GL4-RT5

Systèmes Temps-Réel

Chapitre 4 :

POSIX
Exercice

Olfa Mosbahi

olfamosbahi@gmail.com

Exemple

Un pont supporte une charge maximale de 15 tonnes. Ce pont est traversé par des camions dont le poids est de 15 tonnes ainsi que par des voitures dont le poids est de 5 tonnes. On vous demande de gérer l'accès au pont de sorte que la charge maximale du pont soit respectée.

Questions :

1. Donnez un programme comportant un moniteur (une fonction d'acquisition et une fonction de libération du pont) qui simule les règles de partage du pont ci-dessus. Votre programme modélisera les camions et voitures sous la forme de threads.

2. Quand un véhicule quitte le pont, on souhaite donner la priorité aux camions : lorsqu'une voiture et un camion sont bloqués en attente d'obtenir l'accès au pont, le camion doit être réveillé en premier, sous réserve, bien sûr, que la capacité maximale du pont soit respectée.

Correction de l'Exemple 1 (Q1)

```
#include <time.h>
#include <pthread.h>
#include <semaphore.h>
#include <stdlib.h>


#define NB_CAMIONS      5
#define NB_VOITURES     5
#define NB_VEHICULES  (NB_CAMIONS+NB_VOITURES)



void attendre(double max);
int tirage_aleatoire(double max);


/* Code des threads */
void* voiture(void* arg);
void* camion(void* arg);


  sem_t sem_compteur;

  pthread_mutex_t camion_sc;
```

```
id acceder_au_pont(int tonnes)


      if (tonnes==15)
              {
              pthread_mutex_lock(&camion_sc);
              while(tonnes>0)
                      {
                      sem_wait(&sem_compteur);
                      tonnes=tonnes-5;
                      };
              pthread_mutex_unlock(&camion_sc);
              }
      else  sem_wait(&sem_compteur);



id quitter_le_pont(int tonnes)

      while(tonnes>0)
              {
              sem_post(&sem_compteur);
              tonnes=tonnes-5;
              };
}
```

Correction de l'Exemple 1 (Q1)

En Posix :

- Un sémaphore initialisé a 1, est de type *pthread_mutex_t*

- **Un sémaphore initialisé a 3, doit être de type *sem_t***

- ***sem_compteur : sémaphore entre camions***
- ***camion_Sc : sémaphore entre camions et voitures***

```
int tirage_aleatoire(double max)
{
        int j=(int) (max*rand()/(RAND_MAX+1.0));
        if(j<1)
                j=1;
        return j;
}



void attendre(double max)
{
        struct timespec delai;

        delai.tv_sec=tirage_aleatoire(max);
        delai.tv_nsec=0;
        nanosleep(&delai,NULL);
}
```

```
void* voiture(void* arg)
{
        int pid=*((int*)arg);

        attendre(5.0);
        acceder_au_pont(5);
        printf("Voiture %d : je traverse le pont\n",pid);
        attendre(5.0);
        printf("Voiture %d : je quitte le pont\n",pid);
        quitter_le_pont(5);

        pthread_exit(NULL);
}

void* camion(void* arg)
{
        int pid=*((int*)arg);

        attendre(5.0);
        acceder_au_pont(15);
        printf("Camion %d : je traverse le pont\n",pid);
        attendre(5.0);
        printf("Camion %d : je quitte le pont\n",pid);
        quitter_le_pont(15);

        pthread_exit(NULL);
}
```

Correction de l'Exemple 1 (Q1)

```
int main(int argc, char* argv[])
{
        int i;
        pthread_t id;


        sem_init(&sem_compteur,0,3);

        for(i=0; i<NB_VEHICULES;i++)
                        {
                        int* j=(int*)malloc(sizeof(int));
                        *j=i;
                        if (i<NB_CAMIONS)
                                pthread_create(&id,NULL,camion,j);
                        else    pthread_create(&id,NULL,voiture,j);
                        }


        pthread_exit(NULL);
};
```

Correction de l'Exemple 1 (Q2)

```
#include <time.h>
#include <pthread.h>
#include <semaphore.h>
#include <stdlib.h>


#define NB_CAMIONS      5
#define NB_VOITURES     5
#define NB_VEHICULES  (NB_CAMIONS+NB_VOITURES)


#define ATTENDRE 1
#define RIEN 2
#define TRAVERSER 3


void attendre(double max);
int tirage_aleatoire(double max);


/* Code des threads */
void* voiture(void* arg);
void* camion(void* arg);


/* Section critique pour proteger les variables
        d'etat                                  */
pthread_mutex_t sc;
```

```
/* Semaphore prive pour bloquer les vehicules */
/* Un semaphore par vehicule   */
sem_t sempriv[NB_VEHICULES];


/* Etat des vehicules */
int etat[NB_VEHICULES];


/* Nombre de camions actuellement bloques */
int nb_camions_bloques=0;


/* Etat du pont : poids actuellement sur le pont */
int seuil=0;
```

Correction de l'Exemple 1 (Q2)

```
void acceder_au_pont(int tonnes, int id)
{
        pthread_mutex_lock(&sc);

        if(seuil+tonnes <= 15)
                {
                seuil=seuil+tonnes;
                etat[id]=TRAVERSER;
                sem_post(&sempriv[id]);
                }
        else    {
                etat[id]=ATTENDRE;
                if(tonnes==15)
                        nb_camions_bloques++;
                }

        pthread_mutex_unlock(&sc);
        sem_wait(&sempriv[id]);
}
```

```
void quitter_le_pont(int tonnes, int pid)
{
        int i;

        pthread_mutex_lock(&sc);

        etat[pid]=RIEN;
        seuil=seuil-tonnes;

        /* On libere les camions d'abord */
        for(i=0; i<NB_CAMIONS;i++)
                if ( (etat[i]==ATTENDRE) && (seuil==0) )
                        {
                        sem_post(&sempriv[i]);
                        seuil=15;
                        nb_camions_bloques--;
                        }

        /* On s'occupe des voitures maintenant */
        /* Si plus d'un camion est bloque, on ne
                libere pas de voiture */
        for(i=NB_CAMIONS; i<NB_VEHICULES;i++)
                if ( (seuil<15) && (nb_camions_bloques==0)
                        && (etat[i]==ATTENDRE) )
                        {
                        seuil=seuil+5;
                        sem_post(&sempriv[i]);
                        }

        pthread_mutex_unlock(&sc);
}
```

Correction de l'Exemple 1 (Q2)

```
int main(int argc, char* argv[])
{
        int i;
        pthread_t id;


        for(i=0; i<NB_VEHICULES;i++)
                        {
                        etat[i]=RIEN;
                        sem_init(&sempriv[i],0,0);
                        }

        pthread_mutex_init(&sc,0);

        for(i=0; i<NB_VEHICULES;i++)
                        {
                        int* j=(int*)malloc(sizeof(int));
                        *j=i;
                        if (i<NB_CAMIONS)
                                pthread_create(&id,NULL,camion,j);
                        else    pthread_create(&id,NULL,voiture,j);
                        }


        pthread_exit(NULL);
};
```

```
void* voiture(void* arg)
{
        int pid=*((int*)arg);

        attendre(5.0);
        acceder_au_pont(5, pid);
        printf("Voiture %d : je traverse le pont\n",pid);
        attendre(5.0);
        printf("Voiture %d : je quitte le pont\n",pid);
        quitter_le_pont(5, pid);

        pthread_exit(NULL);
}

void* camion(void* arg)
{
        int pid=*((int*)arg);

        attendre(5.0);
        acceder_au_pont(15, pid);
        printf("Camion %d : je traverse le pont\n",pid);
        attendre(5.0);
        printf("Camion %d : je quitte le pont\n",pid);
        quitter_le_pont(15, pid);

        pthread_exit(NULL);
}
```