Les commandes, fonctions et appels systèmes importants :

- ipcs(1) et ipcrm(1) permettent respectivement de visualiser et de supprimer les segments de mémoire partagée (ainsi que les sémaphores et les files de messages, non abordées dans ce TD) ;
- ftok(3C) génère une clé nécessaire à l'appropriation d'un objet IPC ;
- shmget(2) récupère un identificateur de segment de mémoire partagée ;
- shmat(2) attache un segment de mémoire partagée dans l'espace de mémoire du processus ;
- shmdt(2) détache un segment de mémoire ;
- shmctl(2) opérations diverses telles que la suppression définitive du segment.

Après sa création, un segment de mémoire partagée existe jusqu'à sa destruction explicite. Il faut donc ne pas oublier de le détruire lorsqu'il n'est plus utilisé.

Rappel sémaphore:

On peut envoyer un signal à un autre processus en appelant la primitive qui envoie le signal de numéro signum au processus de PID pid.

int kill(int pid, int signum)

Pour spécifier qu'une fonction C doit être appelée lors de la réception d'un signal non mortel, on utilise :

void (*signal(int signum, (void *handler)(int))))(int);

L'argument handler est une fonction C qui prend un entier en argument et sera automatiquement appelée lors de la réception du signal par le processus. Voici un exemple de fonction traitante :

```
void traite_signal_usr1( int signum )
{
printf("signal %d recu.\n", signum );
}
```

Cette fonction est installée par l'appel :

signal(SIGUSR1, traite_signal_usr1);

Pour forcer un processus à s'endormir en attente de la réception d'un signal, on utilise :

int pause(void).

Exercice 1

Ecrire un premier programme C permettant de créer une zone de mémoire partagée (SHM) puis créer deux programmes C permettant l'un d'écrire une chaîne de caractères dans cette SHM, l'autre permettant d'afficher le contenu de la SHM.

Primitives :

int shmget(key_t key, int size, int shmflg) ; permet soit de créer une SHM soit de récupérer un shmid pour une clef donnée (auquel cas size et shmflg ne sont pas utilisés).

char *shmat(int shmid, char *shmaddr, int shmflg) ; permet d'attacher la SHM identifiée par un shmid au segment de données du processus appelant.

int shmdt(char * shmaddr) ; permet de détacher la SHM du processus appelant qui ne pourra donc plus y accéder.

Exercice 2

Un processus père p1 dispose d'un tableau de 10 caractères (contenant une chaîne quelconque que nous nommerons `A`). De même, un processus fils `p2` dispose d'un tableau de 10 caractères (contenant une chaîne quelconque que nous nommerons `B`). Le but est d'échanger les chaînes respectives de `p1` et `p2` au moyen d'un unique segment de mémoire partagée de taille 15 caractères. Ecrire les codes de ces deux processus pour que chacun récupère les données de l'autre séquentiellement. Le processus p1 écrit ses données en SHM et envoie un signal SIGUSR1 au processus p2. Le processus p2 n'ira lire les données de p1 (et les afficher) puis les remplacer par les siennes qu'à la réception du signal SIGUSR1 puis à son tour envoie un signal SIGUSR2 au processus p1. Enfin à la réception du signal SIGUSR2 le processus p1 lit les données de p2 les affiche et meurt.