Segments de mémoire partagée

Segment de mémoire partagée

- Les processus adressent des espaces mémoire séparés → une variable d'un processus ne peut pas être modifiée par un autre processus.
- possibilité de communication : fichier (régulier, tube, socket locale, etc.)
- autre possibilité : allouer une zone mémoire accessible en R/W par plusieurs processus

Identification

- Un segment de mémoire partagée est identifié à deux niveaux :
  - au niveau utilisateur : une clé numérique par segment
  - au niveau système : un entier
- Un segment de mémoire partagée est, comme les fichiers, muni d’un ID de propriétaire (uid) et de droits d’accès

Allocation

```
int shmget (key_t k, int size, int flag)
```

k clé

size nombre d'octets (arrondi au multiple d'une page)

flag IPC_CREAT et/ou 9 bits de permission

retour : un identifiant de segment de mémoire partagée

Attachement/détachement

```
char* shmat(int id, char* addr, int flag);
```

addr NULL pour attachement à une adresse quelconque

flag 0 ou SHM_RDONLY

retour : une adresse (le segment se gère comme une zone allouée par malloc)

```
int   shmdt (char* addr);
```

destruction si plus aucun attachement

Segment et processus

- Un segment peut être attaché plus d'une fois dans un même processus (par exemple, en lecture, puis en lecture/écriture) → plusieurs adresses différentes pour les mêmes données physiques.
- Après un `fork()`, le fils attache les mêmes segments aux mêmes adresses
- Après `exec()` ou `exit()`, les segments sont détachés automatiquement
- Conflits d'accès (deux processus qui accèdent en même temps à la même donnée) non gérés ; c'est au programmeur de le prévoir (sémaphores, mutex, etc.)

IPC

- Les segments de mémoire partagée font partie d'un ensemble de mécanismes appelés IPC (Inter Process Communication) spécifiques à UNIX (SystèmeV) :
  files de messages, sémaphores, mémoire partagée

- Files de messages, voir :
  `msgctl(2), msgget(2), msgrcv(2), msgsnd(2)`

- Sémaphores, voir :

  `semctl(2), semget(2), semop(2), semtimedop(2)`