

Gestion Mémoire

B. Dupouy

Plan

- Allocation contigüe
 - Partitions de taille fixe
 - Partitions de taille variable
 - gestions des trous ? (compactage, ramasse miettes)
 - stratégies de placement (Worst fit, best fit, first fit)
 - évaluation de Knuth
- Pagination
 - La table de pages
- Mémoire virtuelle
 - Taille de la table de pages, pagination à deux niveaux
 - Temps pour les traductions d'adresses (Translation buffer)
- Algorithmes de remplacement de pages (fifo, lru, lfu)
 - exemple : Unix
 - exemple : VMS
- Conditions de bon fonctionnement
 - Taille de la page (pbs : fragmentation, transferts)
 - Swap ou pagination
 - Translation buffer
 - Principe de localité
 - Caches mémoire

Allocation contigüe

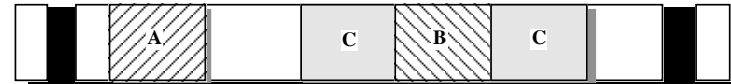
- Partitions de taille fixe
 - égales
 - différentes (background, foreground / code absolu, relogeable)
- Partitions de taille variable
 - gestions des trous ? (compactage, ramasse miettes)
 - stratégies de placement (Worst fit, best fit, first fit)
 - évaluation de Knuth
- Swapping

Gestion mémoire

Gestion mémoire

Evaluation de Knuth

- Evaluer le nombre de blocs libres par rapport au nombre de blocs occupés. Il y a trois types de blocs occupés :



O Nbre de blocs occupés

L Nbre de blocs libres

$$(1) O = O_A + O_B + O_C \quad (\text{où } O_i \text{ signifie : nbre de blocs de type } i)$$

$$(2) L = O_A + O_C/2 + k$$

$$\text{Prob diminuer nbre blocs libres : } P(L \text{ fi } L-1) = O_A/O + q$$

$$\text{Prob de l'augmenter : } P(L \text{ fi } L+1) = O_B/O \text{ (libérer un bloc B)}$$

En période de croisière :

$$O_A/O + q = O_B/O, \text{ fi } O_A = O_B$$

de (1) et (2) on déduit :

$$O = 2*O_A + O_C \text{ et } L = O_A + O_C/2$$

$$\text{Nbre de blocs occupés} = 2 \text{ fois nbre de blocs libres}$$

Gestion mémoire

Gestion mémoire

Gestion de la mémoire virtuelle

- La table de pages
- Problèmes dus à la pagination
 - Taille de la table de pages (réponse : pagination à plusieurs niveaux)
 - Temps pour les traductions d'adresses (réponse : *translation buffer*)
- Algorithmes de remplacement de pages (fifo, lru, lfu)
 - exemple : Unix
 - exemple : VMS
- Conditions de bon fonctionnement
 - Taille de la page (pbs : fragmentation, transferts)
 - Swap ou pagination
 - Translation buffer
 - Principe de localité
 - Caches mémoire

Gestion mémoire

Le principe de gestion de la mémoire en mémoire virtuelle repose sur l'utilisation conjointe de la mémoire "centrale" et d'une mémoire de masse (disque). Cette dernière est destinée à compléter l'espace de mémorisation relativement réduit offert par la mémoire centrale. L'utilisateur perçoit l'ensemble *mémoire centrale - mémoire de masse* comme une seul et même espace d'adressage de très grande taille.

Un mécanisme de transfert entre mémoire de masse et mémoire physique prend en charge la gestion de cet espace, de telle sorte que l'utilisateur n'a pas à se soucier de l'emplacement effectif des informations qu'il manipule. Les données sont rangées et les programmes écrits comme si l'on disposait d'une mémoire centrale de taille égale à celle dont on dispose dans l'ensemble (ou dans un sous-ensemble) des unités de stockage. La seule chose à laquelle l'utilisateur sera sensible sera le retard que pourront subir les réponses à ses requêtes, à cause des transferts entre la mémoire de masse et la mémoire centrale.

L'exécution des programmes engendre des adresses dites *virtuelles* ou *logiques*. Le mécanisme de gestion assure une *séparation* complète entre l'espace logique vu par le programmeur et l'espace physique où sont implantées les informations.

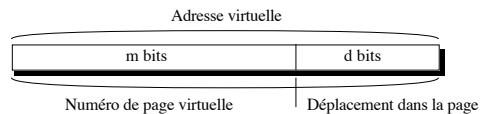
Ce mécanisme assure la *traduction* (*Translation*) des adresses virtuelles en adresses *réelles* en mémoire centrale.

Le dispositif de traduction est capable de détecter si une information se trouve ou non en mémoire centrale. Si l'information à laquelle on veut accéder ne s'y trouve pas, il faudra l'y amener, en libérant, si nécessaire, de la place. On parle, dans ce cas, de traitement de *défaut de page* (*Page Fault*).

On ne charge en mémoire que la page à laquelle appartient l'information recherchée :

- les pages étant toutes de même dimension, le *remplacement* des pages les unes par les autres est simple
- les critères de choix gouvernant le remplacement des pages varient suivant les systèmes : pages les moins couramment utilisées, les plus anciennes en mémoire, ...

La traduction repose sur une technique de *numérotation* des pages. Le numéro de page est obtenu à partir des m bits de poids fort de l'adresse virtuelle.

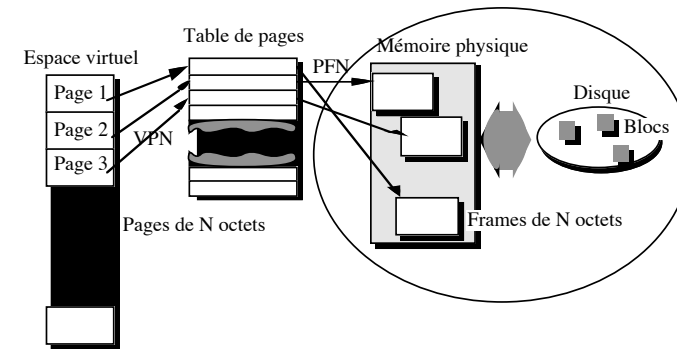


Le mécanisme de gestion consulte la table de pages pour vérifier si la page à laquelle on veut accéder est bien en mémoire centrale.

Gestion mémoire

La table de pages

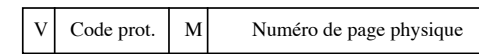
- Rôle de la table de pages



- Terminologie :

- VPN virtual page number, numéro de page virtuelle,
- PFN page frame number, numéro de page physique,
- PTE : page table entry, entrée dans la table de pages :

- Format d'une entrée :



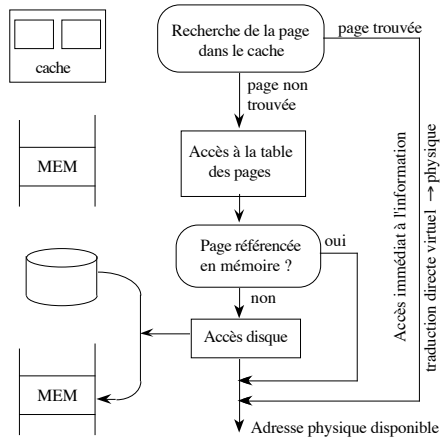
le bit V (valid) indique si la page est en mémoire,
le bit M (modified) indique si on a écrit dans cette page (dirty bit).

Gestion mémoire

Opération de lecture :

- l'unité centrale émet une adresse virtuelle. Le mécanisme de traduction indique la présence ou l'absence en mémoire centrale de la page qui contient l'information cherchée,
 - si la page contenant l'information est présente, l'accès est immédiat,
 - sinon il faut amener la page contenant cette information de la mémoire de masse vers la mémoire centrale. Pour faire cela, il va falloir libérer de la place. Or celle-ci peut correspondre à une page qui a été modifiée depuis qu'elle a été amenée en mémoire centrale :
 - si c'est effectivement le cas, cette page doit être recopiée sur la mémoire de masse.
- Ensuite, la page recherchée est amenée en mémoire centrale,
- dans le cas contraire, la page recherchée est amenée en mémoire centrale.

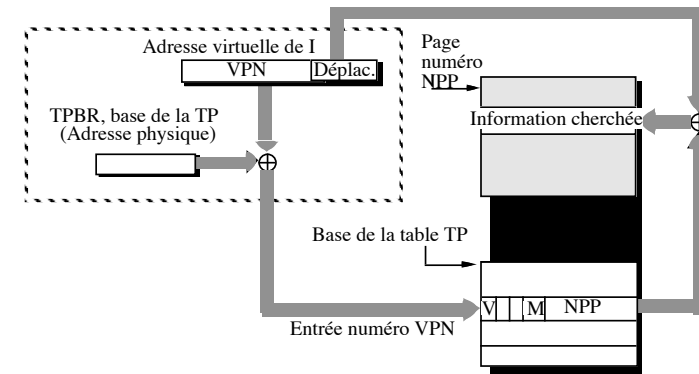
Principe d'accès à l'information



Gestion mémoire

Pagination simple

- Calcul de l'adresse physique de la cellule I dont on connaît l'adresse virtuelle : (VPN, Déplac).



- le registre TPBR contient l'adresse (adresse **physique**) d'implantation de la table de pages TP

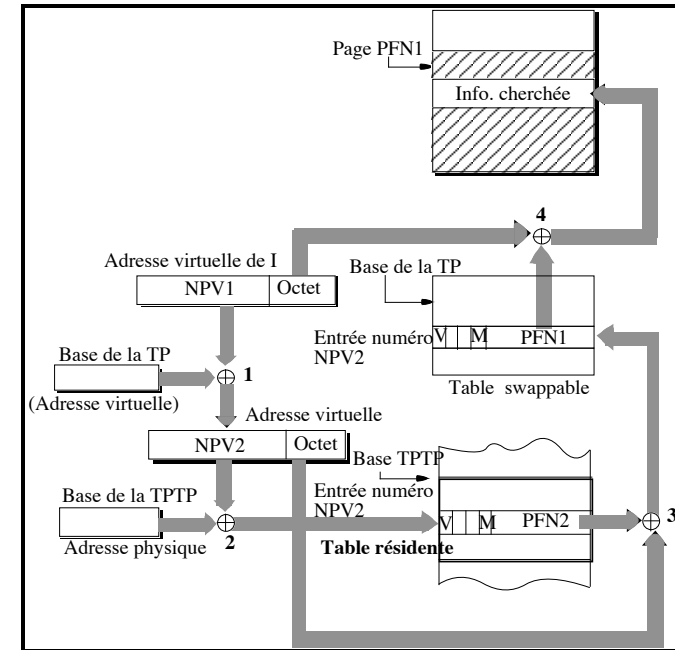
Gestion mémoire

La table de pages peut occuper une place considérable en mémoire. On prendra pour exemple une machine à bus d'adresse de 32 bits : si la taille des pages est de 1 kilo-octets, soit 10 bits de déplacement, le numéro de page est sur 22 bits. La table de page doit donc disposer de 2^{22} octets, soit 4 méga-octets, pour ranger les adresses physiques de pages. Comme chaque numéro de page physique est lui-même sur 22 bits, et qu'il faut ajouter à cela des bits de contrôle, une entrée de la table de pages est constituée de 32 bits. La table occupera 16 méga-octets ! Cela explique pourquoi on a introduit un niveau supplémentaire de table, le premier étant paginé par le second : la table de pages est gérée comme un simple tableau de l'utilisateur : elle est elle-même paginée.

Il faut de plus souligner que lorsque la machine travaille en multi-tâche, une table de pages est associée à chacune des tâches.

Gestion mémoire

Pagination à deux niveaux



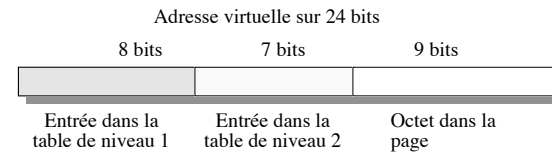
1	Ceci donne une adresse virtuelle, il faut la traduire en adresse physique,
2	la partie page de cette adresse virtuelle ajoutée à l'adresse de base fournit une entrée dans la TPTP. Le contenu de cette case donne PFN2 qui est le numéro de page où se trouve la partie de la table de de page qui nous intéresse.
3	On ajoute la partie octet de l'adresse virtuelle de la page pour trouver l'entrée dans TP
4	le contenu de cette dernière donne le numéro de page physique où est implantée la variable I. On lui ajoute la partie octet de l'adresse virtuelle de I, cette opération donne l'adresse physique de la cellule où est logée I. Ouf.

Gestion mémoire

Gestion mémoire

Pagination à deux niveaux faite par matériel

- L'adresse est divisée en trois champs :



- L'adresse de la table (256 entrées) de niveau 1 est donnée par un registre
- Les 7 bits suivants donnent une entrée, parmi 128, dans une des 256 tables de pages.

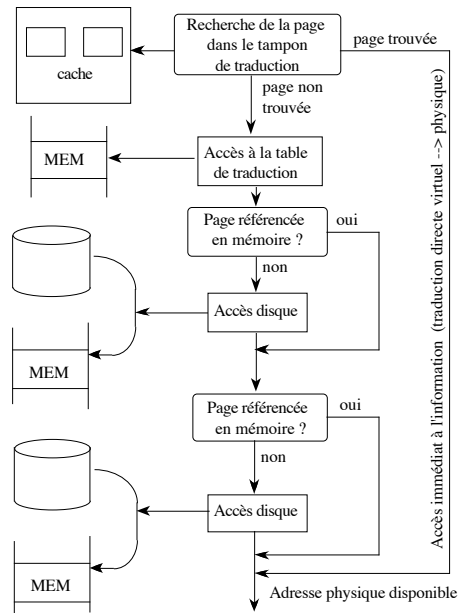
Gestion mémoire

Les huit bits de poids fort fournissent une entrée, parmi 256, dans une table appelée *table de traduction* donnant les adresses de tables de pages. L'adresse de cette table, résidant de façon permanente en mémoire, est construite à partir du contenu d'un registre désigné par RPTP (*Registre Pointeur dans la Table des Pages*).

Les 7 bits suivants, combinés aux 15 bits fournis par la première table donnent une entrée, parmi 128, dans une des tables de pages.

Les 9 bits de poids faible fournissent le déplacement, dans la page désignée par la table de pages, de l'octet recherché.

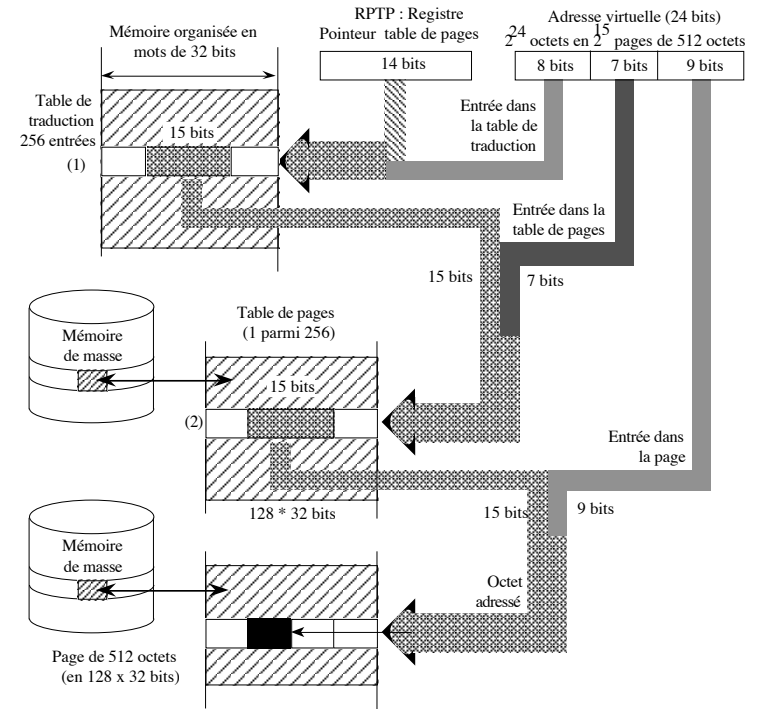
Le principe de l'accès à l'information est le suivant :



La gestion en mémoire virtuelle s'applique aux tables de pages comme aux pages. Seule la table dite table de traduction reste en mémoire.

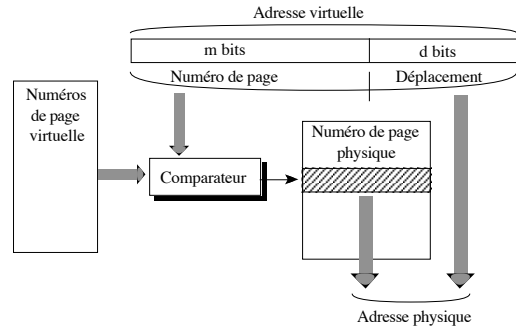
Gestion mémoire

Pagination à deux niveaux faite par matériel (2)



Gestion mémoire

La consultation des tables augmente le temps d'accès à l'information cible. On introduit un cache associatif (appelé *Translation Buffer*) qui reçoit les numéros des dernières pages virtuelles auxquelles on a fait référence.



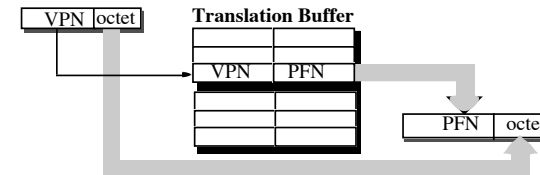
Si un numéro de page virtuelle se trouve dans ce cache, (*hit*) la traduction adresse virtuelle → adresse physique est immédiate et n'engendre aucun accès à la mémoire.

Dans le cas contraire, il faut consulter les tables résidant, ou non, en mémoire centrale.

Gestion mémoire

Tampon de traduction (TLB)

- Pour éviter de consommer trop de temps en calcul d'adresse, les processeurs disposent d'un cache de pagination appelé translation buffer (TLB) ou tampon de traduction, dans lequel sont rangés les numéros des pages physiques (frames) correspondant aux pages virtuelles les plus récemment utilisées.



Gestion mémoire

Le choix de la taille des pages est un facteur influant sur les performances d'un système informatique faisant appel à la gestion en mémoire virtuelle. La taille des pages est généralement comprise entre 256 octets et 4 kilo-octets. En effet :

- plus réduite est la taille des pages, plus grandes sont les tables de description, d'où l'intérêt d'avoir des pages de taille relativement importante,
- la taille des programmes ou autres structures de données n'étant pas exactement multiple de la taille des pages, il y aura toujours de la place perdue (on appelle ce phénomène la *fragmentation interne*). Aussi aura-t-on intérêt à avoir des pages de taille réduite,
- le temps de transfert entre mémoire de masse et mémoire centrale est d'autant plus court que la taille de la page est réduite. Il ne faut cependant pas en conclure que cette dernière doit être la plus petite possible : il n'y a, en effet, pas proportionnalité entre taille du bloc transféré et temps de transfert : le temps de positionnement des têtes est souvent plus important que le temps de transfert lui-même. Ainsi sera-t-il beaucoup plus rapide de transférer 1 bloc de 2 kilo-octets que 2 blocs de 1 kilo-octet,
- le principe de localité plaide en faveur de pages de petite taille : à un instant donné, on n'a pas besoin de tout le programme dans la mémoire centrale,
- d'un autre côté, si les pages sont trop petites, il y aura beaucoup de transferts entre mémoire de masse et mémoire centrale, d'où perte de temps due aux sauvegardes de contexte et à l'exécution de l'algorithme d'allocation de place en mémoire centrale,
- un autre facteur de choix est la relation existant entre la taille du secteur physique sur la mémoire de masse (unité physique du transfert) et la taille de la page (unité logique du transfert).

En pratique on choisit une taille des pages égale à une à deux fois la taille des secteurs disque, ou égale à l'unité d'allocation disque (*granule* ou *cluster*).

Gestion mémoire

Pagination

- algorithmes : FIFO, LRU, LFU
 - implantation de LRU :
 - matérielle -> matrice d'accès (sert aussi pour les caches)
 - logicielle -> *aging*
 - choix de la taille de la page
 - petite -> moins de fragmentation interne
 - grande -> TP plus petite
- Soit :

Taille programme moyen	P octets
Taille page	p octets
Taille entrée TP	e octets

-> TP occupe $P/p * e$ octets

-> "Gaspillage" mémoire = TP + frag. = $P*e/p + p/2$

-> $p = (2*P*e)^{1/2}$

- notion d'espace de travail (working set) :

L'ensemble des pages que le processus peut atteindre sans défaut de page, ou encore le nombre de pages qu'il peut garder en mémoire.

Gestion mémoire

Gestion mémoire

Swapping -Pagination

- swapping/pagination

L'ensemble des working sets présents en mémoire s'appelle le balance set et la pagination est au working set ce que le swapping est au balance set : il s'agit de choisir quel élément remplacer par un nouveau.

Le swapping concerne **l'ensemble** des pages d'un processus, alors que la pagination ne met en cause qu'une seule de ces pages. Les algorithmes de remplacement dans le cas du swapping sont beaucoup plus simples que ceux de la pagination : il s'agit de choisir le processus le moins prioritaire pour le ranger sur disque dans le fichier de swap, appelé swap file.

objet	sujet	outil
working set (ensemble de pages)	processus	pagination
balance set (ensemble de working sets)	système	swapping

La pagination concerne la gestion de la ressource mémoire au niveau processus tandis que le swapping la considère du point de vue de l'ensemble du système.

Gestion mémoire

Anomalie de Belady

L'algorithme de remplacement de page est FIFO, un processus fait accès à 5 pages virtuelles dans l'ordre suivant :

Page appelée : 0 1 2 3 0 1 4 0 1 2 3 4

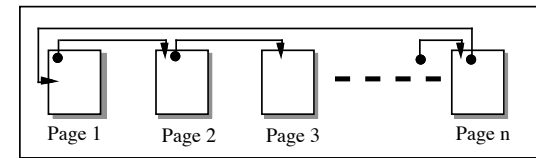
Plus jeune	0	1	2	3	0	1	4	4	4	2	3	3
		0	1	2	3	0	1	1	1	4	2	2
Plus vieille			0	1	2	3	0	0	0	1	4	4
Défauts de page	1	1	1	1	1	1	1	0	0	1	1	0

Plus jeune	0	1	2	3	3	3	4	0	1	2	3	4
		0	1	2	2	2	3	4	0	1	2	3
			0	1	1	1	2	3	4	0	1	2
Plus vieille				0	0	0	1	2	3	4	0	1
Défauts de page	1	1	1	1	0	0	1	1	1	1	1	1

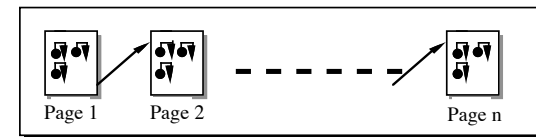
Gestion mémoire

Limites de la transparence

- Une dégradation des performances peut être provoquée par l'utilisateur s'il met, volontairement ou non, en cause le principe de localité.
- Par exemple, l'accès à un tableau de 512*100 octets (sur une machine à pages de 512 octets) peut se faire des deux façons suivantes, la première remet en cause le principe de localité :



512 * 100 manipulations de pages



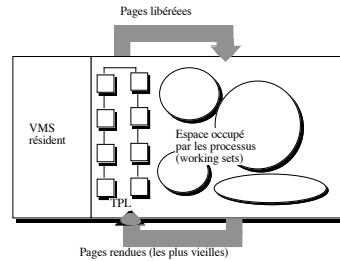
100 manipulations de pages

- Il faut toujours parcourir un tableau dans l'ordre physique où il a été rangé (ligne par ligne en C, colonne par colonne en Fortran...)

Gestion mémoire

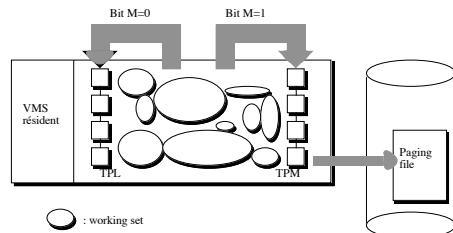
Autres mécanismes de pagination : VMS (Virtual Memory System) de DEC

- L'ensemble des pages libres chaînées entre elles forme la table des pages libres (TPL). Au démarrage celle-ci contient toutes les pages de la mémoire moins celles qui sont affectées au système



- Les pages souvent référencées n'arrivent pas jusqu'en haut de la pile;
- L'algorithme FIFO est donc pondéré par un second FIFO sur la table de pages libres et la combinaison de ces deux algorithmes approxime un LRU.
Sur VMS : FIFO + FIFO = LRU

- VMS utilise une deuxième table pour les pages modifiées dans laquelle les pages qui arrivent au sommet de la pile sont recopiées dans le fichier de pagination



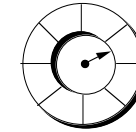
Remarques :

- La TPL et la TPM jouent le rôle d'un cache disque
- La TPM compense la faible taille de pages, puisqu'on écrit (ou lit) plusieurs pages à la fois dans le fichier de pagination.
- Dans l'espace de travail le processus ne pagine que contre lui-même, au contraire dans la table de pages les processus volumineux pénalisent ceux qui utilisent peu de mémoire.

Gestion mémoire

Pagination sous Unix

- L'algorithme de remplacement de page des Unix Berkeley s'appelle *clock hand algorithm*. Son fonctionnement est le suivant : les pages mémoire sont vues comme un tampon circulaire scruté par une aiguille.



Dans chaque page figure un bit appelé *r* (*reference bit*), ce bit *r* est mis à 1 chaque fois qu'un processus utilise cette page.

Voici ce que fait l'aiguille sur chaque page :

- (1) si $r = 1$, alors elle met $r = 0$,
- (2) si $r = 0$, alors on considère que la page est libre.

- Approximation d'un algorithme du type *Global LRU* (*Least Recently Used*)
- Certaines versions (BSD 4.3) utilisent deux aiguilles séparées par *P* pages. La première fait (1) et la seconde (2) puis elles avancent.

Gestion mémoire

Gestion mémoire

Régulation sous Unix

Régulation swap/pagination :

