TD Programmation Système

Série 4 Mémoires Partagées : Correction

Exercice 1 : Création Mémoires Partagées

P1.C

```
#include <sys/types.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>
#include <sys/ipc.h>
#include <sys/shm.h>
#define key 100

main(int argc, char *argv[])
{
 int idM;
 float *N;

/* Création de la mémoire partagée de
type float N et son attachement*/

 idM=shmget(key,sizeof(float),
             IPC_CREAT|0666);
 if(idM==-1)
 {
      perror("shmget");
      exit(0);
 }
 N=(float*)shmat(idM,0,0);

/* initialisation de N par la valeur
passée en paramètre */

*N=atof(argv[1]);

  printf("%f\n",*N);

  shmdt(N);
}
```

P2.C

```
#include <sys/types.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>
#include <sys/ipc.h>
#include <sys/shm.h>
#define key 100

main(int argc, char *argv[])
{
 int idM;
 float *N;

/*Accès à la mémoire partagée de type
float N par la clé 100*/
idM=shmget(key,sizeof(float),IPC_E
XCL);
 if(idM==-1)
 {
 perror("shmget");
 exit(0);
 }

 N=(float*)shmat(idM,0,0);

/* calcul du carrée de la valeur
récupérer */

 *N=(*N)*(*N);
printf("Nombre = %f\n",*N);

shmdt(N);
/* suppression de la mémoire
partagée*/
shmctl(idM,IPC_RMID,0);

}
```

Exercice 2 :

Le processus P1 crée la mémoire partagée de type entier l'utilise pour afficher les multiple de 2 jusqu'au premier multiple de 5 le processus P2 accède à la mémoire partagée récupère la dernière valeur l'utilise pour afficher les 5 premières valeurs par ajout de 3.

P1.C

```
#include<stdio.h>
#include<stdlib.h>
#include<sys/types.h>
#include<sys/ipc.h>
#include<sys/shm.h>
#include<unistd.h>
#define key 100

main()
{
int idM,*n;

idM=shmget(key,sizeof(int),IPC_CRE
AT|0666);
if(idM==-1)
{
 perror("shmget");
exit(0);
}
n=(int*)shmat(idM,0,0);

*n=0;

/* affichage des 5 premiére valeurs
multiple de 2*/
do
      {
      *n=*n+2;
      printf(" %d ",*n);
      }while (*n%5 !=0);
printf("\n");

shmdt(n);
}
```

P2.C

```
#include<stdio.h>
#include<stdlib.h>
#include<sys/types.h>
#include<sys/ipc.h>
#include<sys/shm.h>
#include<unistd.h>
#define key 100

main()
{
int idM,*n;
idM=shmget(key,sizeof(int),IPC_EXC
L|0666);
if(idM==-1)
{
 perror("shmget");
exit(0);
}
n=(int*)shmat(idM,0,0);

/* accès à la dernière valeur de n
partagée avec P1 et affichage des
5 premières valeurs par ajout de 3 */

do
{
      *n=*n+3;
      printf(" %d ",*n);
}while (*n%5 !=0);
printf("\n");

shmdt(n);
shmctl(idM,IPC_RMID,0);
}
```

Exercice 3 : Synchronisation

Les processus P1 et P2 crée la mémoire partagée pour stocker et accéder aux valeurs de leurs pid et de la valeur n.

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

Au début le processus P2 et en pause le processus P1 affiche les valeurs de n jusqu'au premier multiple de 5 envoi un message au processus 2 pour le réveiller et se met en pause. Le processus P2 accède à la dernières valeur de n affiche les valeurs obtenues en ajoutant 3 jusqu'au premier multiple de 5 envoi un message au processus 1 pour le réveiller et se met en pause. Les processus P1 et P2 répètent le même traitement jusqu'a la valeur d'arrêt où n=100.

Les processus P1 et P2 redéfinissent le signal SIGUSR1 qui sera utiliser pour la communication. Le nouveau comportement et sans code car le processus est utiliser seulement pour le réveil.

```
void handler(){}
.......

signal(SIGUSR1,handler);
```

P1 réveille P2 par l'envoi du signal : SIGUSR1

```
kill(mp->pid2,SIGUSR1);
pause();
```

P2 réveille P1 par l'envoi du signal : SIGUSR1

```
kill(mp->pid1,SIGUSR1);
pause();
```

P1.C

```
#include <stdio.h>
#include <stdlib.h>
 #include <sys/ipc.h>
#include <sys/shm.h>
#include <signal.h>
#define key 130

typedef struct MP{
  int pid1;
  int pid2;
  int n;
}MP;

void handler(){}

main(){
 int idm;
 MP *mp;

 signal(SIGUSR1,handler);

idm=shmget(key,sizeof(MP),IPC_CREA
T|0666);
    if(idm==-1){perror("shmget");
                exit(0);}

mp=(MP*)shmat(idm,0,0);
mp->pid1=getpid();
mp->n=0;

while (mp->n <= 100)
    {
         do {
         mp->n=mp->n+2;
            printf(" %d",mp->n);
             } while (mp->n%5!=0);

        kill(mp->pid2,SIGUSR1);
        pause();
        printf("\n");
      }shmdt(mp);
}
```

P2.C

```
#include <stdio.h>
#include <stdlib.h>
 #include <sys/ipc.h>
#include <sys/shm.h>
#include <signal.h>
#define key 130

typedef struct MP{
  int pid1;
  int pid2;
  int n;
}MP;

void handler(){}

main(){
int idm;
MP *mp;

signal(SIGUSR1,handler);

idm=shmget(key,sizeof(MP),IPC_CRE
AT|0666);
  if(idm==-1){perror("shmget");
                exit(0);}

mp=(MP*)shmat(idm,0,0);
mp->pid2=getpid();


while (mp->n < 100)
    {
     do {
        mp->n=mp->n+3;
           printf(" %d",mp->n);
           } while (mp->n%5!=0);

       kill(mp->pid1,SIGUSR1);
       pause();
       printf("\n");
    }

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