Couplage de fichiers en mémoire virtuelle (Linux)

Idée : utiliser le mécanisme de mémoire virtuelle
pour gérer les mises à jour du contenu d'un fichier

Principe

Associer un (fragment de) fichier à une zone (contiguë) de mémoire virtuelle

Le (fragment de) fichier est vu par le mécanisme de mémoire virtuelle
comme la copie sur disque de la zone de mémoire virtuelle à laquelle il est associé

Avantages

- plus efficace (évite une recopie entre l'espace utilisateur et les caches système)
- programmation transparente, plus simple :
  - ◊ pas d'ouverture/fermeture de fichiers
  - ◊ fichier = (grosse) variable → pas d'opérations de lecture/écriture explicites

Définition et gestion du couplage

Création : mmap(...) `(sys/mman.h)`

L'opération `mmap()` permet d'associer une zone de mémoire virtuelle à une zone de fichier.

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

- `addr` : adresse zone mémoire obtenue / adresse zone mémoire demandée (NULL = pas de préférence)
- `len` : taille mémoire demandée
- `prot` :
  - **PROT_READ** accès permis en lecture
  - **PROT_WRITE** accès permis en écriture
  - **PROT_EXEC** accès permis en exécution
  - **PROT_NONE** accès interdit (→ SIGSEGV ou SIGBUS)
- `flags` : **MAP_SHARED**, ou **MAP_PRIVATE** (obligatoire)
  - **MAP_ANON** pas de fichier existant
  - **MAP_FIXED** adresse demandée impérative
- `fd` : descripteur fichier couplé (-1 = pas de fichier)
- `offset` : origine du fragment de fichier couplé

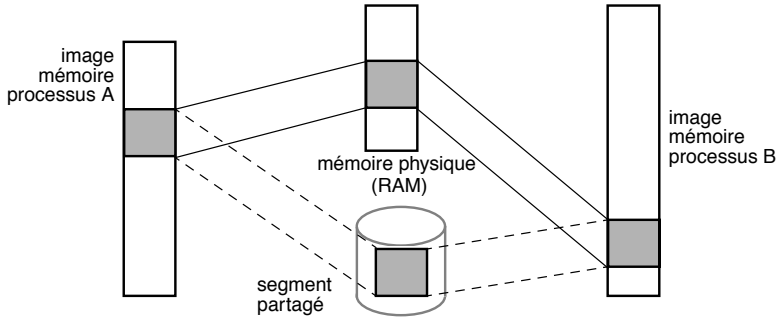
Exemples

```
long pagesize = sysconf(_SC_PAGESIZE);
int cf = open("contenu",O_RDWR);
char* base = mmap(0,pagesize, PROT_WRITE|PROT_READ,MAP_SHARED,cf,0);
/* adresses [base,base+pagesize[ accessibles en lecture/´ecriture */

char* b = mmap(0,pagesize,PROT_WRITE|PROT_READ,MAP_SHARED|MAP_ANON,-1,0);
```


Partage de pages (indicateur `MAP_SHARED`)

Un même fragment de fichier peut être couplé à plusieurs zones de mémoire virtuelle, de processus différents, ce qui permet de définir un espace mémoire partagé.



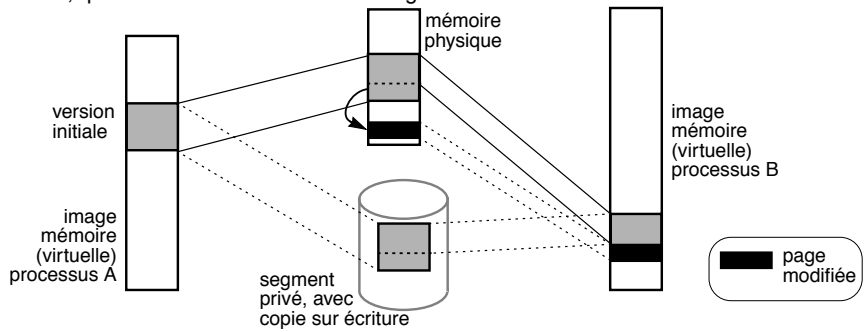
Une modification dans l'une des projections partagées d'un segment, ou sur le segment lui-même, est répercutée (immédiatement) sur :

- le segment (fichier)
- l'ensemble des projections partagées

Couplage d'un fragment de fichier (segment) en mode privé (indicateur `MAP_PRIVATE`) : mécanisme de copie sur écriture (copy on write)

Un segment couplé en mode privé privé se comporte comme un segment partagé tant qu'il n'est pas modifié : il peut être associé à une zone de la mémoire virtuelle de plusieurs processus.

Quand une page du segment est modifiée pour la première fois par un processus ayant couplé le segment en mode privé, une nouvelle page est allouée *en mémoire physique* et la modification est reportée sur cette seule page. La modification n'est ni répercutée sur le fichier, ni visible aux autres processus, qui continuent à voir la version originelle.



***Note** : l'effet d'une modification par un processus ayant couplé le segment en mode partagé sur les processus l'ayant couplé en mode privé n'est pas défini. La mise en œuvre la plus simple est de répercuter, pour les processus ayant couplé le segment en mode privé, mais ne disposant pas (encore) de version locale de la page.*

Découplage : munmap(...)

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

début de la zone mémoire à découpler

longueur de la zone à découpler (multiple de la taille d'une page)

Les références à la zone découplée engendrent des erreurs d'adressage

Mise à jour des droits d'accès à un segment : mprotect(...)

```
int mprotect (void * addr, size_t taille, int prot);
```

début de la zone mémoire concernée

longueur de la zone concerrnée

PROT_READ, PROT_WRITE , **PROT_EXEC**,**PROT_NONE**