TD Programmation Système Série 8

Correction : THREAD

Exercice 1 : Thread Identificateur

```
#include<stdio.h>
#include<stdlib.h>
#include <sys/types.h>
#include <unistd.h>
#include <pthread.h>

#define MAX 100


void *funct1(void * arg)
{
        long num= (long) arg;

printf("Hello je suis le thread  numéro : %ld identificateur :  %ld \n", num, pthread_self());

}



int main(int argc, char *argv[]){

        long i;
        int nb;

        pthread_t t[MAX];

        nb=atoi(argv[1]);

        for(i=0;i<nb;i++)
                pthread_create(&t[i], NULL, funct1, (void *) i);

        for(i=0;i<nb;i++)
                pthread_join(t[i], NULL);


        return 0;


}
```

Exercice 2 : Synchronisation avec les Threads

1.

```
#include<stdio.h>
#include<stdlib.h>
#include <sys/types.h>
#include <unistd.h>
#include <pthread.h>

#define MAXCLIENT 5
#define INIT_STOCK 20

static int stock= INIT_STOCK; //stock Initial

void *fstock(void * arg)
{
      while(1)
      {
       if (stock <= 0)
           {stock= INIT_STOCK;
           printf(" Remplissage du stock  = %d\n", stock); }
       }
pthread_exit(NULL);
}

void *fclient(void * arg)
{ long num= (long) arg;
 int val;
while(1)
     {      val = rand() % INIT_STOCK;
            sleep(1);
            stock=stock - val;
       printf("Je suis le client  numéro : %ld, valeur commandé : %d, stock = %d \n", num,val,stock);
       }
pthread_exit(NULL);
}

int main(int argc, char *argv[]){
           long i;
           int nb, retour;
           pthread_t Tclient[MAXCLIENT], Tsock;

           /* Thread Stock */
           retour = pthread_create(&Tsock, NULL,fstock , (void *) i);

           if(retour==0)
           {   /* Thread Client */
           for(i=0;i<MAXCLIENT;i++)
                      pthread_create(&Tclient[i], NULL, fclient, (void *) i);

           /* Attente de fin de Threads Client */
           for(i=0;i<MAXCLIENT;i++)
                      pthread_join(Tclient[i], NULL);

       /* Attente de fin de Thread Tstock */
       pthread_join(Tsock, NULL);
            }
           return 0;
}
```

2. Pour gérer le problème de l'exclusion mutuelle, le programme principale (main()) reste le même, ce qui change se sont les fonctions associées aux threads.

```
#include<stdio.h>
#include<stdlib.h>
#include <sys/types.h>
#include <unistd.h>
#include <pthread.h>

#define MAXCLIENT 5
#define INIT_STOCK 20

pthread_mutex_t mutex = PTHREAD_MUTEX_INITIALIZER; //Initialiser mutex

static int stock= INIT_STOCK;


void *fstock(void * arg)
{
while(1)
          {
          pthread_mutex_lock(&mutex) ;//Bloqué l'accès Exclusion Mutuelle
           if (stock <= 0)
          {
          stock= INIT_STOCK;
          printf(" Remplissage du stock  = %d\n", stock);
          }
      pthread_mutex_unlock(&mutex);
      }
pthread_exit(NULL);
}

void *fclient(void * arg)
{
long num= (long) arg;
int val;

while(1)
    {
       val = rand() % INIT_STOCK;
       sleep(1);

       pthread_mutex_lock(&mutex); //Bloqué l'accès Exclusion Mutuelle
       stock=stock - val;
       printf("Je suis le client  numéro : %ld, valeur commandé : %d, stock = %d \n", num,val,stock);
       pthread_mutex_unlock(&mutex); // Débloqué l'accès Exclusion Mutuelle
      }

pthread_exit(NULL);
}
```

3. Pour gérer le problème de l'exclusion mutuelle et de la synchronisation, le programme principale (main()) reste le même, ce qui change se sont les fonctions associées aux threads.

```
#include<stdio.h>
#include<stdlib.h>
#include <sys/types.h>
#include <unistd.h>
#include <pthread.h>

#define MAXCLIENT 5
#define INIT_STOCK 20

// Initialisation de mutex et condition
pthread_mutex_t mutex = PTHREAD_MUTEX_INITIALIZER;
pthread_cond_t cond   = PTHREAD_COND_INITIALIZER;

static int stock= INIT_STOCK;

void *fstock(void * arg)
{
while(1)
          {
          pthread_mutex_lock(&mutex) ;//Bloqué l'accès Exclusion Mutuelle
      if (stock <= 0)
          {
          stock= INIT_STOCK;
          printf(" Remplissage du stock  = %d\n", stock);
          pthread_cond_signal(&cond); // déverouillé accès sous condition
          }
       pthread_mutex_unlock(&mutex);
       }
pthread_exit(NULL);
}

void *fclient(void * arg)
{
long num= (long) arg;
int val;

while(1)
{           val = rand() % INIT_STOCK;
            sleep(1);
            pthread_mutex_lock(&mutex); //Bloqué l'accès Exclusion Mutuelle
        stock=stock - val;
        if ((stock <=0) || (val <= stock))
        {
          pthread_cond_wait(&cond,&mutex); // accès verouillé en attente
          stock=stock - val;
        }
      printf("Je suis le client  numéro : %ld, valeur commandé : %d, stock = %d \n", num,val,stock);
      pthread_mutex_unlock(&mutex); // Débloqué l'accès Exclusion Mutuelle
       }

pthread_exit(NULL);
}
```