TD Programmation Système

SMI S6

Série 4 : Mémoires Partagées

Exercice 1 : Communication par message

Deux processus P1 et P2 sans liens désire réaliser la communication suivante :

- Le processus P1 récupère un message passé en paramètre, l'envoi au processus 2
- Le processus P2 récupère le message envoyé par P1 et l'affiche en majuscule

On suppose que le message envoyé par P1 est une chaine de caractères.

Ecrire en C les programmes P1 et P2 en utilisant les mémoires partagées.

Indication :

Les deux processus partagent un segment de mémoire de type chaine de caractères de taille 1024 octet. P1 crée la mémoire partagée l'initialise. Le processus 2 récupère la chaine du segment de mémoire partagée et l'affiche en majuscule.

Appels Systèmes: shmget(), shmat(), shmctl(),shmdt()

Exercice 2 : communication unidirectionnelle

Deux processus P1 et P2 sans liens parenté partagent un segment de mémoire de type entier.

1. On désire avoir l'affichage suivant à l'exécution

P1 : 2	4	6	8	10
P2 : 13	16	19	22	25

Ecrire les deux programmes C sous Unix implémentant P1 et P2.

Indication :

Le processus P1 crée la mémoire partagée l'utilise pour afficher les multiples de 2 jusqu'au premier multiple de 5 et il arrête son exécution, cède la main au processus P2 qui accède à la mémoire récupère la dernière valeur l'utilise pour afficher la suite des nombres obtenus en ajoutant 3 jusqu'au premier multiple de 5.

Exercice 3 : communication bidirectionnelle et synchronisation

On désire reprendre l'exercice 2, pour réaliser une communication bidirectionnelle. Soit P1 et P2 deux processus qui partagent un segment de mémoire représentant un entier N et s'exécutant en parallèle et permettant l'affichage suivant :

Terminal 1

```
P1 : 2 4 6 8 10

P1 : 27 29 31 33 35

P1 : 52 54 56 58 60

.....................

.....................
```

Terminal 2

```
P2 : 13 16 19 22 25

P2 : 38 41 44 47 50

P2 : 63 66 69 72 75

.................

.................
```

Ecrire les deux programmes C sous Unix implémentant P1 et P2.

Indication :

Les deux processus se synchronisent par l'envoi de message en utilisant la fonction kill(). Chacun doit avoir connaissance du pid de l'autre, une mémoire partagée et alors nécessaire pour pouvoir stocker les deux pid et l'entier N.

```
typedef struct MP{ int pid1 ;
                   int pid2 ;
                   int N ;
                   }MP ;
```

Appels Systèmes : shmget(), shmat(), shmctl(),shmdt(), kill, pause().