

## Master 1 Informatique - MCPR

## Modèles et Concepts du Parallélisme et de la Répartition

### Support de travaux pratiques

## 1 - Outils pour la synchronisation et la communication Posix

### IPC Posix : Sémaphores et segments de mémoire partagée

#### 1.1. Rappels

Lorsqu'un processus est créé, il est créé comme une copie presque parfaite de son « père » : il hérite de ses segments de code, de données et de pile. Mais, il en diffère aussi ; notamment par son pid, son ppid et le fait que ses segments de données et de pile ne sont pas partagés avec son père. En d'autres termes, une variable déclarée localement ou globalement, n'est **jamais partagée entre un processus père et un processus fils**. Pour communiquer une information, des processus peuvent utiliser le mécanisme primitif `exit()`/`wait()` (communication à sens unique, du fils vers son père, lorsque ce fils meurt) ou des tubes de communication. Dans ces deux cas, les processus doivent **avoir un lien de parenté** afin de pouvoir communiquer.

Lorsque les processus n'ont plus aucun lien de parenté, ces mécanismes ne suffisent plus et il faut avoir recours à des outils appelés IPC (Inter Process Communication).

#### 1.2. Généralités sur les IPC

Les outils IPC permettent de faire communiquer et se synchroniser des processus indépendants (mais s'exécutant sur la même machine) et sont au nombre de trois :

1. Les segments de mémoire partagée (*shared memory*) permettent aux processus de partager des données et ainsi de communiquer rapidement ;
2. Les files de messages (*message queues*, ou boîtes aux lettres) permettent à la fois la communication et la synchronisation ;
3. Les sémaphores servent à la synchronisation entre processus.

Ces IPC ont des caractéristiques communes :

- Il est nécessaire d'inclure les fichiers `sys/types.h` et `sys/ipc.h` ;
- Tout comme un fichier, un IPC possède un identificateur externe (qui l'identifie de manière unique sur la machine) et un identificateur interne (qui l'identifie au sein d'un processus). Le premier est de type `key_t` et le second de type `int` ;
- Un IPC peut être utilisé par n'importe quel processus, pouvant appartenir à des utilisateurs différents, un processus peut donc vouloir protéger un IPC contre son utilisation, voire sa destruction par un autre processus. Pour cela, un IPC possède des droits d'accès ;
- Lors de la création d'un IPC, on peut spécifier qu'il ne sera utilisable que par le processus créateur et sa descendance si son identificateur externe est une constante prédéfinie (`IPC_PRIVATE`) ;
- Enfin, les ressources IPC sont *persistantes*, elles continuent à exister sur une machine tant qu'on (le propriétaire ou l'administrateur système) ne les a pas détruites. Or, le nombre d'IPC

- sur une machine est limité au niveau d'un utilisateur, mais aussi au niveau global. Il est donc *nécessaire* de les *détruire* lorsqu'elles ne sont plus utilisées.

#### 1.3. Les sémaphores UNIX System V

```
#include <sys/types.h>
```

```
#include <sys/ipc.h>
```

```
#include <sys/sem.h>
```

##### Ensemble de sémaphores

Le couple (numéro interne de l'ensemble, numéro du sémaphore dans l'ensemble) identifie un sémaphore dans cet ensemble. Le numéro du 1<sup>er</sup> sémaphore d'un ensemble est 0.

##### Créer / Ouvrir un ensemble de sémaphores

```
int semget ( key_t key, int nsems, int semflg ) ;
```

##### Création

Dans le cas où le nom externe (key) vaut `IPC_PRIVATE` ou si le bit `IPC_CREAT` de `semflg` est à 1.

Le bit `IPC_EXCL` a un rôle important :

- 1: la création échoue si le nom externe est déjà utilisé.
- 0: le processus obtient le numéro interne d'un ensemble de sémaphores déjà créé.

9 bits de faible poids de `semflg` = droits d'accès aux sémaphores créés.

##### Ouverture

L'ensemble a été créé auparavant et on souhaite l'utiliser.

`IPC_EXCL` et `IPC_CREAT` ne doivent pas être tous les deux positionnés.

`nsem` doit être 0 ou inférieur ou égal à la valeur précisée lors de la création.

##### Contrôler les attributs d'un ensemble de sémaphores

```
union semun {  
    int          val;          /* Valeur de SETVAL */  
    struct {      semid_ds     *buf;      /* Pour IPC_STAT & IPC_SET */  
        ushort_t  *array;      /* Tableau pour GETALL & SETALL */  
    };  
};
```

```
int semctl( int semid, int semnum, int cmd, union semun arg );
```

Valeurs possibles pour `cmd` :

- `SETVAL` : `arg.val` est affecté à la valeur du sémaphore défini par (`semid`, `semnum`) ;

- SETALL : Les valeurs contenues dans le tableau `argarray` sont affectées aux valeurs des sémaphores de l'ensemble `semid` ;
- IPC\_RMID : La primitive `semctl` détruit l'ensemble de sémaphores `semid` ; tout processus en attente derrière un sémaphore de cet ensemble est alors réveillé et averti de cet événement (voir la primitive `semop`).

### Bloquer / réveiller un processus

```
struct sembuf {
    u_short  sem_num;    /* N° sémaphore dans l'ensemble */
    short     sem_op;     /* Opération sur le sémaphore */
    short     sem_flg;    /* Indicateurs */
};
```

- ```
int semop(int semid, struct sembuf *array, size_t nops);
```
- Le champ `sem_num` est le numéro, dans l'ensemble `semid`, du sémaphore concerné par l'opération.
  - Le champ `sem_op` de `sembuf` décrit l'opération à effectuer :
    - `sem_op > 0` : `sem_op` tickets supplémentaires dans le distributeur (opération V) ;
    - `sem_op = 0` : Blocage de l'appelant tant que le nombre de tickets n'est pas nul. Le bit `IPC_NOWAIT` de `sem_flg` joue le même rôle que dans l'opération P.
    - `sem_op < 0` : Demande de `sem_op` tickets (opération P). L'appelant est bloqué tant que tous ces tickets ne sont pas disponibles. Le blocage est évité si le bit `IPC_NOWAIT` du champ `sem_flg` est positionné. Dans ce cas, l'appelant poursuit son exécution, informé du non aboutissement de sa demande (-1 comme valeur retournée par la primitive `semop` et `EAGAIN` (11) dans la variable `errno`).

## 1.4. Les segments de mémoire partagée

```
#include <sys/shm.h>
```

### Créer / Ouvrir un segment de mémoire partagée

```
int shmget (key_t key, size_t size, int shmflg);
```

Renvoie l'identificateur du segment de mémoire partagée associé à la valeur de l'argument `key`. Un nouveau segment mémoire, de taille (en octets) `size` est créé si `key` a la valeur `IPC_PRIVATE` ou si aucun segment de mémoire partagée n'est associé à `key`, et `IPC_CREAT` est présent dans `shmflg`.

### Contrôler / détruire un segment de mémoire partagée

```
int shmctl(int shmId, int cmd, struct shmId_ds *buf);
```

La valeur du paramètre `cmd` spécifie l'opération de contrôle à effectuer :

- `IPC_STAT` : copie du descriptif du segment dans `buf` ;

- `IPC_SET` : modification de certaines caractéristiques du segment (propriétaire, groupe, droits, etc.) à partir des valeurs contenues dans `buf` ;
- `IPC_RMID` : destruction du segment ;
- `SHM_LOCK` : verrouillage du segment en mémoire centrale ;
- `SHM_UNLOCK` : déverrouillage du segment de mémoire partagée de la mémoire centrale.

### Attacher / détacher un segment de mémoire partagée

```
void *shmat (int shmId, const void *shmaddr, int shmflg);
```

Attache le segment de mémoire partagée identifié par `shmId` au segment de données du processus appelant et retourne l'adresse d'attachement. `shmaddr` est une adresse à laquelle le segment de mémoire doit être attachée ou NULL si le choix est laissé au système. `shmflg` indique que cette adresse doit être arrondie (`SHM_RND`) et/ou que le segment est en lecture seule (`SHM_RDONLY`).

```
int shmdt (const void *shmaddr);
```

Détache le segment de mémoire partagée situé à l'adresse indiquée par `shmaddr`.

## 1.5. Commandes UNIX et IPC

### Connaitre l'état des ressources IPC : `ipcs` (`ipc status`)

`ipcs` fournit des informations sur l'usage des ressources IPC pour lesquelles le processus appelant a accès en lecture. Voir le man.

### Suppression des ressources IPC : `ipcrm` (`ipc remove`)

```
ipcrm [-q msqid] [-m shmId] [-s semId] [-Q msgkey] [-M shmkey] [-S semkey] ..
```

Voir le man.

Remarque : -n : relatif aux segments de mémoire partagée (`shared memory`) ; -q : relatif aux files de messages (message *queue*) ; -s : relatif aux ensembles de sémaphores

### Créer une clé d'IPC « unique »

```
key_t ftok(const char *path, int id);
```

Cette fonction permet de créer un identificateur externe d'IPC (voir man).