TD Programmation Système

Série 6 : Opérations sur les Sémaphores
Correction

Le module permettant la gestion des sémaphores composé des deux fichiers (semaphore.h, semaphore.c) permettant la création, l'initialisation et les opérations sur les sémaphores (P et V). Ce module est utilisé pour implémenter les processus des exercices 1 et 2.

semaphore.h

```
#include<stdio.h>
#include<stdlib.h>
#include <sys/ipc.h>
#include <sys/sem.h>
#include <sys/types.h>

  union semun {
/* Value for SETVAL */
int val;

/* Buffer for IPC_STAT, IPC_SET */
struct semid_ds *buf;

/* Array for GETALL, SETALL */
unsigned short  *array;

/* Buffer for IPC_INFO */
struct seminfo  *__buf;
           };



int CREAT_SEM(int key);
void INIT_SEM(int ids,int valeur);
void P(int ids);
void V(int ids);
```

semaphore.c

```
#include "semaphore.h"

int CREAT_SEM(int key)
{int ids;

 ids=semget(key,1,IPC_CREAT|0777);
     return ids;
}

void INIT_SEM(int ids,int valeur)
{
  union semun ctl;
  ctl.val=valeur;
  semctl(ids,0,SETVAL,ctl);

}

void P(int ids)
{
   struct sembuf arg;
   arg.sem_num=0;
   arg.sem_op=-1;
   arg.sem_flg=SEM_UNDO;
   semop(ids,&arg,1);

}

void V(int ids){

   struct sembuf arg;
   arg.sem_num=0;
   arg.sem_op=1;
   arg.sem_flg=SEM_UNDO;
   semop(ids,&arg,1);
     }
```

Exercice 1 :

Un seul sémaphore S est utilisé pour gérer la synchronisation entre le processus P1 et le processus P2. Il est crée par les deux processus en utilisant la fonction CREAT_SEM(). Il est initialisé par le processus P2 qui l'utilise en premier par la fonction `INIT_SEM()` .

P1.C

```
#include"semaphore.h"
#define key1 200

main(){

            int S,n=0;
            S=CREAT_SEM(key1);

            do{ n=n+2;
                 printf("%d\t", n);
                   }while(n%5!=0);
            V(S);
}
```

P2.C

```
#include"semaphore.h"
#define key1 200

main(){
      int S, n=0;

      S=CREAT_SEM(key1);
      INIT_SEM(S,0);
      P(S);
      do{ n=n+3;
             printf("%d\t", n);
            }while(n%5!=0);
}
```

Exercice 2 :

Deux sémaphores S1 et S2 sont utilisé pour gérer la synchronisation entre le processus P1 et le processus P2 :

- S1 : pour gérer la synchronisation entre P1 et P2
- S2 : pour gérer la synchronisation entre P2 et P1

Il sont crées par les deux processus en utilisant la fonction CREAT_SEM().

- S1 est initialisé par le processus P1 qui l'utilise en premier par la fonction `INIT_SEM()`
- S2 est initialisé par le processus P2 qui l'utilise en premier par la fonction `INIT_SEM()`

P1.C

```
#include <stdio.h>
#include <stdlib.h>
#include <sys/ipc.h>
#include <sys/shm.h>
#include <signal.h>
#include"semaphore.h"
#define keyS1 200
#define keyS2 300
#define keyM 120


main(){
       int idm, *N, S1, S2;


idm=shmget(keyM,sizeof(int),
           IPC_CREAT|0666);

    if(idm==-1){
               perror("shmget");
               exit(0);
               }

     N=(int*)shmat(idm,0,0);

     S1=CREAT_SEM(keyS1);
     S2=CREAT_SEM(keyS2);
     INIT_SEM(S1,0);

    *N=0;
     while (*N<= 100)
    {
        do {
        *N=*N+2;
           printf(" %d",*N);
           } while (*N%5!=0);

       V(S2);
       P(S1);
       printf("\n");
      }shmdt(N);
}
```

P2.C

```
#include <stdio.h>
#include <stdlib.h>
#include <sys/ipc.h>
#include <sys/shm.h>
#include <signal.h>
#include"semaphore.h"
#define keyS1 200
#define keyS2 300
#define keyM 120


main(){
       int idm, *N, S1, S2;


idm=shmget(keyM,sizeof(int),
        IPC_CREAT|0666);

    if(idm==-1){
               perror("shmget");
               exit(0);
               }

     N=(int*)shmat(idm,0,0);

     S1=CREAT_SEM(keyS1);
     S2=CREAT_SEM(keyS2);
     INIT_SEM(S2,0);

    P(S2);
     while (*N <= 100)
    {
        do {
        *N=*N+3;
           printf(" %d",*N);
           } while (*N%5!=0);

       V(S1);
       P(S2);
       printf("\n");
      }

shmdt(N);
shmctl(idm,IPC_RMID,0);
}
```