



Systemes d'exploitation

Mémoire Virtuelle

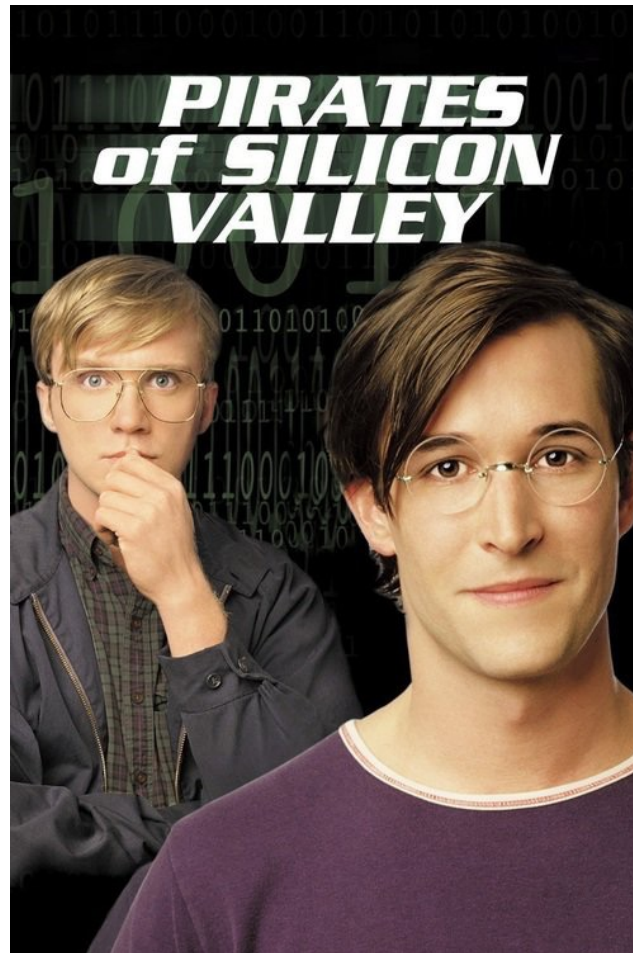
Margaret Hamilton



- Américain
- MIT
- Apollo 11



Pirates of Silicon Valley



Questions?

- Qui a construit le premier ordinateur pour Apple?
- Qu'est-ce que est considéré comme l'un des plus stupide erreurs dans l'histoire de l'ordinateur?
- Lequel des personnages avez-vous le plus aimé?

Questions?

- Qui a construit le premier ordinateur pour Apple?
 - Steve Wozniak
- Qu'est-ce que est considéré comme l'un des plus stupide erreurs dans l'histoire de l'ordinateur?
 - *Les bénéfices sont dans le matériel*
 - *La vente de DOS*
 - *Computers? Why would ordinary people use computers for?*
- Lequel des personnages avez-vous le plus aimé?
 - votre choix

- pagination
- faute de page
- pagination en demande
- copy-on-write
- swap
- mappage de fichier



Bibliographie pour aujourd'hui

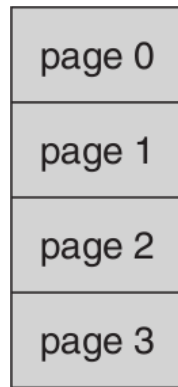
- Modern Operating Systems
 - Chapitre 4
 - 4.4
 - 4.5
 - 4.6
- Operating Systems Concepts
 - Chapitre 9
 - 9.1 – 9.4
 - 9.6
 - 9.7

PAGINATION

Pagination

- La mémoire est divisée en pages
 - en général 4 KB
- Pages
 - Virtuelles (pages)
 - Physique (cadres - frames)
- Tableau de pages
 - un processus a un tableau de pages
- Adresse
 - $\text{adresse physique} = \text{page index} + \text{décalage}$

Tableau de pages

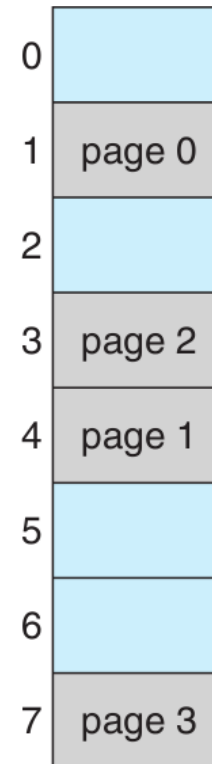


logical
memory

0	1
1	4
2	3
3	7

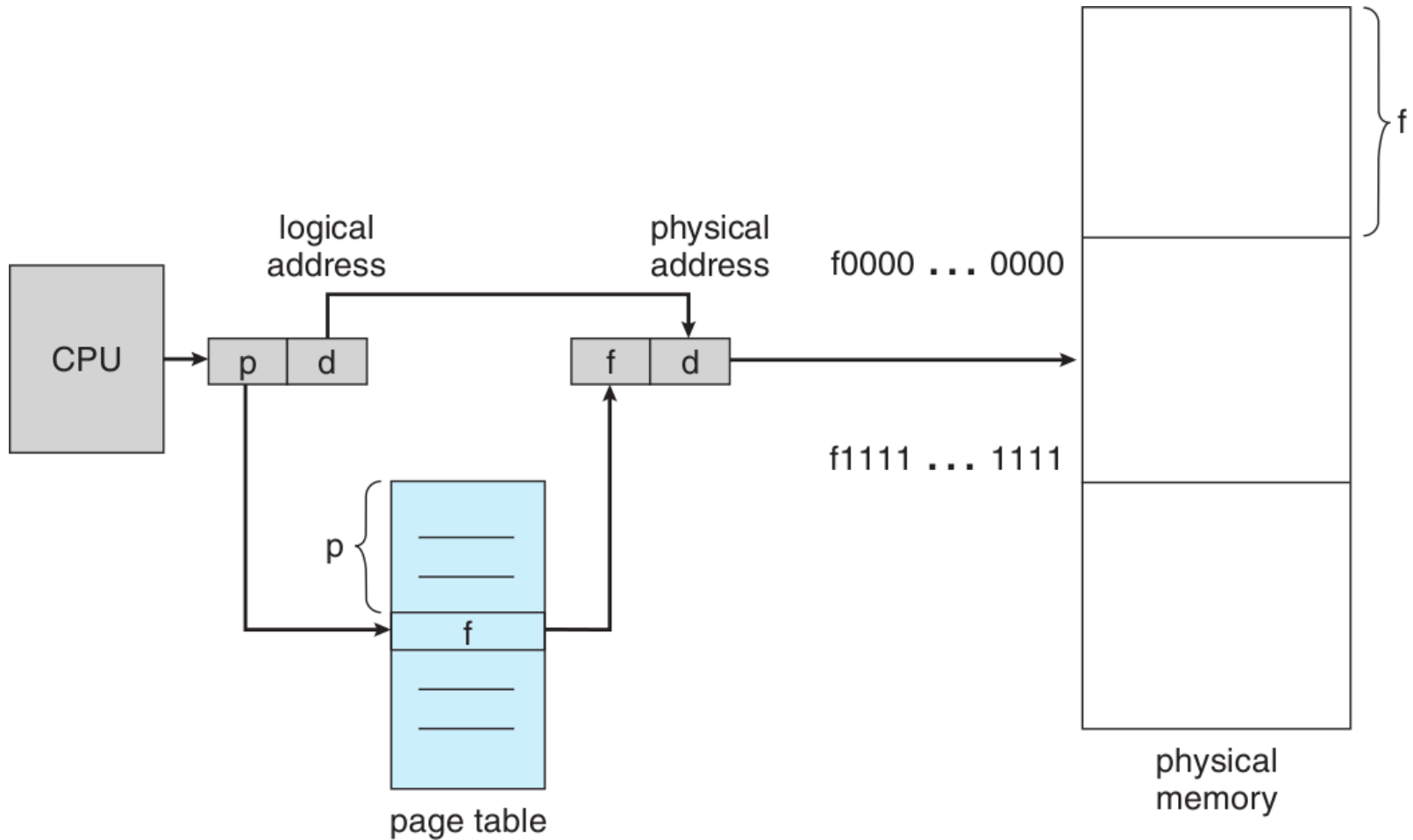
page table

frame
number



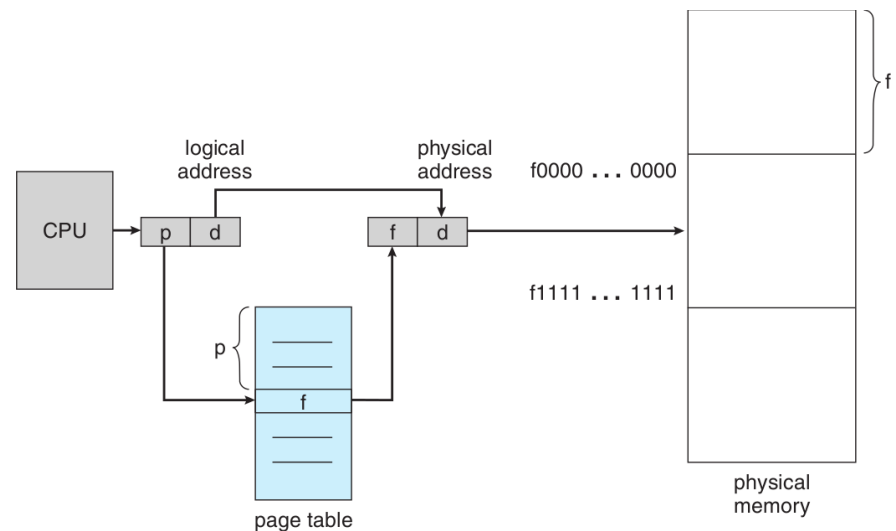
physical
memory

Transformation d'adresse



Transformation d'adresse

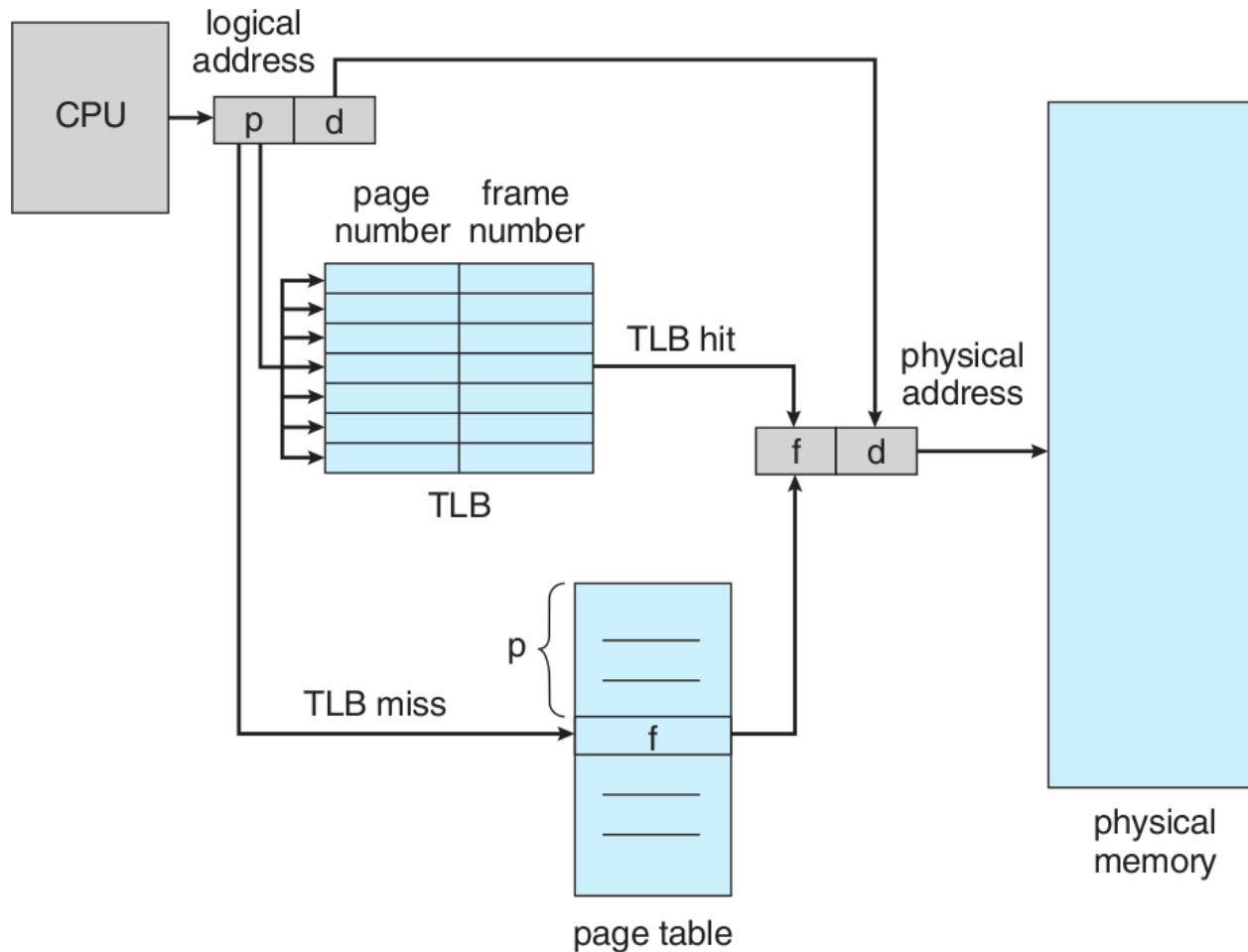
- Pour accède un adresse
 - accès aux tableau (en mémoire)
 - accès en mémoire
- Double accès
 - lente



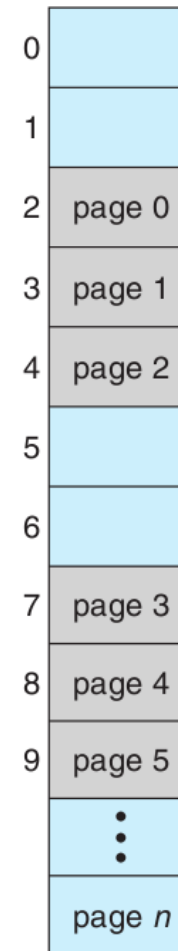
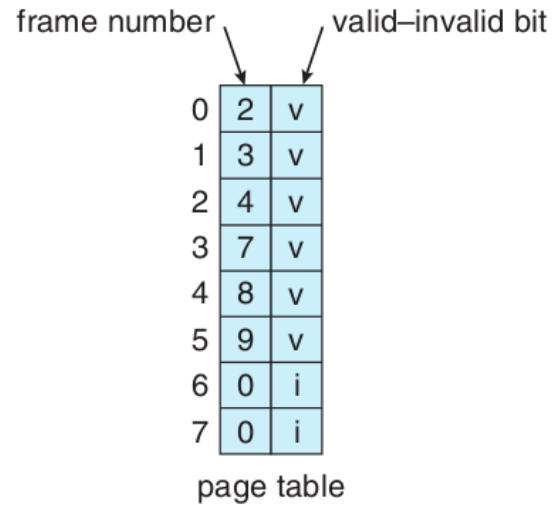
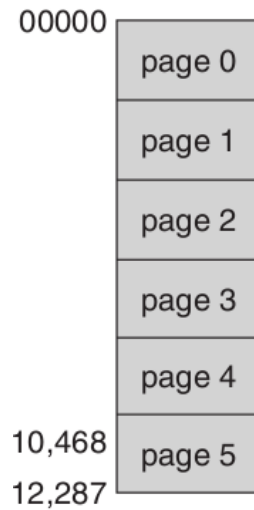
Translation Lookaside Buffer (TLB)

- Mémoire cache spécialisé
 - 256 entrées
- Enregistre par processus
- Changement de contexte
 - change le tableau de pages curent
 - TLB flush (sauf la partie de noyau)

Transformation d'adresse avec TLB



Protection



Entrée de tableau de pages

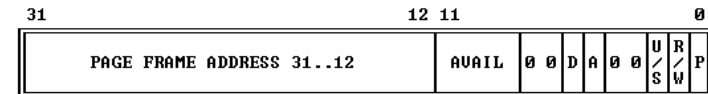
- Numéro de cadre

- Droit d'accès

- aucune
- lire
- écrire

- Valable

Figure 5-10. Format of a Page Table Entry

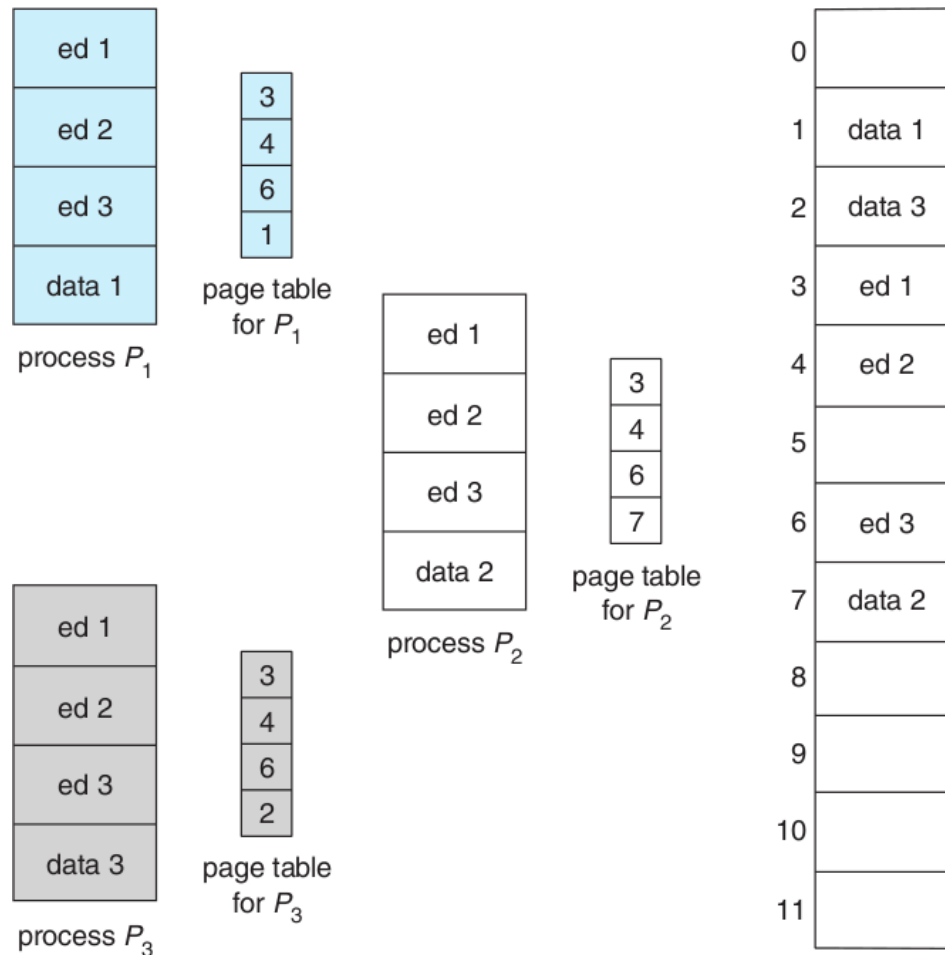


P - PRESENT
R/W - READ/WRITE
U/S - USER/SUPERVISOR
D - DIRTY
AVAIL - AVAILABLE FOR SYSTEMS PROGRAMMER USE

NOTE: 0 INDICATES INTEL RESERVED. DO NOT DEFINE.

exemple sur x86

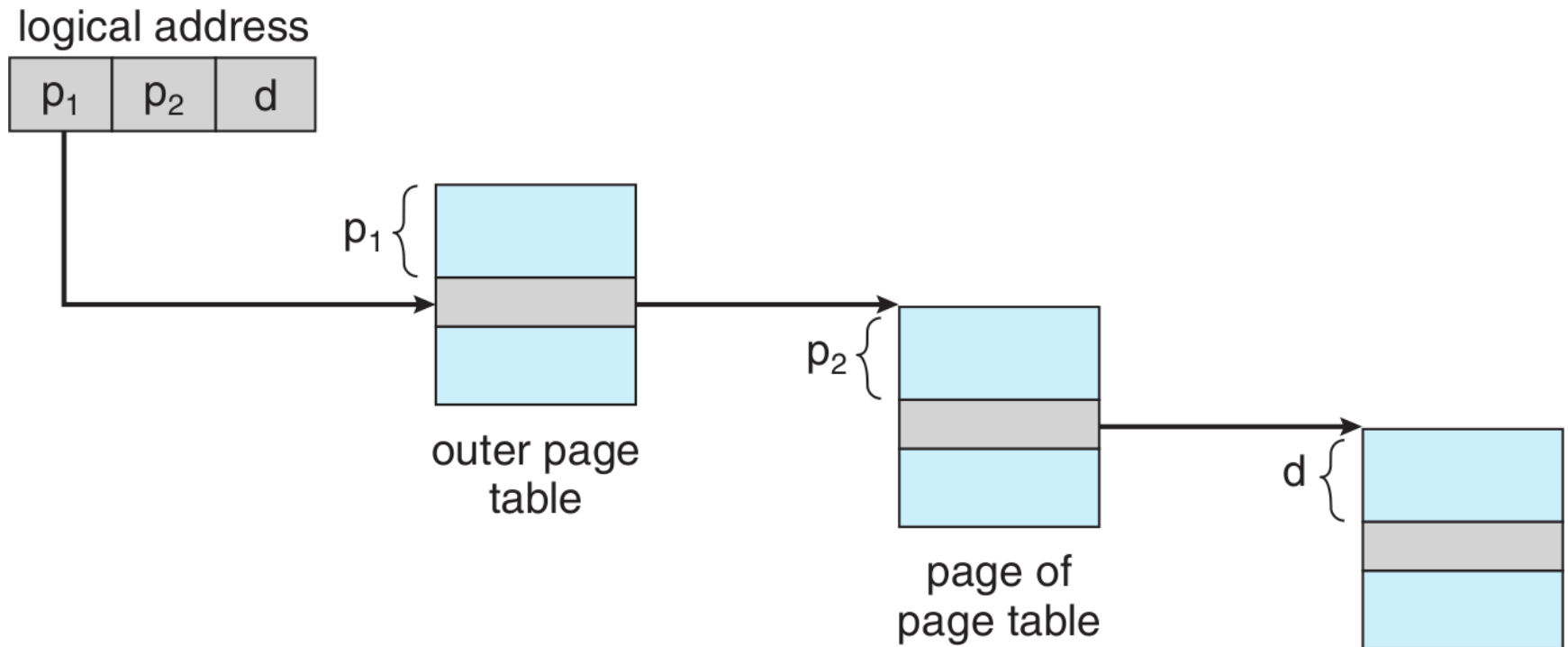
Partage de mémoire



Taille de tableau de memoire

- 32 bits - 4 GB Mémoire
 - taille de page = 4 KB
 - 2^{20} pages de mémoire
 - entrée de tableau de mémoire = 4 B
- Taille $4 * 2^{20} = 4 \text{ MB}$ / processus
- 64 bits?

Pagination hiérarchique



MÉMOIRE VIRTUELLE

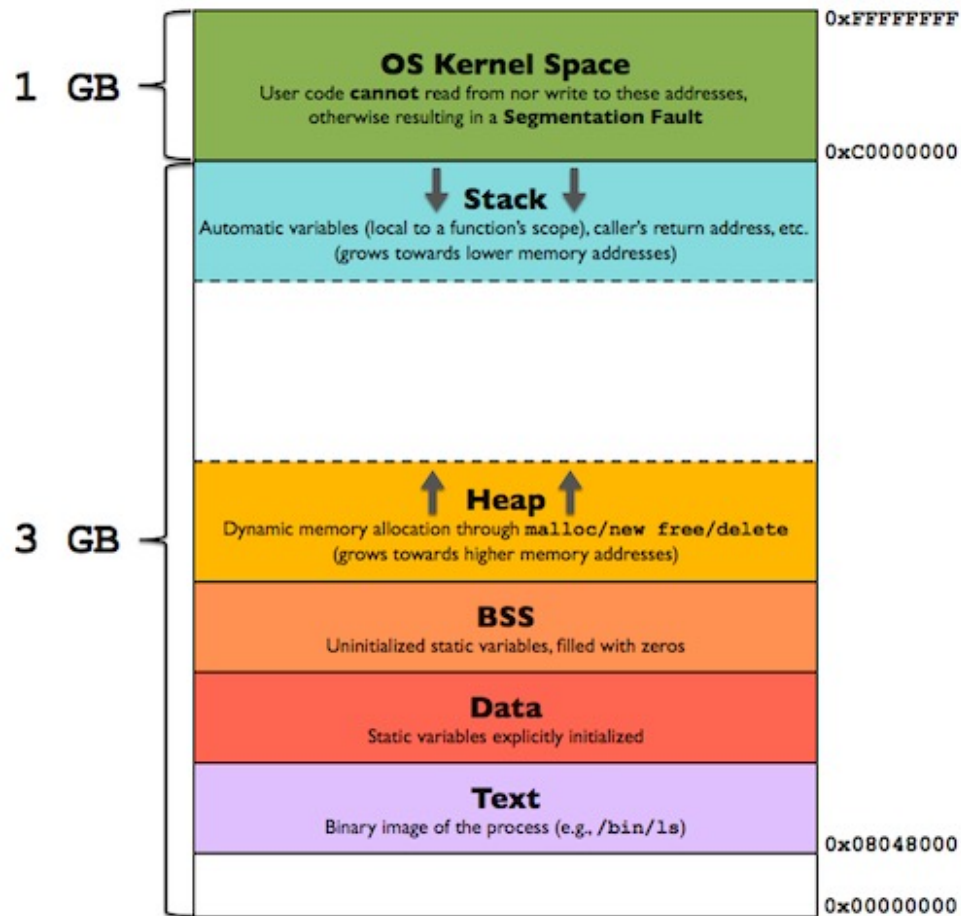
Memorie Virtuelle

- Séparation entre
 - Espace physique (RAM)
 - Espace logique (mémoire vue par un processus)

Avantages es désavantages

- Partage de mémoire
- Toutes les adresses sont disponibles
- Utilisation de plus de mémoire que disponible
- Double accès au mémoire
 - tableau et adresse effective
- Support matériel
- Le SE doit avoir un component spéciale

Espace d'adresse simplifié



FAUTE DE PAGE

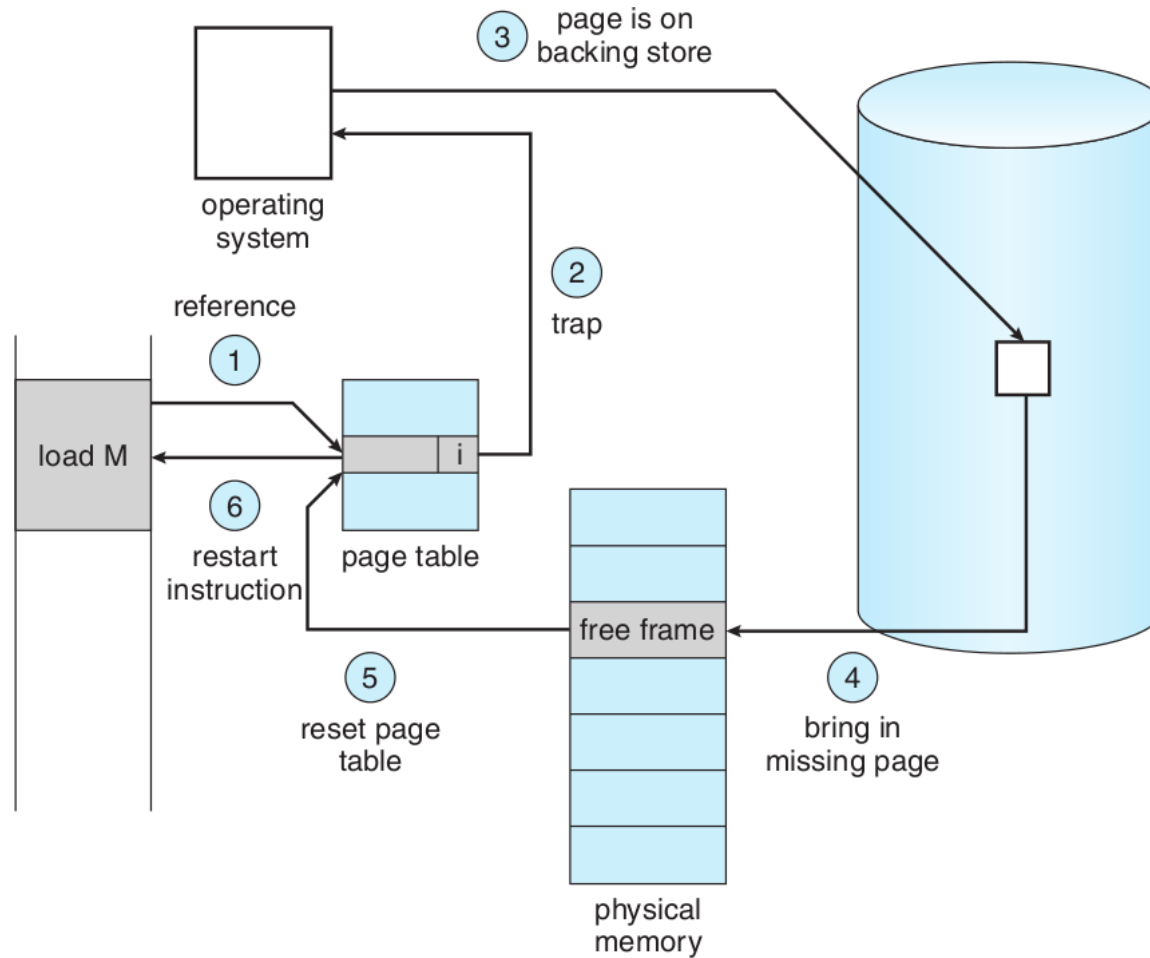
Faute de Page

- Accès a un page
 - non mappé
 - invalide
 - sans droits d'accès
- Exception de CPU
 - est exécuté le handler d'exception

Exemples de fautes de page

- non mappe
 - page en demande
 - swap
- invalide
 - la page virtuelle n'est pas alloué
- sans droits d'accès
 - read-only
 - page de SE
 - copy-on-write

Faute de page



PAGE EN DEMANDE

Etapes de allocation de mémoire

1. Allocation de page virtuelle
 - entre dans le tableau de pages

2. Allocation de page physique
 - entre dans le tableau de cadres (frames)

3. Mappage de page
 - relier la page et le cadre

Allocation de mémoire

- **réserve** (malloc)
 - allocation de page virtuelle (sans cadre en RAM)
 - marqué *invalid*
- **commettre** (premier accès, $*p = \dots$ ou $\dots = *p$)
 - faute de page mineur
 - allocation de cadre physique (frame) en RAM
 - relier la page et le cadre
 - marqué *valide*

SWAP

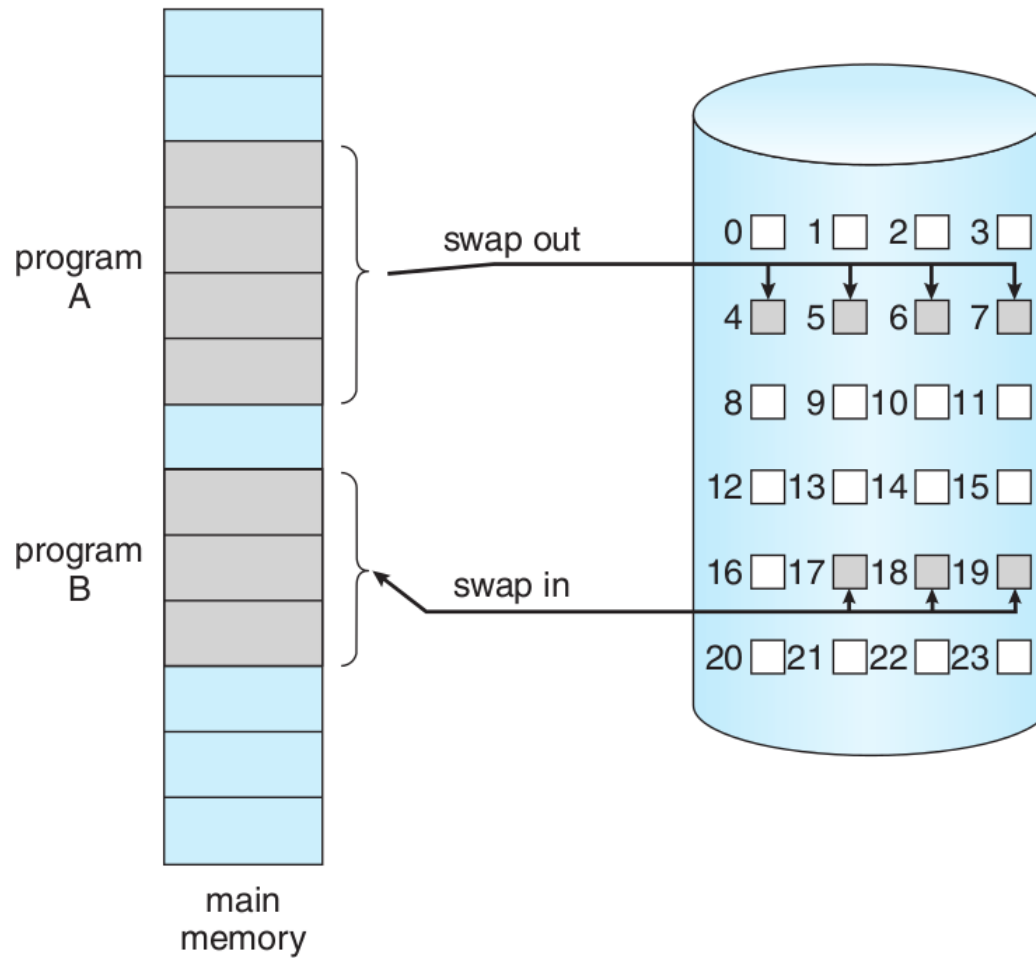
Swap

- Nous pouvons utiliser plus de mémoire que la RAM disponible
- Une partie de pages sont mémorisée sur un autre système de stockage
 - HDD
 - SSD
- Peut être lente

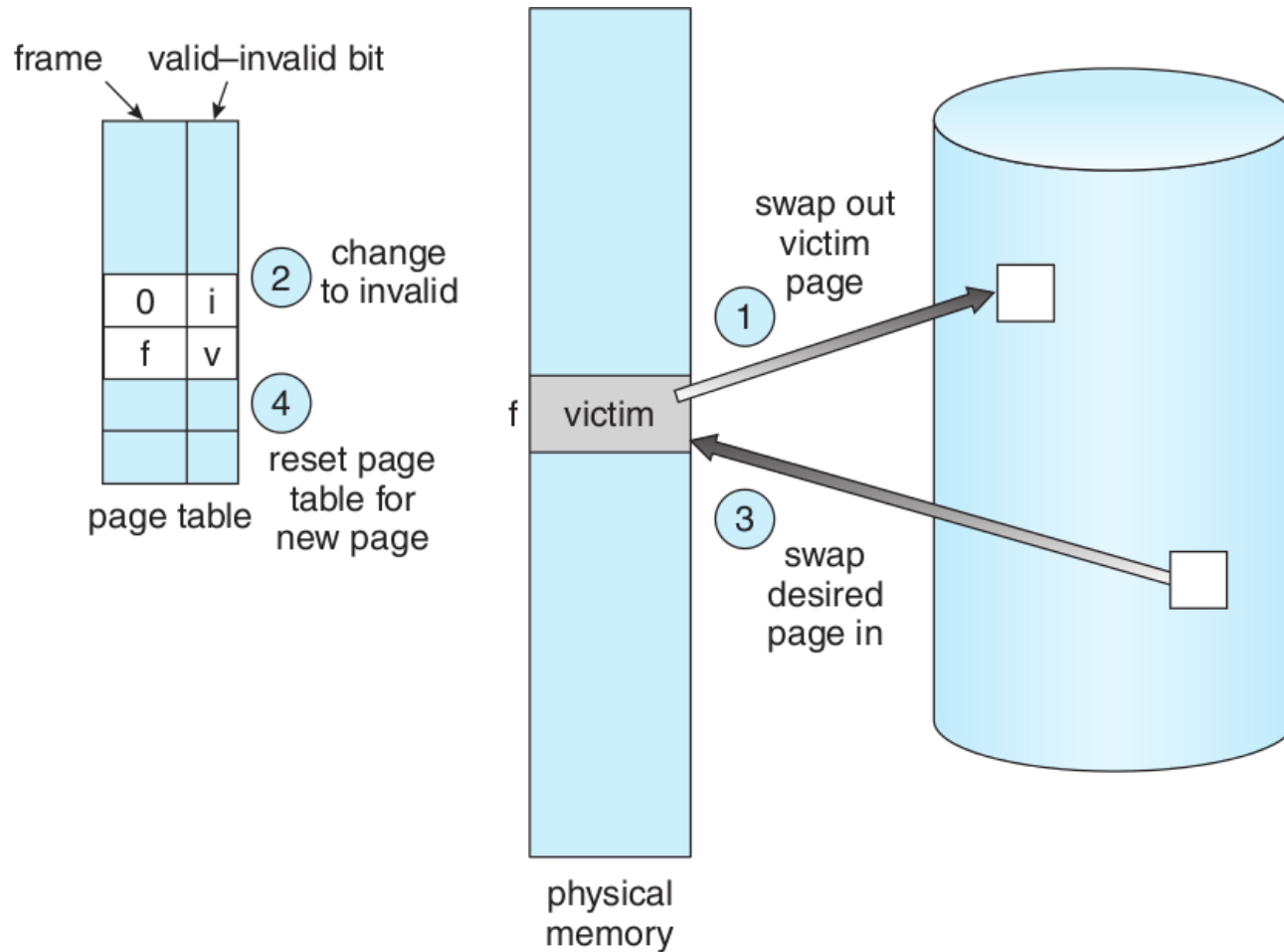
Swap en Windows et Linux

- Windows
 - C:\pagefile.sys
- Linux
 - Partition différentes

Swap



Remplacement de page



Les pages déposés en swap?

- FIFO
- LRU
 - Least Recently Used
- Second Chance
 - mieux FIFO

FIFO

- queue de pages
- la premier page dans la queue est déposé en swap (swapped out)
- la page chargée du swap est placée à la fin de queue (swapped in)

Example

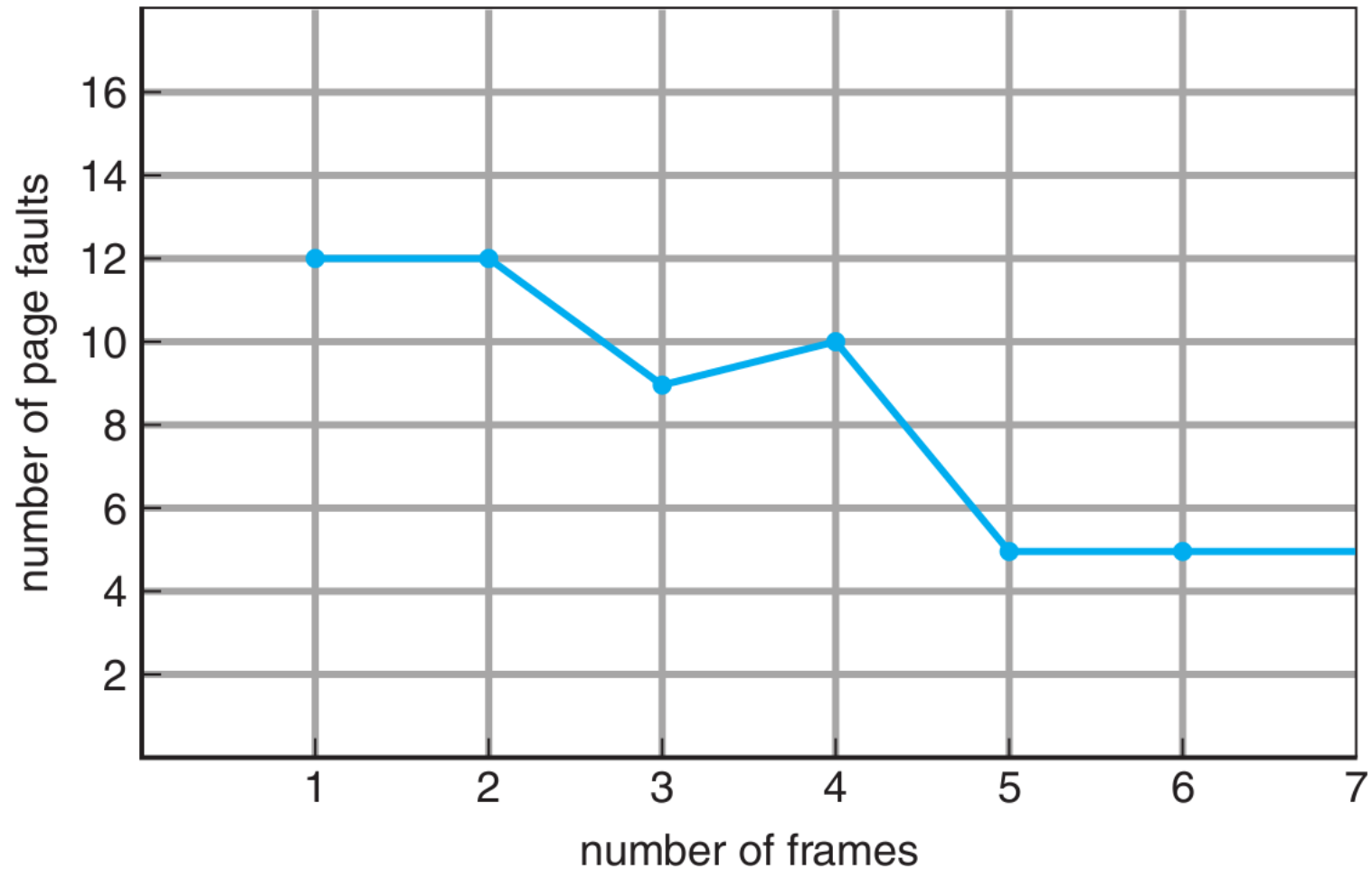
reference string

7 0 1 2 0 3 0 4 2 3 0 3 2 1 2 0 1 7 0 1

7	7	7	2																
	0	0	0																
		1	1																

page frames

L' anomalie de Belady



- l'idée idéal: remplace la page qui ne va être utilise pour le plus long temps
- attache sur les pages la date de dernier d'accès
- la page avec le pus vieux date d'accès est dépose en swap

Example

reference string

7 0 1 2 0 3 0 4 2 3 0 3 2 1 2 0 1 7 0 1

7	7	7	2		2		4	4	4	0		1		1		1
	0	0	0		0		0	0	3	3		3		0		0
		1	1		3		3	2	2	2		2		2		7

page frames

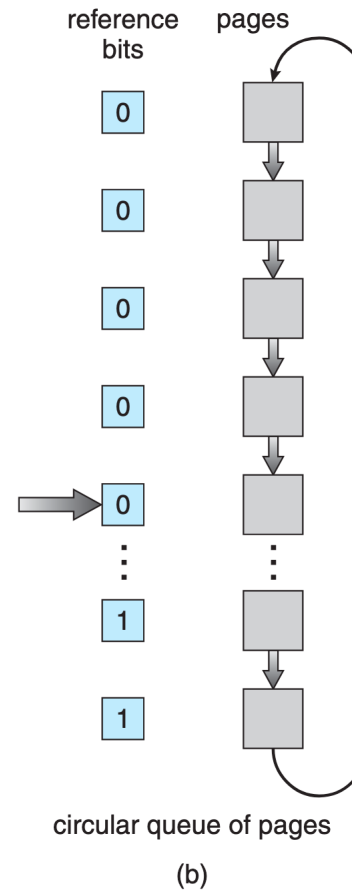
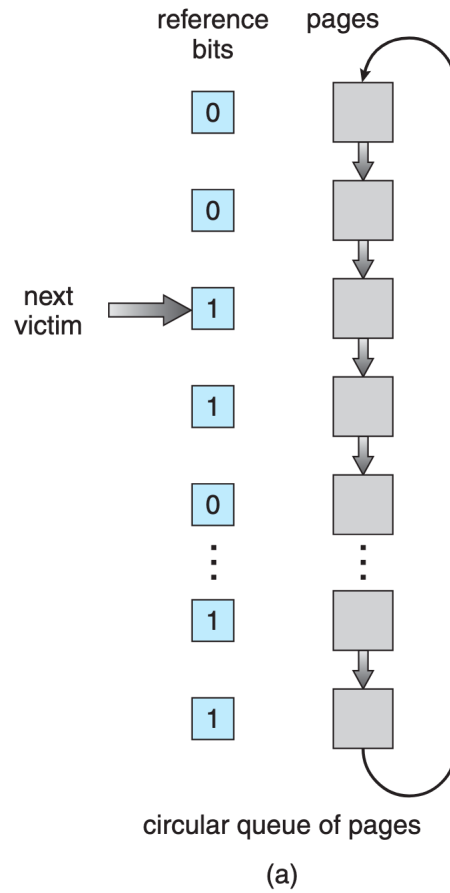
Difficulté d'implémentation

- Le matériel doit ajoute la date de dernier accès
 - avec une implémentation en logiciel le temps d'accès est 10x
- Beaucoup de mémoire pour mémorise la date d'accès

Second Chance

- Page de mémoire a deux paramètres (bits)
 - R – Référencé (*read* ou *write*)
- Queue de pages
 - Si en page a $R = 0$, la page est déposé en swap
 - Si en page a $R = 1$, R este fait 0 et la page este placée a la fin de la queue

Example



NRU - Not Recently Used

- Page de mémoire a deux paramètres (bits)
 - R – Référencé (*read*)
 - M – Modifie (*write*)
- Nous choisissons la page qui a la classe minimale de round robin

Classe	R	M
1 (probabilité minimale d'être utilisé)	0	0
2	0	1
3	1	0
4 (probabilité maximale d'être utilisé)	1	1

Page residente

- la page se trouve en RAM
- RSS de processus
 - Resident Set Size (les pages qui sont présente en RAM)
- Allocation résidente
 - La page ne peut être déposé en swap
 - mémoire de noyau

```
int mlock(const void *addr, size_t len);
```

```
int munlock(const void *addr, size_t len);
```

addr – multiple de page

len – multiple de page

Swap Trashing

- Le système a besoin de beaucoup de temps pour
 - Déposé de pages dans le swap
 - Prendre de pages depuis le swap

COPY-ON-WRITE

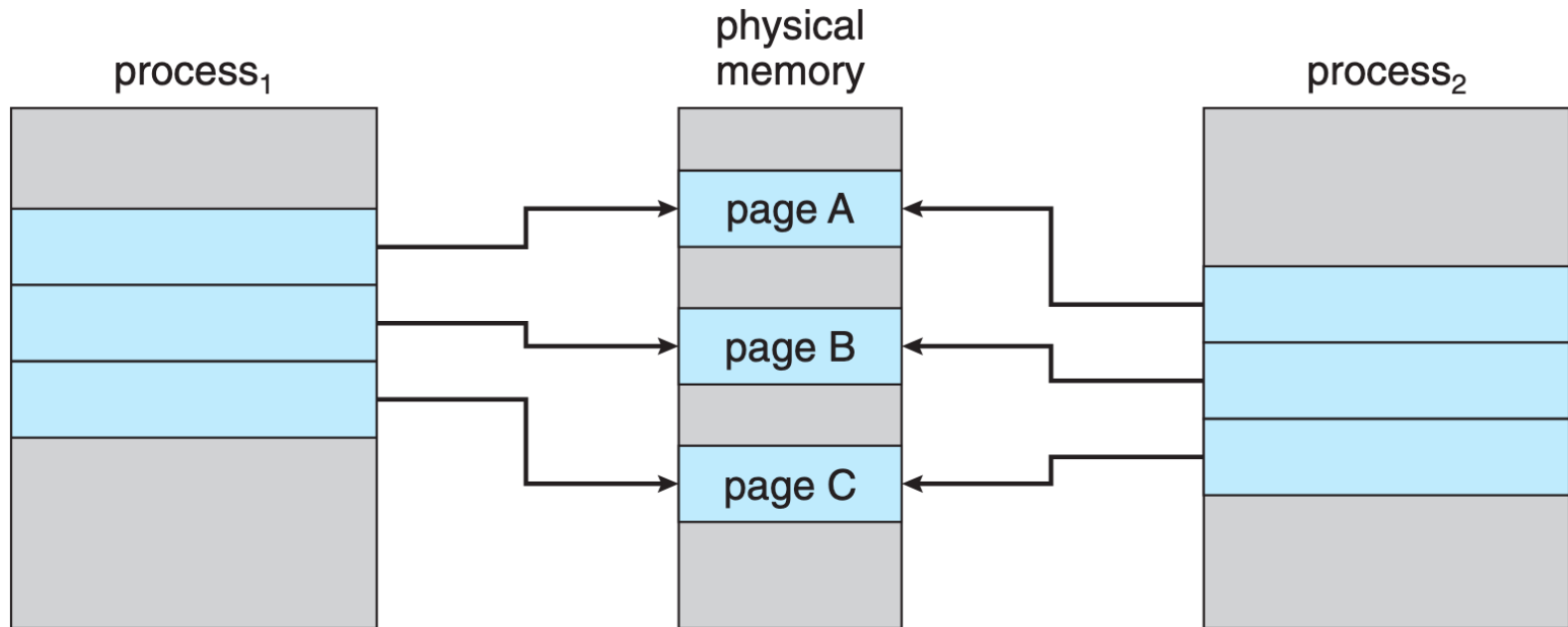
Nouveau Processus - UNIX

- **fork**: créer un nouveau processus (enfant)
(presque identique au processus parent)
 - copie de mémoire
- **exec**: chargement d'informations d'un exécutable dans la mémoire du processus enfant
 - mémoire nouveau

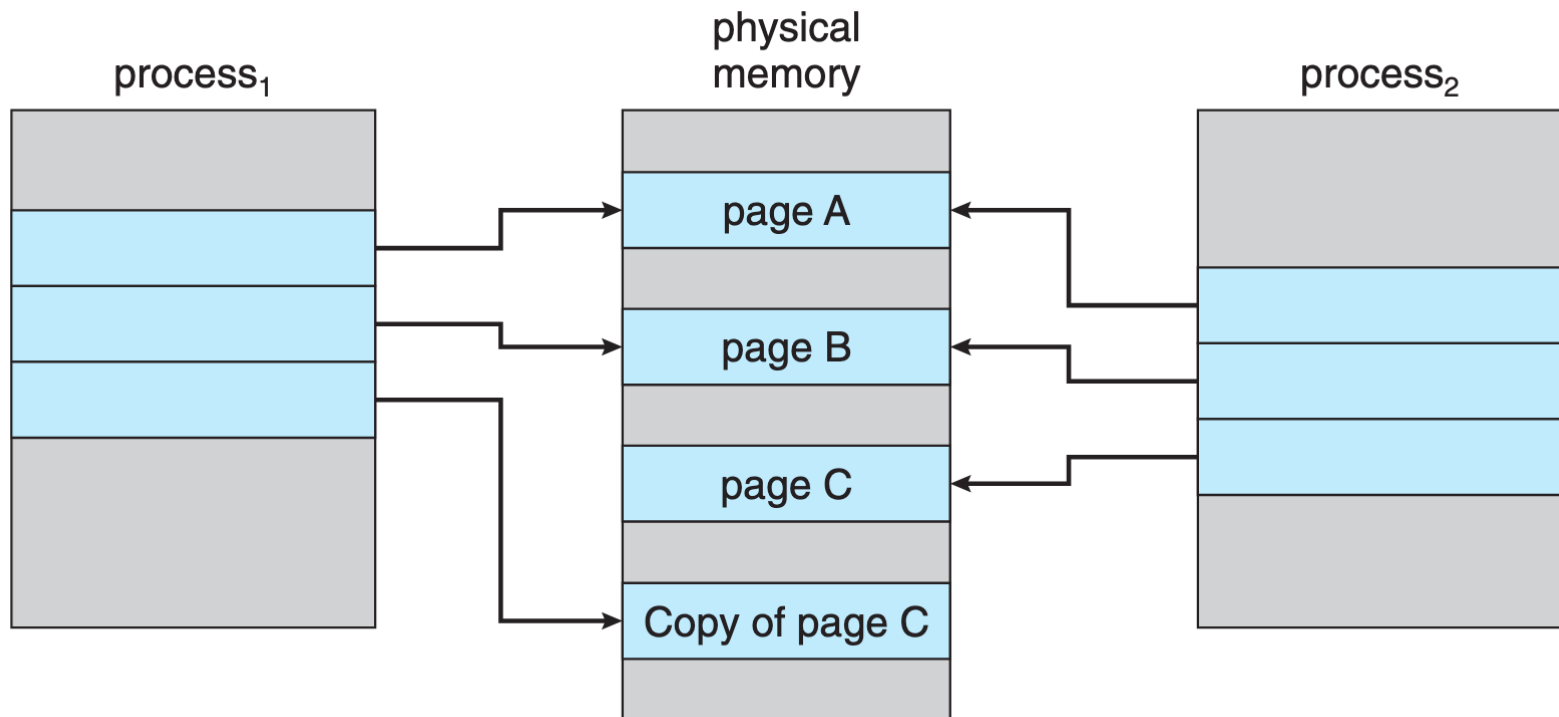
Copy on write - fork

- seule le tableau de pages est copiée
- toutes le pages sont marqué READ-ONLY
 - le deux processus partage la mémoire
- Si en processus voit écrire
 - faute de page
 - la page de mémoire este copie
 - Le deux processus ont de pages différentes

Example



Example

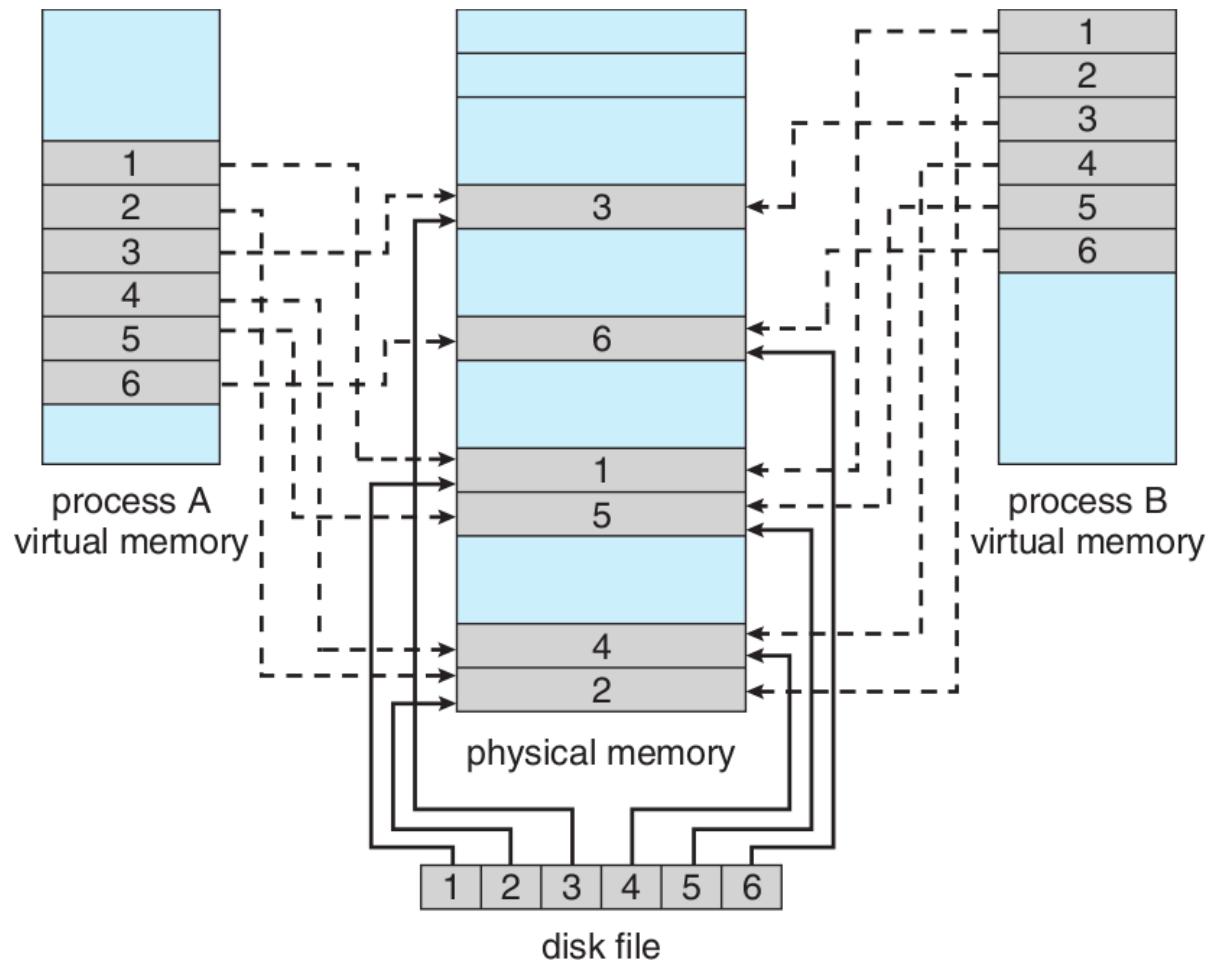


Mappage de fichiers

- une partie d'un fichier est mappée sur une adresse mémoire
- accès aux fichiers uniquement par accès à la mémoire
- écrire dans le fichier n'est pas immédiat

chargement des exécutables et des
bibliothèques

Example



Linux

```
void *mmap(void *addr, size_t len, int prot,  
           int flags, int fd, off_t offset);
```

```
int munmap(void *addr, size_t len);
```

```
int msync(void *addr, size_t len, int flags);
```

- PROT_NONE Pages may not be accessed.
- PROT_READ Pages may be read.
- PROT_WRITE Pages may be written.
- PROT_EXEC Pages may be executed.

Example

```
int fd = open ("fichier.dat", "rw");
char *p = NULL;
if (fd >= 0) {
    p = mmap (NULL, 4096, PROT_WRITE, MAP_SHARED, fd, 0);
    if (p!=NULL) {
        strcpy (p, "sde");
        msync (p, 4096, M_SYNC);
        munmap (p, 4096);
    }
    close (fd);
}
```

- Adresse logique
- Adresse physique
- Pagination
- TLB
- Tableau de pages
- Faute de page
- Demande de page
- Swap
- Remplacement de pages
- FIFO
- LRU
- Second Chance
- NRU
- Copy on write
- Mappage de mémoire

Questions

