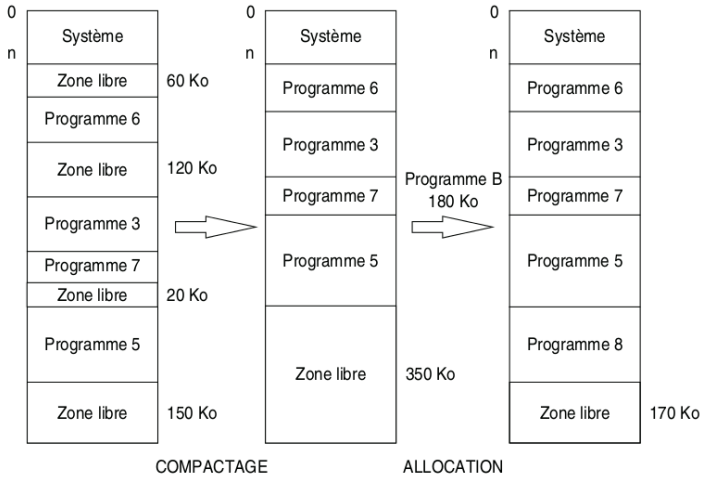
Ces zones libres peuvent être chacune trop petites pour permettre l'allocation de nouveaux programme, alors que la somme totale de l'espace libre en mémoire centrale est suffisant : on parle alors de *fragmentation externe* de la mémoire centrale.



Compactage de la mémoire centrale

Une solution au problème de la fragmentation externe est apportée par l'opération de *compactage de mémoire centrale*.

Ceci consiste à déplacer les programmes en mémoires centrale de manière à ne créer qu'une seule et unique zone libre. C'est une opération coûteuse.

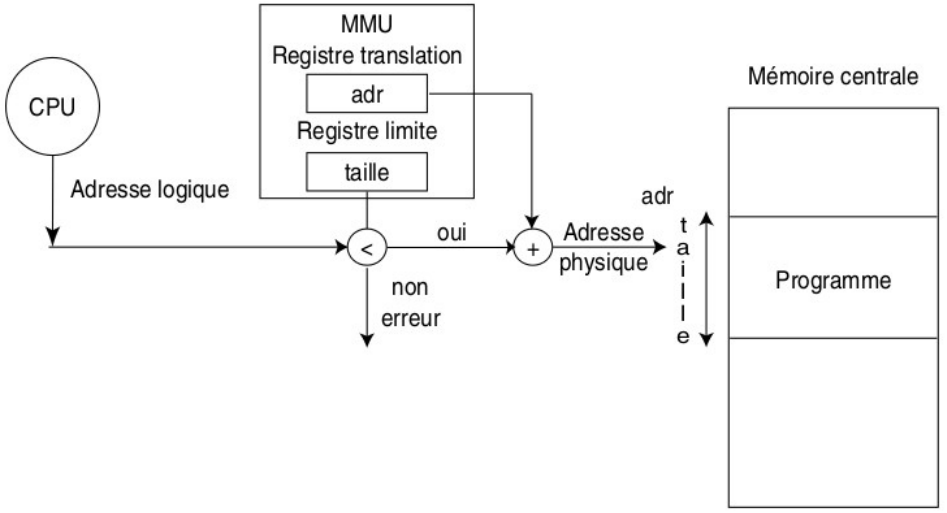


Gestion de la mémoire centrale

Allocation de la mémoire physique – allocation contiguë

Protection des partitions

Chaque partition allouée à un programme constitue un espace d'adressage protégé pour le processus correspondant.



Cette protection est réalisée en vérifiant que chaque adresse logique générée par le processeur pour le compte d'un processus est bien inférieure strictement à la taille de la partition allouée au processus concerné.

La taille limite de la partition allouée à un processus fait partie du contexte du processus au même titre que la valeur de l'adresse d'implantation `adr` en mémoire centrale de cette partition.
Les valeurs de ces grandeurs pour le processus actif sont chargées dans des registres de la MMU.

Gestion de la mémoire centrale

Allocation de la mémoire physique – allocation non contiguë

La méthode d'allocation par partitions variables présente deux difficultés :

• Comme le programme à charger en mémoire centrale est considéré comme un ensemble de mots insécable, elle nécessite de trouver en mémoire centrale une zone libre d'un seul tenant suffisammentgrande pour y placer le programme.
• L'allocation et les désallocation successive engendre une fragmentation externe de la mémoire centrale, nécessitant une opération de compactage de la mémoire centrale excessivement coûteuse.

Une solution est de considérer que l'ensemble des mots constituant un programme est maintenant un ensemble sécable.
Ainsi, le programme à charger est divisé en un ensemble de morceaux, chaque morceau étant lui-même un ensemble de mots contigus insécable.
Chaque morceau du programme est alors alloué en mémoire centrale indépendamment des autres :

• Si le programme est divisé en un ensemble de morceaux de taille fixe et égale, le mécanisme d'allocation de la mémoire centrale est alors celui de la ***pagination***.
Chaque morceau est appelé ***page***.

• Si le programme est divisé en un ensemble de morceaux de taille variable, le mécanisme d'allocation de la mémoire centrale est alors celui de la ***segmentation***.
Chaque morceau est appelé ***segment***.

Gestion de la mémoire centrale

Allocation de la mémoire physique – allocation non contiguë
Pagination

Principe de la pagination

Dans le mécanisme de la pagination, l'espace d'adressage du programme est découpé en morceaux linéaires de même taille appelés *pages*. L'espace de la mémoire physique est lui-même découpé en morceaux linéaires de même taille appelés *case* ou *cadre*. La taille d'une case est égale à la taille d'une page. Cette taille est définie par le matériel, comme étant une puissance de 2, variant selon les OS entre 512 octets et 8192 octets. (cf taillePage.c)

Charger un programme en mémoire centrale consiste à placer les pages dans n'importe qu'elle case disponible.
Le système maintient une table appelée *table des cadres de pages* ou *table des cases* qui indique pour chaque case de la mémoire physique, si la case est libre ou occupée, et si elle est occupée, quelle page et quel processus la possèdent.

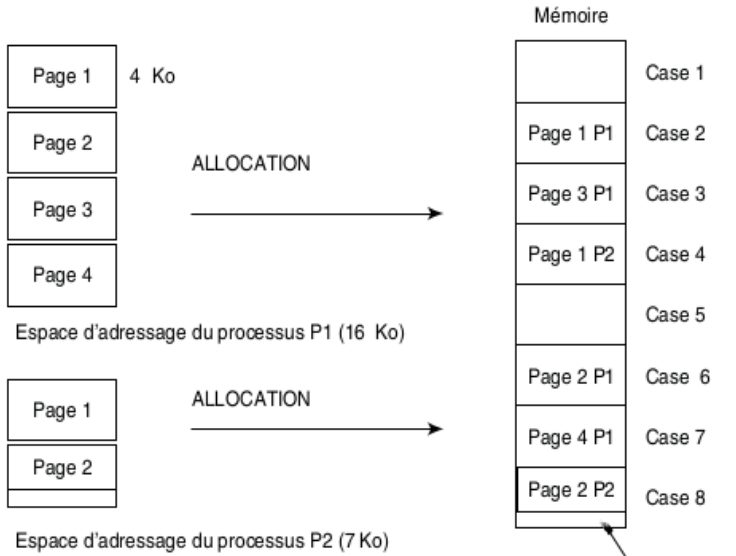
Ce mécanisme d 'allocation n'engendre pas de fragmentation externe. Il peut provoquer une fragmentation interne dans la mesure ou la taille du programme à allouer n'est pas forcément un multiple de la taille des pages.

Gestion de la mémoire centrale

Allocation de la mémoire physique – allocation non contiguë
Pagination

Principe de la pagination

La figure donne un exemple d'application d'allocation pour deux processus P1 et P2 dont les espaces d'adressage sont respectivement égaux à 16 Ko et 7 Ko. Les pages et les cases ont-elles une taille de 4 Ko.



	Page	Processus
Case 1	– 1 (libre)	
Case 2	1	P1
Case 3	3	P1
Case 4	1	P2
Case 5	– 1 (libre)	
Case 6	2	P1
Case 7	4	P1
Case 8	2	P2

Table des cases

Gestion de la mémoire centrale

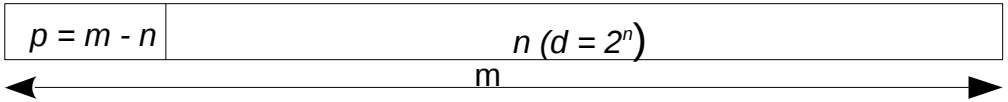
Allocation de la mémoire physique – allocation non contiguë
Pagination

Adresse logique et et table des pages

Conversion adresse logique – adresse physique

L'espace d'adressage du processus étant découpé en pages, les adresses générées dans cet espace d'adressage sont des adresses paginées, i.e qu'un octet est repéré par son emplacement relativement au début de la page à laquelle il appartient.

L'adresse d'un **octet = < numéro de page *p* à laquelle appartient l'octet, déplacement *d* relativement au début de la page p>**.



Pour une adresse logique de *m* bits, en considérant des pages de 2^n octets, les *m-n* premiers bits correspondent au numéro de page *p* et les *n* bits restants au déplacement d dans la page.
Les octets dans la mémoire physique ne peuvent être adressés au niveau matériel que par leur adresse physique. Pour toute opération concernant la mémoire, il faut donc convertir l'adresse paginée générée au niveau du processeur en une adresse physique équivalente.

L'adresse physique d'un octet s'obtient à partir de son adresse logique en remplaçant le numéro de page *p* par l'adresse physique d'implantation de la case contenant la page p et en ajoutant à cette adresse physique le déplacement *d* de l'octet dans la page. C'est la MMU qui effectue cette conversion.

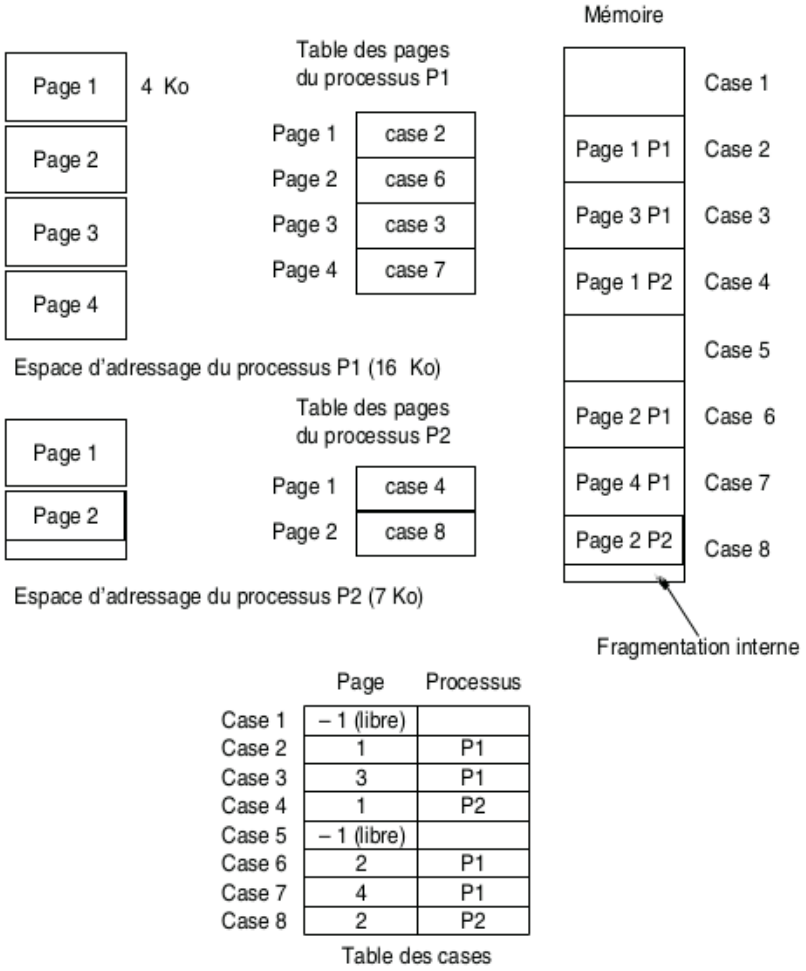
Pour toute page, il faut connaître dans quelle case de la mémoire centrale celle-ci a été placée. Cette correspondance s'effectue grâce à une structure particulière appelée la ***table de pages***.

Gestion de la mémoire centrale

Allocation de la mémoire physique – allocation non contiguë
Pagination

La table des pages

Dans une première approche, la table des pages :
• Contient autant d'entrées que de pages dans l'espace d'adressage d'un processus.
• Chaque processus a sa propre table des pages.
• Chaque entrée de la table est un couple **<numéro de page, numéro de case physique dans laquelle la page est chargée>**.



	Page	Processus
Case 1	– 1 (libre)	
Case 2	1	P1
Case 3	3	P1
Case 4	1	P2
Case 5	– 1 (libre)	
Case 6	2	P1
Case 7	4	P1
Case 8	2	P2

Table des cases

Gestion de la mémoire centrale

Allocation de la mémoire physique – allocation non contiguë
Pagination

La table des pages

Puisque chaque processus dispose de sa propre table des pages, chaque opération de commutation de contexte se traduit également par un changement de table des pages, de manière à ce que la table active corresponde à celle du processus élu.

Deux approche existent pour la réalisation de la table des pages :
• La table des pages est une structure matérielle réalisée grâce à des registres de la MMU. L'ensemble des registres composant la table des pages est sauvegardé avec le contexte processeur dans le PCB du processus ;
• La table des pages est une structure logicielle placée en mémoire centrale. L'adresse en mémoire centrale de la table des pages du processus actif est placée dans un registre de la MMU, le PTBR (*page-table base register*). Chaque processus sauvegarde dans son PCB la valeur de PTBR correspondant à sa table.

Dans la première approche, accéder à un emplacement mémoire nécessite seulement un accès à la mémoire, celui nécessaire à la lecture ou l'écriture de l'octet recherché puisque la table des page est stockée dans les registres du processeur. Cependant, cette solution ne peut convenir que pour de petites tables des pages, n'offrant par exemple pas plus de 256 entrées.

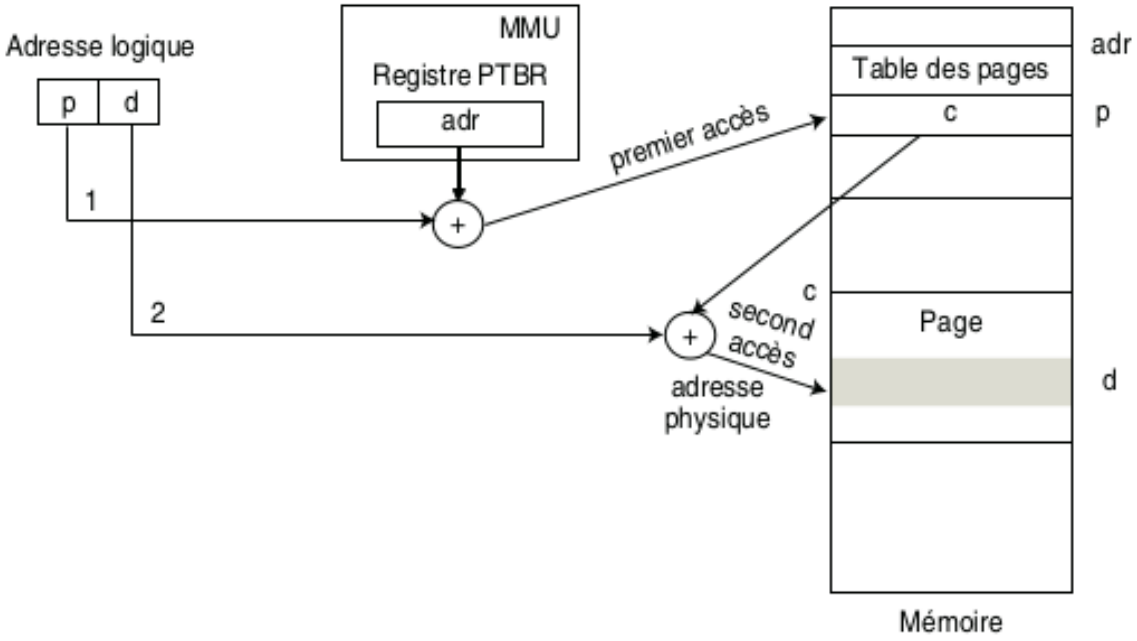
La deuxième approche permet de réaliser des tables de très grande taille. Cependant accéder à un emplacement mémoire à partir d'une adresse paginée <*p,d*> nécessite maintenant deux accès à la mémoire.

Gestion de la mémoire centrale

Allocation de la mémoire physique – allocation non contiguë
Pagination

La table des pages

- Un premier accès permet de lire l'entrée de la table des pages correspondant à la page *p* cherchée et délivre une adresse physique c de case dans la mémoire centrale;
- Un second accès est nécessaire à la lecture ou l'écriture de l'octet recherché à l'adresse *c* + *d*.



--KOMO-- A faire cache associatif ?? --KOMO--

Gestion de la mémoire centrale

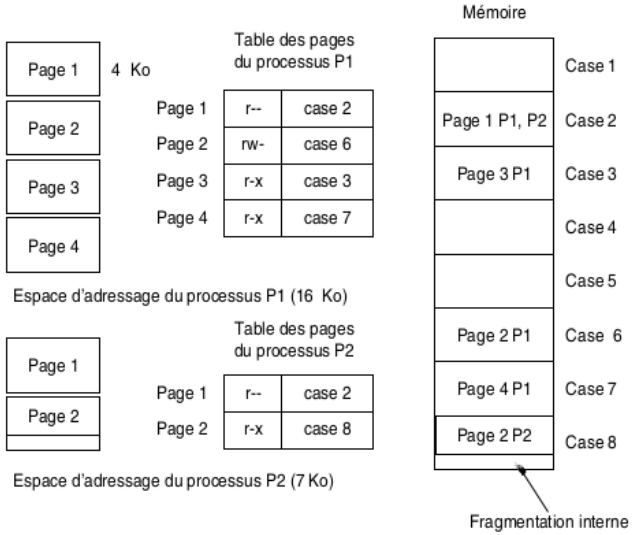
Allocation de la mémoire physique – allocation non contiguë
Pagination

Protection de l'espace d'adressage des processus

• Des bits de protections sont associés à chaque page de l'espace d'adressage du processus et permettent ainsi de définir le type d'accès autorisés à la page.
•Ces bits de protection sont mémorisés pour chaque page, dans la table des pages du processus.

• Sur la figure la page du processus P1 est définie avec le seul accès en lecture (r--) autorisé.
• La demande d'exécution d'une instruction du type `STORE D R1 page1,` d demandant l'écriture du contenu du registre R1 dans la cellule située à l'emplacement d de la page 1 provoque une trappe (soulève une exception) et l'arrêt de l'exécution du processus P1.

• Pour que deux processus puissent partager un ensemble de pages, il faut que chacun d'eux référence cet ensemble de pages dans sa table des pages respective. Ainsi, sur la figure, les processus P1 et P2 référencent tous les deux la même page 1, située dans la case 2 de la mémoire centrale.



	Page	Processus
Case 1	– 1 (libre)	
Case 2	1	P1, P2
Case 3	3	P1
Case 4	– 1 (libre)	
Case 5	– 1 (libre)	
Case 6	2	P1
Case 7	4	P1
Case 8	2	P2

Table des cases

--KOMO-- A faire pagination multiniveaux ?? --KOMO--

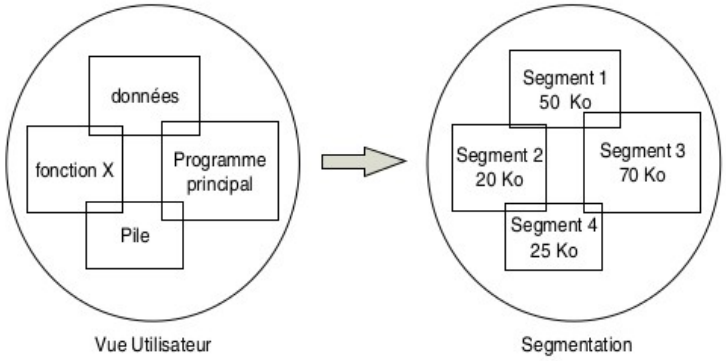
Gestion de la mémoire centrale

Allocation de la mémoire physique – allocation non contiguë
Segmentation

• La pagination constitue un découpage de l'espace d'adressage qui ne correspond pas à l'image que le programmeur a de son programme.
• Pour le programmeur, un programme est généralement constitué des données manipulées, d'un programme principal, de procédures séparées et d'une pile d'exécution.

• La *segmentation* est un découpage de l'espace d'adressage qui cherche à conserver cette vu du programmeur.

• Ainsi lors de la compilation, le compilateur associe un segment à chaque morceau du programme compilé. Un *segment* est un ensemble d'emplacement mémoire non sécable.
• A la différence des pages, les segments d'un même espace d'adressage peuvent être de taille différente.



D'une manière générale, on trouvera un segment de code, un segment de données et un segment de pile.

Gestion de la mémoire centrale

Allocation de la mémoire physique – allocation non contiguë
Segmentation

Adresse logique et table des segments

Conversion adresse logique – adresse physique

- La segmentation de l'espace d'adressage d'un processus génère des *adresses segmentées*, i.e qu'un octet est repéré par son emplacement relativement au début du segment auquel il appartient.
- L'adresse d'un octet est donc formée par le couple
<numéro de segment *s*, déplacement *d* relativement au début du segment *s*>

- Pour toute opération concernant la mémoire, il faut ici encore convertir l'adresse segmentée générée au niveau du processeur en une adresse physique équivalente.

- L'adresse physique d'un octet s'obtient à partir de son adresse segmentée en remplaçant le numéro de segment *s* par l'adresse physique d'implantation en mémoire centrale et en ajoutant à cette adresse physique, le déplacement *d* de l'octet dans le segment. C'est la MMU qui effectue cette conversion.

- Il faut donc connaître pour tout segment, l'adresse d'implantation dans la mémoire centrale du segment : cette correspondance s'effectue grâce à une structure particulière appelée la *table des segments*.

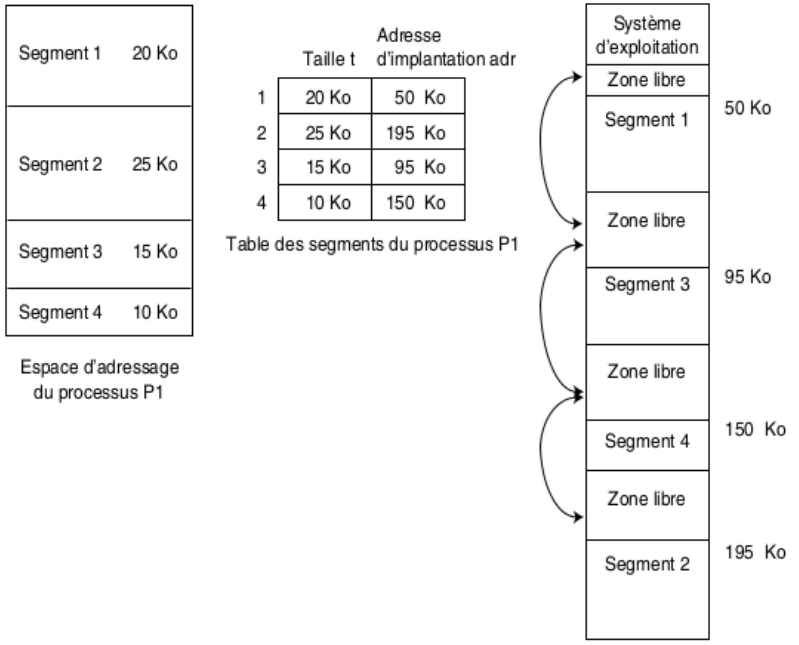
Gestion de la mémoire centrale

Allocation de la mémoire physique – allocation non contiguë
Segmentation

La table des segments

- La table des segments est une table contenant autant d'entrées que de segments dans l'espace d'adressage d'un processus.
- Chaque entrée *i* de la table est un couple **<adresse adr d'implantation du segment, taille t du segment i>**

- Sur la figure, le processus P1 a 4 segments dans son espace d'adressage, donc la table des segments a 4 entrées. Chaque entrée établit l'équivalence entre d'une part le numéro de segment, d'autre part l'adresse d'implantation du segment et sa taille.



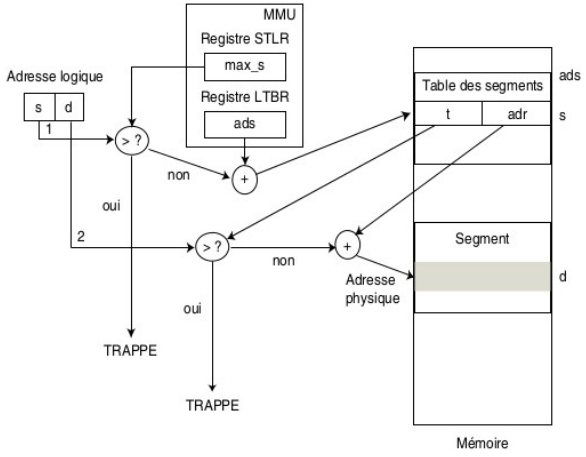
Gestion de la mémoire centrale

Allocation de la mémoire physique – allocation non contiguë
Segmentation

La table des segments

• Tout comme la table des pages, la table des segments peut être réalisée à l'aide de registre de la MMU, mais avec les mêmes limites au niveau de la taille de la table.
• Aussi, la table des segments est le plus souvent placée en mémoire centrale.
• Un registre STBR (*Segment-Table Base Register*) contient l'adresse d'implantation en mémoire centrale de la table de segments du processus actif, tandis qu'un registre STLR (*Segment-Table Length Register*) contient le nombre de segment existant dans l'espace d'adressage du processus actif.
• Les valeurs contenues dans les registres STLR et STBR sont sauvegardées pour chaque processus dans son PCB.

• La conversion d'une adresse segmentée <*s,d*> suit les étapes suivantes :
➔ s > max_s, alors une trappe est levée car le segment n'existe pas ;
➔ Sinon *s* est additionné au contenu du registre STBR de manière à indexer l'entrée de la table concernant le segment *s*. on récupère alors l'adresse d'implantation `adr` du segment s en mémoire centrale.
➔ Le déplacement *d* est alors comparé à la taille *t* du segment. Si *d* > *t*, alors une trappe est levée car le déplacement est en dehors du segment. Sinon, le déplacement d est ajouté à l'adresse d'implantation du segment pour générer l'adresse physique.



Gestion de la mémoire centrale

Allocation de la mémoire physique – allocation non contiguë
Segmentation

Protection de l'espace d'adressage d'un processus

Reprendre la même présentation que pour la pagination, et remplacer le terme page par segment.

Pagination des segments

• L'allocation des segments en mémoire centrale s'effectue selon le même principe que pour l'allocation de partitions variables. Pour allouer un segment de taille *t*, il faut trouver une zone libre dont la taille soit au moins égale à la taille du segment.
• Elle engendre les mêmes problèmes de fragmentation externe.

• Une solution, très largement répandue, au problème de la fragmentation externe, est de combiner pagination et segmentation, i.e de paginer les segments.

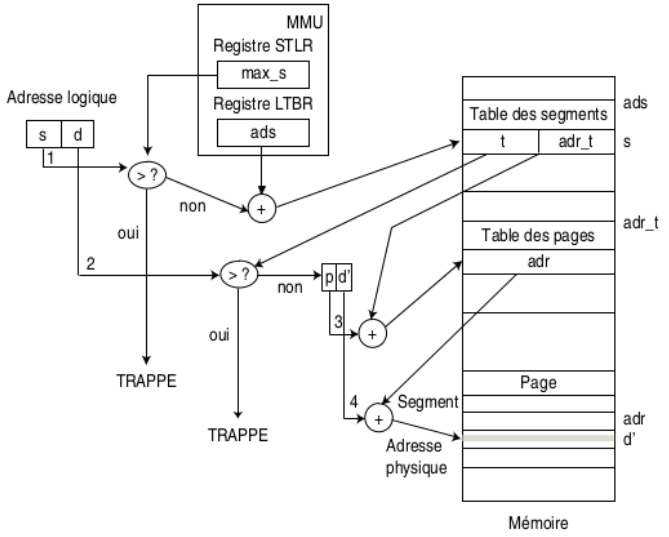
• Il existe donc une table des pages pour chaque segment. Ainsi une entrée de la table des segment ne contient plus l'adresse du segment correspondant en mémoire physique mais contient l'adresse de la table des pages en mémoire physique pour ce segment.

• L'adresse d'un octet dans l'espace d'adressage du processus devient un couple <*s, d*>, le déplacement *d* étant à son tour interprété comme étant un couple **<numéro de page p, déplacement d' dans cette page>**.

Gestion de la mémoire centrale

Allocation de la mémoire physique – allocation non contiguë
Segmentation

Pagination des segments



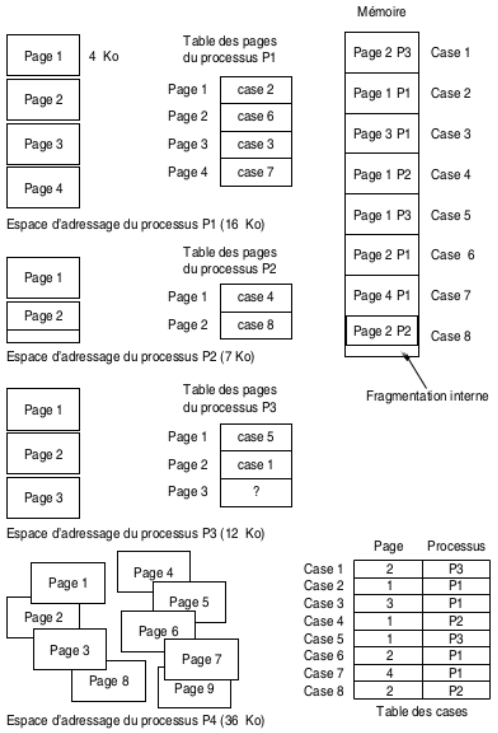
• La conversion d'une adresse segmentée <s,d> suit les étapes suivantes :
➔ s > max_s, alors une trappe est levée car le segment n'existe pas;
➔ Sinon s est additionné au contenu du registre STBR de manière à indexer l'entrée de la table concernant le segment s. On récupère alors l'adresse d'implantation `adr_t` de la table des pages en mémoire centrale.
➔ Le déplacement d est alors comparé à la taille t du segment. Si d > t, alors une trappe est levée car le déplacement est en dehors du segment.
➔ Sinon, le déplacement d est alors décomposé en numéro de page p et un déplacement d' dans la page p. Le numéro de page p est ajouté à l'adresse `adr_t` de la table des page du segment s pour indexer dans la table l'entrée concernant la page p.
• L'adresse `adr` d'implantation en mémoire centrale de la case contenant la page p est ainsi récupérée;
• Enfin, le déplacement d' est ajouté à `adr` pour constituer l'adresse physique de l'octet recherché.

Gestion de la mémoire centrale

Mémoire virtuelle - Principe de la mémoire virtuelle

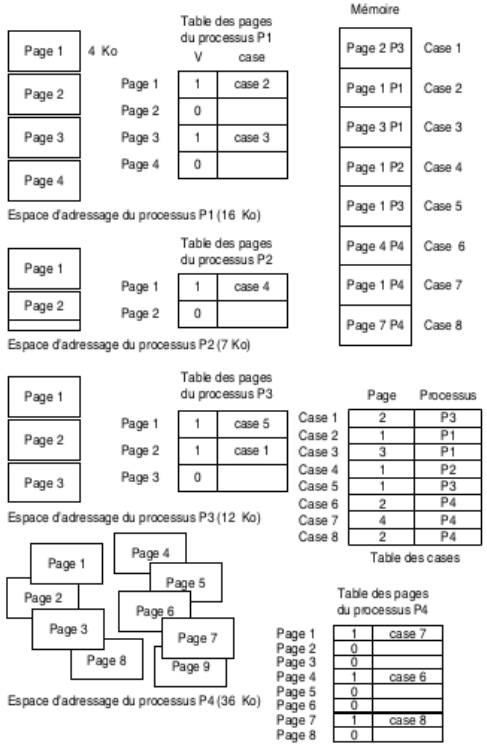
- La multiprogrammation implique de charger plusieurs programmes en mémoire centrale de manière à obtenir un bon taux d'utilisation du processeur.
- Supposons comme sur la figure que l'exécution des processus P1, P2 et P3 soit nécessaire pour obtenir ce taux d'utilisation du processeur. On peut remarquer qu'une fois les pages des processus P1 et P2 chargées dans la mémoire, il ne reste pas assez de cases libres pour charger la totalité des page du programme 3. ce dernier ne peut être donc chargé.

- Lorsqu'on regarde l'exécution d'un processus, on s'aperçoit qu'à un instant donné le processus n'accède qu'à une partie de son espace d'adressage.
- Une solution pour pouvoir charger plus de processus en mémoire centrale est donc de ne charger pour chaque processus que les pages couramment utilisées.



Gestion de la mémoire centrale

Mémoire virtuelle - Principe de la mémoire virtuelle



• Sur la figure, seule les page 1 et 3 de P1 sont chargées ainsi que la page 1 du processus P2 et les page 1 et 2 de P3.

• Par ailleurs, on remarque que l'espace d'adressage de P4 est plus grand que la mémoire physique disponible. Le *principe de la mémoire virtuelle*, qui consiste donc à ne charger à un instant donné en mémoire centrale que la partie couramment utile de l'espace d'adressage des processus, permet de résoudre ce problème et autorise la constitution de programme dont la taille n'est plus limitée par celle de la mémoire physique.

• Pour que le processeur puisse détecter l'éventuelle absence de pages en mémoire centrale, chaque entrée de la table des pages comporte un champ validation V, qui est à 1 si la page en mémoire centrale, 0 sinon.

• Le principe de la mémoire virtuelle est couramment implémenté avec la pagination à la demande , i.e que les pages des processus ne sont chargées en mémoire centrale que lorsque le processeur demande à y accéder.

Gestion de la mémoire centrale

Mémoire virtuelle - Le défaut de page

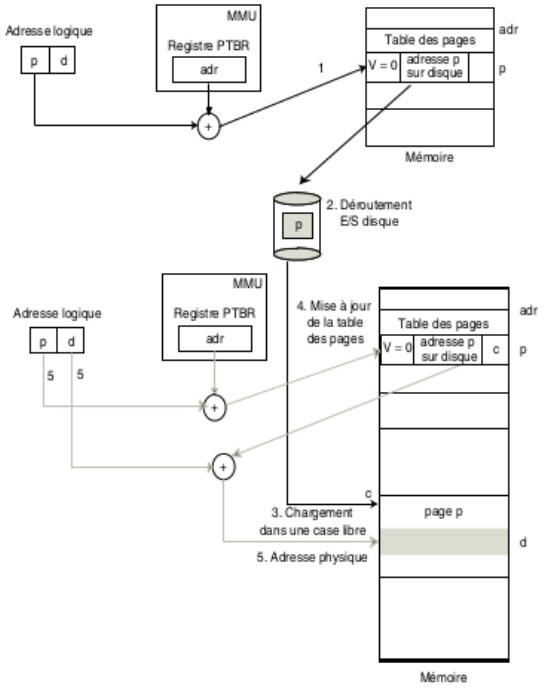
Que se passe t-il à présent lorsqu'un processus tente d'accéder à une page de son espace d'adressage qui n'est pas en mémoire centrale ? La MMU accède à la table des pages pour effectuer la conversion de l'adresse logique vers l'adresse physique et teste la valeur du bit de validation. Si la valeur du bit V est à 0, cela signifie que la page n'est pas chargée dans une case et donc la conversion ne peut se faire. Une trappe appelée *défaut de page* est levée qui suspend l'exécution du processus puis initialise une opération d'entrées/sorties afin de charger la page manquante en mémoire centrale dans une case libre. Les pages constituants l'espace d'adressage du processus sont stockées dans une zone particulière du disque communément appelée *zone de swap*. L'adresse de chaque page sur le disque est mémorisée dans l'entrée correspondante de la table des pages.

Gestion de la mémoire centrale

Mémoire virtuelle - Le défaut de page

• Avec le principe de pagination à la demande, le mécanisme de conversion d'une adresse logique <p,d> vers l'adresse physique correspondante devient :

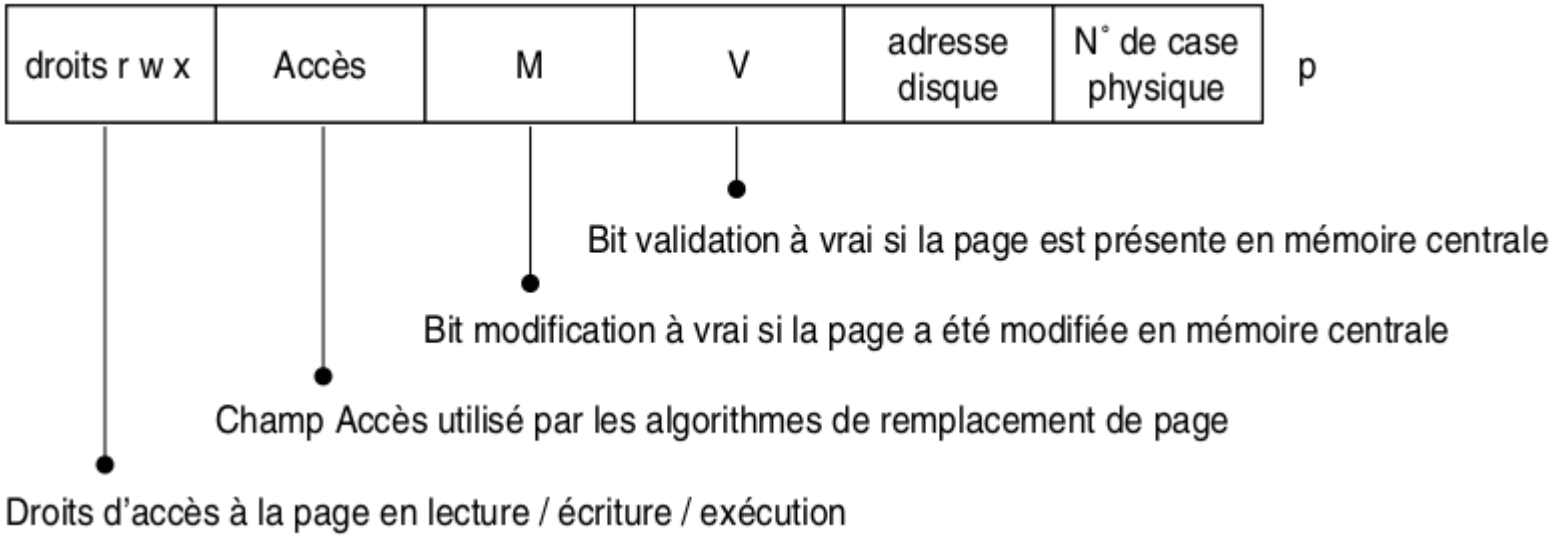
• Accès à l'entrée p de la table des pages du processus actif et test de la valeur du bit de validation V;
• Si la valeur du bit V est à 0, alors il y a défaut de page. Une opération d'entrées/sorties est lancée pour charger la page dans la mémoire centrale (l'adresse de la page sur le disque est stockée dans la table des pages) ;
• La page est placée dans une case libre, trouvée par l'intermédiaire de la table des cases;
• La table des pages du processus est mise à jour i.e que le bit de validation V passe à 1 et le champ numéro de case est renseigné avec l'adresse d'implantation de la case abritant maintenant la page p;
• La conversion de l'adresse logique vers l'adresse physique est reprise selon le principe vu précédemment



Gestion de la mémoire centrale

Mémoire virtuelle - Le remplacement de pages

Lors d'un défaut de page, la page manquante est chargée dans une case libre. Cependant, la totalité des cases de la mémoire centrale peut être occupée. Il faut donc libérer une case de la mémoire physique pour y placer la nouvelle page. Lors de la libération d'une case, la page victime contenue dans cette case doit être sauvegardée sur le support de masse si elle a été modifiée lors de son séjour en mémoire centrale. Un bit M de modification mis à 1 si la page est modifiée est ajouté à chaque entrée de la table des pages afin d'être à même de savoir si la page doit être réécrite sur le disque avant d'être écrasée par la nouvelle page.



Gestion de la mémoire centrale

Mémoire virtuelle - Le remplacement de pages

• Plusieurs algorithmes, appelés *algorithmes de remplacement de pages*, ont été définis qui tendent plus ou moins vers l'objectif optimal. Ce sont les stratégies : FIFO, LRU (*Least Recently Used*), algorithme de la seconde chance, LFU (*Least Frequently used*) et MFU (Most Frequently Used).

• L'évaluation de ces différentes stratégies s'effectue en comptant sur une même suite de référence à des pages, le nombre de défaut de pages provoqués. Une telle suite de référence à un même ensemble de pages est appelée *chaîne de références*.
• Ainsi dans la suite nous donnons un exemple pour les stratégies FIFO et LRU sur la chaîne de références 8, 1, 2, 3, 1, 4, 1, 5, 3, 4, 1, 4,3 où chaque chiffre i correspond à un accès à la page i.
• Nous supposerons une mémoire centrale composée de trois cases initialement vides.
La page victime apparaît en italique. La lettre D signale l'occurrence d'un défaut de page.

Chaîne de référence

	8	1	2	3	1	4	1	5	3	4	1	4	3
case 1	**8**	8	8	**3**	3	3	3	**5**	5	5	**1**	1	1
case 2		**1**	1	1	1	**4**	4	4	**3**	3	3	3	3
case 3			**2**	2	2	2	**1**	1	1	**4**	4	4	4
	D	D	D	D		D	D	D	D	D	D		

Remplacement FIFO

Avec cet algorithme, c'est la page la plus anciennement chargée qui est remplacée. La mise en œuvre de cet algorithme est simple, mais ses performances ne sont pas bonnes. En effet l'age d'une page n'est pas le reflet de son utilité.

Chaîne de référence

	8	1	2	3	1	4	1	5	3	4	1	4	3
case 1	***8***	8	8	***3***	3	3	3	***5***	5	5	***1***	1	1
case 2		***1***	1	1	1	1	1	1	***1***	4	4	4	4
case 3			***2***	2	2	***4***	4	4	3	***3***	3	3	3
	D	D	D	D		D		D	D	D	D		

Remplacement LRU

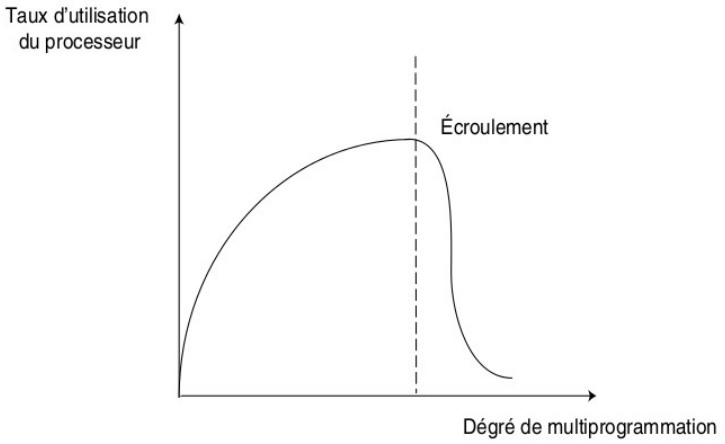
Avec cet algorithme, c'est la page la moins récemment utilisée qui est remplacée. L'algorithme LRU est l'un des algorithme les plus utilisés car il est considéré comme bon. Cependant sa mise en œuvre est couteuse et nécessite le recours à des compteurs matériels spécifiques pour la délivrance des dates d'accès aux pages.

Gestion de la mémoire centrale

Mémoire virtuelle - Notion d'écroulement

On appelle *écroulement*, une haute activité de pagination. Un processus s'écroule lorsqu'il passe plus de temps à paginer qu'a s'exécuter.

La courbe représente le taux d'utilisation du processeur en fonction du degré de multiprogrammation, i.e en fonction du nombre de processus chargés en mémoire centrale. On voit clairement que l'utilisation du processeur augmente jusqu'à un certain seul au-delà duquel cette utilisation chute complètement. Cette chute correspond à une trop grande activité de pagination des processus qui passent le plus clair de leur temps en opérations d'entrées/sorties car il n'ont pas suffisamment de cases mémoires disponible pour contenir les pages relatives à leur espace de travail courant.



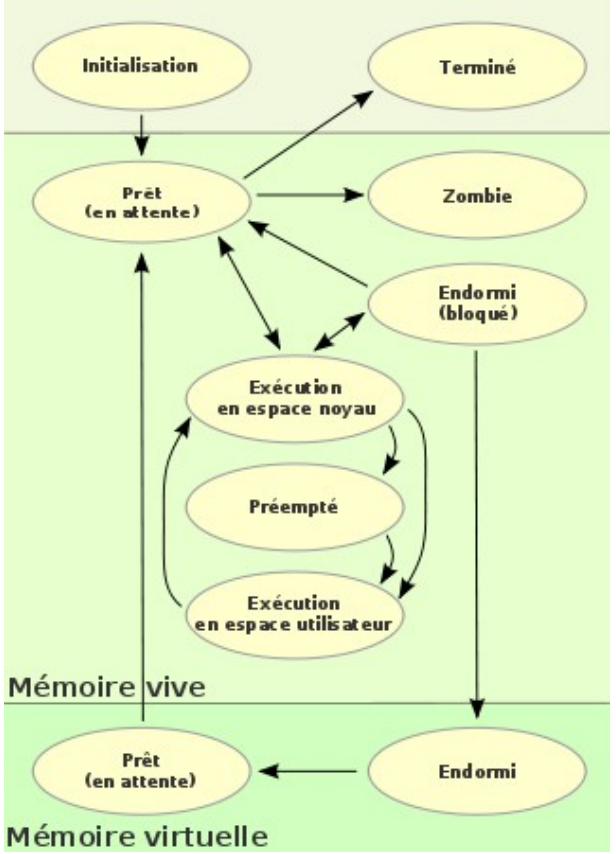
Gestion de la mémoire centrale

Allocation de la mémoire physique – Mémoire virtuelle
Swapping des processus

• L'opération de vidage (*swap-out*) d'un processus consiste à transférer sur disque dans la zone de swap, l'ensemble des pages présente en mémoire centrale de l'espace d'adressage de ce processus. Le processus victime du vidage est un processus endormi (état bloqué) en attente d'une ressource pour la poursuite de son exécution.
=> désengorgement d'un seul coup de la mémoire centrale.

• Le vidage peut être aussi déclenché en cas de situation de crise au niveau de l'occupation de la mémoire centrale.

• L'opération inverse de *swap-in* consiste à ramener les pages du processus depuis la zone de swap lorsque celui-ci redevient actif.



Gestion de la mémoire centrale

Conclusion

L'allocation en **partition variable** considère le programme comme un ensemble d'adresses insécables. Ce type d'allocation pose un problème de fragmentation et nécessite des opérations de compactage de la mémoire centrale.

La **pagination** découpe l'espace d'adressage du programme en pages et la mémoire physique en cases de même taille. Une adresse générée par le processeur est de la forme <numéro de page, déplacement dans la page>. La table des pages du processus permet de traduire l'adresse paginée en adresse physique.

La **segmentation** découpe l'espace d'adressage du programme en segment correspondant à des morceaux du programme. Une adresse générée par le processeur est de la forme <numéro de segment, déplacement dans le segment>. La table des segment du processus permet de traduire l'adresse l'adresse segmentée en adresse physique.

Remarque : Segmentation et pagination sont très souvent associées.

Lorsque le principe de la **mémoire virtuelle** est appliqué, les pages d'un processus ne sont chargées en mémoire centrale que lorsque le processus y accède (***pagination à la demande***). Lorsqu'un processus accède à une page non présente en mémoire centrale, il se produit un **défaut de page**. La page manquante est alors chargée dans une case libre. Si aucune case n'est libre, le système utilise un **algorithme** de **remplacement** de **pages** pour choisir une case à libérer.

L'écroulement est la situation pour laquelle un ou plusieurs processus passent plus de temps à paginer qu'à exécuter.

Gestion de la mémoire centrale

Exercices

Exercice 1 : Gestion de la mémoire par partitions variables

Soit un système qui utilise l'allocation par partitions variables. On considère à l'instant t, l'état d'allocation de la mémoire centrale représenté sur la figure 16.1, où les zones grisées représentent les zones libres.

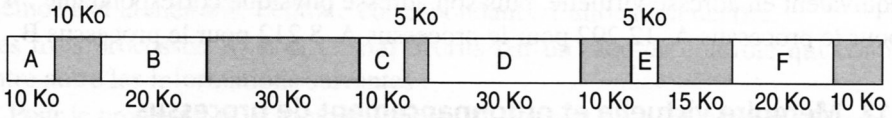


Figure 16.1 Allocation de la mémoire à t.

1. Représentez l'évolution de la mémoire centrale, suite à chacun des événements suivants, en supposant une stratégie de choix *First Fit* :
   1) arrivée du programme G de taille égale à 20 Ko;
   2) départ du programme B;
   3) arrivée du programme H de taille égale à 15 Ko;
   4) départ du programme E;
   5) arrivée du programme I de taille égale à 40 Ko.
2. Même question mais en supposant une stratégie de choix *Best Fit*.

Gestion de la mémoire centrale

Exercices

Exercice 2 : Remplacement de pages

Soit la liste de pages virtuelles référencées aux instants t = 1, 2, ..., 11 : 3, 5, 6, 8, 3, 9, 6, 12, 3, 6, 10

Donner l'état de la mémoire central composée de 4 cases selon les algorithmes FIFO et LRU.

Gestion de la mémoire centrale

Exercices

Exercice 3 : Mémoire paginée et segmentée

On considère une mémoire segmentée paginée pour laquelle les **cases** en mémoire centrale sont de **4 Ko**. La **mémoire** centrale compte au total **15 cases** numérotées de 1 à 15. Dans ce contexte, on considère deux **processus A** et **B**. Le processus **A** a un **espace d'adressage** composé de trois segments **S1A**, **S2A** et **S3A** qui sont respectivement de **8 Ko**, **12 Ko** et **4 Ko**. Le processus **B** a un espace d'adressage composé de deux segments **S1B** et **S2B** qui sont respectivement de **16 Ko** et **8 Ko**. Pour le processus **A**, seules les **pages 1** et **2** du segment **S1A**, la **page 2** du segment **S2A** et la **page 1** du segment **S3A** sont **chargées** en mémoire centrale respectivement **dans** les **cases 4, 5, 10, 6**. Pour le processus **B**, seules les **pages 2 et 3** du segment **S1B** et la **page 1** du segment **S2B** sont **chargées** en mémoire centrale respectivement **dans** les **cases 11, 2 et 15**.

1. Représentez sur un dessin les structures allouées (table des segments, tables des pages) et la mémoire centrale correspondant à l'allocation décrite.
2. Soit l'adresse logique <S1A, page 1, 12>. Quelle adresse réelle lui correspond-elle ?
3. Soit l'adresse logique <S2B, page 2, 10>. Quelle adresse réelle lui correspond-elle ?
4. Dans ce même contexte, donnez pour chacune des adresses linéaires suivantes, son équivalent en adresse virtuelle, puis son adresse physique correspondante : 4 098 pour le processus A, 12 292 pour le processus A, 8 212 pour le processus B.