Université Ibn Tofail
Faculté des Sciences

Le 10 Juin 2019

Programmation Système
S6 SMI

Exercice I : Cours

1. Pour quelle raison un tube de communication dispose de deux descripteurs ?
2. Quelles sont les différences entre un fichier et un tube nommé ?
3. Donner la syntaxe de la fonction permettant de redéfinir le comportement d'un signal en expliquant ses paramètres dans le cas général
4. Quelle est la signification de la valeur retournée par la fonction *shmget() ?* par quel moyen un autre processus puisse y accéder ?

Exercice II : Communication par Signaux

On considère deux processus P1 et P2 qui s'exécutent en parallèle. On suppose que P1 est identifié par *pid1* et P2 par *pid2 et* que le processus P1 dispose de l'identificateur du processus P2.

1. Ecrire le code C du processus P1 lui permettant d'envoyer un ensemble de signaux passés en paramètre au processus P2 (le processus P2 s'exécute en boucle).
2. Ecrire le code C du processus P2 lui permettant d'ignorer les deux signaux (SIGINT, SIGSTP) et de compter le nombre de signaux SIGCONT reçus

Exercice III : Mémoire Partagée et Sémaphores

Deux processus P1 et P2 partagent une PILE d'entiers fonctionnant suivant le principe LIFO(Last in First out). Le processus P1 permet d'ajouter des éléments dans la pile et le processus P2 permet de retirer des éléments de la pile. La pile est représentée par un tableau de taille N et le **Sommet** de la pile représentant l'indice du **dernier** élément. La pile est structurée comme suite :

```
typedef struct Pile{
        int tab[N];
        int sommet;
      } Pile;
```

1. Donner la portion de code permettant de créer et d'attacher la pile partagée par les deux processus.

2. On suppose que les deux processus P1 et P2 s 'exécutent en parallèle et qu'on désire gérer la synchronisation entre P1 et P2 comme suite :

- Si la pile est vide P2 ne peut pas retirer d'élément de la pile
- Si la pile est pleine P1 ne peut pas ajouter d'élément dans la pile
- P1 et P2 ne peuvent pas accéder en même temps à la pile

Ecrire en C le code de P1 et P2 permettant de gérer la synchronisation en utilisant les sémaphores.

On suppose qu'on dispose du code des fonctions sur les sémaphores suivantes :

int Creat_Sem(int cle) : fonction qui permet la création d'un sémaphore
void Init_Sem(int id, int valeur) : procédure qui permet d'initialiser la valeur d'un sémaphore
void P(int id) : opération P sur un sémaphore
void V(int id) : opération V sur un sémaphore