Université de ***Batna 2***

Communication interprocessus : **Segment de Mémoire Partagée**

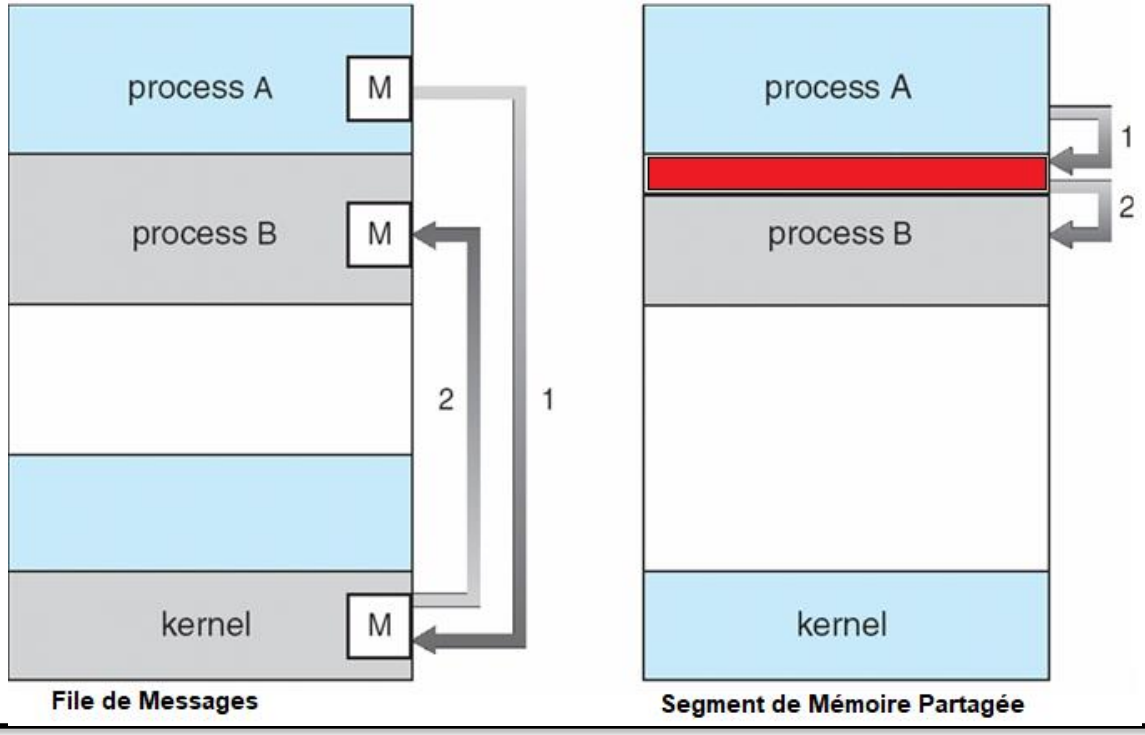
Département d'Informatique

Master ISI

Plan

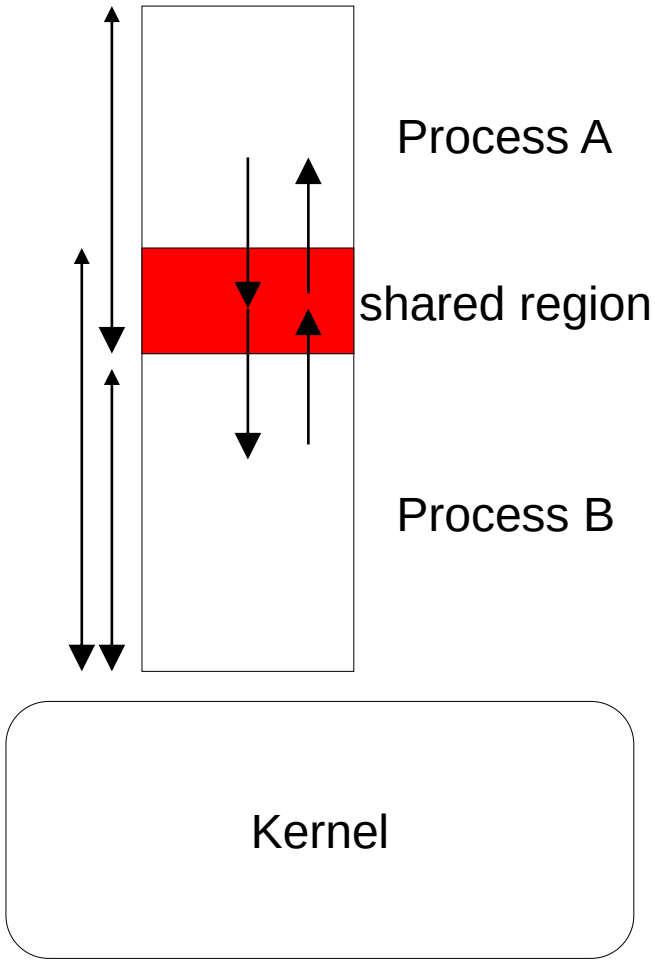
Introduction

- **Les IPC (*Inter Process Communication*) *UNIX* System V présentent trois outils de communication entre des processus situés sur une même machine.**
  - Files de messages.
  - **Segments de mémoire partagée.**
  - Sémaphores.



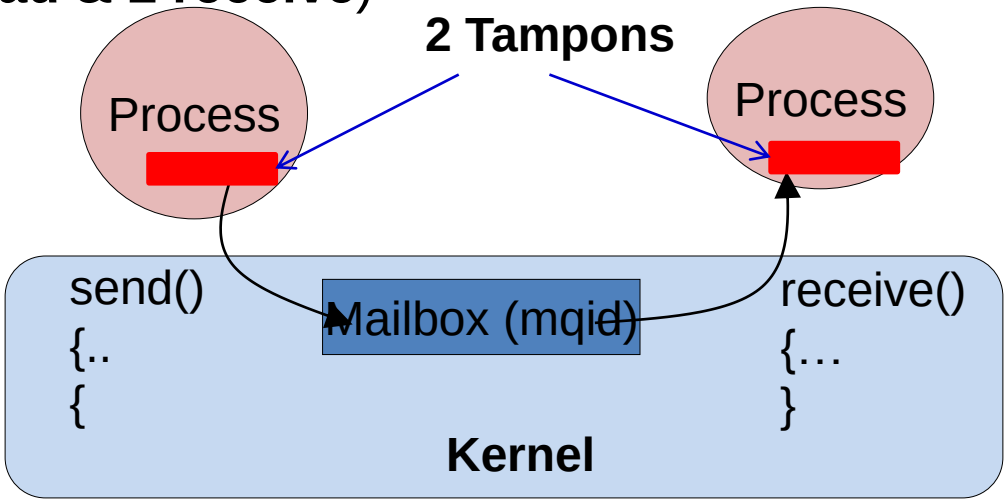
Segment de Mémoire Partagée

- **Principe**
  - En Anglais : **Shared Memory** (SM) Segment.
  - Est une région de mémoire paratgée établie entre 2 ou plusieurs processus.
  - L'établissement de cette région est faite par l'assistance du OS (kernel de linux)
  - Donc, les precessus peuvent diretement lire et écrire dans le SM comme il s'agit d'un accès aux variables locales.
    - L'accès à SM se fait indépendamment du kernel.
    - En conséquence, il **est rapide**



Segment de Mémoire Partagée

- **Quelle est la différence entre SM et les autres mécanismes?**
  - Le problème avec les **pipes**, **FdM** est que pour 2 processus les échanges d'information se passent à travers le Kernel (noyau).
  - l'envoie et la réception de l'information se fait via au moins 2 tampons (**2 read & 2 receive**)



  - ***Néanmoins***, pour les **SM** les échanges d'information se passent indépendamment du Kernel (accès directe).

Segment de Mémoire Partagée

- **Clef & Descripteur**
  - une clef (définie par le créateur), qui pourra être utilisée par les autres processus pour accéder à la SM.
  - un descripteur (identificateur), retourné à la création, et utilisé pour les manipulations du SM au sein du processus.
- **Un segment de mémoire partagée sera créé par un processus.**
- **Chaque processus voulant y accéder demandera à s'attacher**
- **Le segment, après vérification des droits, il disposera d'un pointeur vers ce segment.**
- **Les processus accédant devront gérer la synchronisation :**
  - exclusion et protection. Classiquement, ils utiliseront des sémaphores.
- **La destruction de l'espace devra être faite par un processus ayant le droit de destruction.**
- **En cas d'arrêt du système, le segment sera perdu : fonctionnement identique `a celui des files de message.**

Segment de Mémoire Partagée : Opérations

- **Les opérations réalisables sur un segment de mémoire partagée :**
  - Création d'un segment ;
  - Attachement (obtention d'un pointeur) ;
  - Détachement (abandon d'accès) ;
  - Contrôle des paramètres dont suppression (comme pour les files de messages).
- **Remarque : l'accès en lecture/écriture se fait de manière classique, en utilisant un pointeur. Il n'y a donc pas de primitives dédiées.**

Segment de Mémoire Partagée : Opérations

- **Création d'un SM**
  - include: <sys/types.h> <sys/ipc.h> <sys/shm.h>
  - int **shmget** (key_t **cle**, int taille, int option);
  - **cle** : clé obtenue avec **ftok()** ou IPC_PRIVATE
  - taille : indique la taille du segment (inférieure ou égale si segment existant) en octets
  - option : IPC_CREAT et IPC_EXCL et 9 bits d'autorisation
  - La fonction retourne un identificateur du segment **ou** une valeur négative en cas d'échec

Segment de Mémoire Partagée : Opérations

- **Création d'un SM (Exemple)**

```
struct uneChaine {
    char  c ;
    int     x, y ;
    struct uneChaine *suiv;
};
int shid = shmget (key, size_t (30 * sizeof(uneChaine)),
IPC_CREAT|0666);
```

permet de créer un segment avec les droits d'accès de lecture et écriture à tout processus de tout utilisateur.

```
int shid = shmget (key, size_t(0) , O RDONLY);
```

est une demande d'accès en lecture seule, en supposant que le segment existe.

Segment de Mémoire Partagée : Opérations

- **Création d'un SM (shmid_ds)**
  - Pour chaque SM, une structure de données gérée par le kernel se crée.
  - Elle permet d'enregistrer toutes les infos sur SM

```
/* One shmid data structure for each shared memory segment in the
system. */
struct shmid_ds {
  struct ipc_perm shm_perm;     /* operation perms */
  int shm_segsz;                /* size of segment (bytes) */
  time_t shm_atime;             /* last attach time */
  time_t shm_dtime;             /* last detach time */
  time_t shm_ctime;             /* last change time */
  unsigned short shm_cpid;      /* pid of creator */
  unsigned short shm_lpid;      /* pid of last operator */
  short shm_nattch;             /* no. of current attaches */
        /* the following are private */
  unsigned short shm_npages;    /* size of segment (pages) */
  unsigned long *shm_pages; /* array of ptrs to frames -> SHMMAX */
  struct vm_area_struct *attaches; /* descriptors for attaches */
};
```

Segment de Mémoire Partagée : Opérations

- **Attachement**
  - include: <sys/types.h> <sys/ipc.h> <sys/shm.h>
  - void ***shmat** (int shmid, void *shmaddr, , int option);
  - Retourne un pointeur sur le premier byte du segment. Valeur négative en cas d'échec.
  - ***shmaddr** : permet, s'il est différent de **NULL** de spécifier une adresse résultat choisie (forcée) par l'utilisateur : rare.
  - **Shmid** : un identificateur valide du SM.
  - **Option**: les droits d'accès, Le type d'accès par défaut est en lecture et écriture. Options permet de le modifier, par exemple de demander l'accès en lecture seule avec SHM RDONLY.

Segment de Mémoire Partagée : Opérations

- **Attachement**

  Exemple : Un segment contenant des entiers, **p** contient l'adresse de début du segment

  – int ***p**;

  – **p** = (int*) shmat (shid, NULL, 0);

  – Le pointeur **p** permet ensuite de gérer tous les emplacements du segment comme un tableau pour les opérations classiques de lecture ou écriture dans le segment.

Segment de Mémoire Partagée : Opérations

- **Détachement**
  - On abandonne l'accès en détachant l'espace commun.
  - int **shmdt** (const void * adrAtt)
    - adrAtt : adresse d'attachement.
  - Retourne (**0** en cas de réussite ou **-1** en cas d'échec),
  - A la fin du processus, tous les segments préalablement attachés, dans ce processus, sont détachés.

Segment de Mémoire Partagée : Opérations

- **Opérations de contrôle**
  - int **shmctl** (int shmid, int cmd, struct shmid_ds * buf);
    - **shmid** : Un identificateur valide
    - **cmd** : par exemple IPC_RMID pour supprimer le SM
    - **buf** : pour récupérer la table associée au SM

  - *Exemple*

    shmctl (shid, IPC_RMID, 0); {Supprime le segment d'identificateur shid}

Segment de Mémoire Partagée : Cas pratique

- **Exemple**

Fin

Merci pour votre attention