Esercizio E5.3[1]

Impostazione

1. Quali processi?
   a. Tifosi Italiani
   b. Tifosi Stranieri

2. Quale struttura per i processi ?

Sia `S` la struttura dati che contiene i dati relativi allo stadio e `porta` l'identificatore del cancello scelto per l'ingresso allo stadio:

Tifoso Italiano:

```
entra_corridoio_IN(&S, porta, ITA);
< attraversa corridoio per entrare... >
esci_corridoio_IN(&S, porta, ITA);
< vede la partita>
entra_corridoio_OUT(&S, porta, ITA);
< attraversa corridoio per uscire... >
esci_corridoio_OUT(&S, porta, ITA);
```

Tifoso Straniero

```
entra_corridoio_IN(&S, porta, STRA);
< attraversa corridoio per entrare... >
esci_corridoio_IN(&S, porta, STRA);
< vede la partita>
entra_corridoio_OUT(&S, porta, STRA);
< attraversa corridoio per uscire... >
esci_corridoio_OUT(&S, porta, STRA);
```

3. Definizione del monitor **stadio**:

Dati:

```
typedef struct{
 pthread_mutex_t lock; /* mutua esclusione nell'accesso al monitor */
/*INGRESSO: 1 coda x ogni cancello e x ogni tipo di tifoso:*/
 pthread_cond_t codaIN[Nporte][Ntipitifosi];
 int sospIN[Nporte][Ntipitifosi]; /*numero tifosi sospesi per ogni coda */
/*USCITA: 1 coda x ogni cancello e x ogni tipo di tifoso*/
 pthread_cond_t codaOUT[Nporte][Ntipitifosi]; ];
 int sospOUT[Nporte][Ntipitifosi]; /*numero tifosi sospesi per ogni coda */
 int DENTRO; /* numero tifosi nello stadio (<=MAXC) */
 int DENTRO_CORR_IN[Nporte][Ntipitifosi]; /*tifosi nei corr. in ingresso */
 int DENTRO_CORR_OUT[Nporte][Ntipitifosi]; /*tifosi nei corr. in uscita */
} stadio;
```

Operazioni :

`entra_corridoio_IN(stadio *s, int porta, int t)`
operazione eseguita da ogni thread di tipo t per l'accesso al corridoio (porta) in ingresso.

`esci_corridoio_IN(stadio *s, int porta, int t)`
operazione eseguita da ogni thread di tipo t per uscire dal corridoio (porta) in ingresso.

[1] Per semplicità, il testo dell'esercizio è stato modificato rispetto alla versione pubblicata sul libro, eliminando la distinzione tra tifosi bambini e adulti.

`entra_corridoio_OUT(stadio *s, int porta, int t)`
operazione eseguita da ogni thread di tipo t per l'accesso al corridoio (porta) in uscita.

`esci_corridoio_OUT(stadio *s, int porta, int t)`
operazione eseguita da ogni thread di tipo t per uscire dal corridoio (porta) in uscita.

Soluzione:

```
#include <stdio.h>
#include <stdlib.h>
#include <pthread.h>
#define Nporte 2
#define Ntipitifosi 2
#define ITA 0
#define STRA 1
#define MAXC 100
#define MAXT 50

typedef struct{
      pthread_mutex_t lock;
      pthread_cond_t codaIN[Nporte][Ntipitifosi];
      int sospIN[Nporte][Ntipitifosi];
      pthread_cond_t codaOUT[Nporte][Ntipitifosi];
      int sospOUT[Nporte][Ntipitifosi];
      int DENTRO; /* numero tifosi nello stadio (<=MAXC) */
      int DENTRO_CORR_IN[Nporte][Ntipitifosi];
      int DENTRO_CORR_OUT[Nporte][Ntipitifosi];
} stadio;

stadio S;
int other(int n)
{ return (!n);}

void entra_corridoio_IN(stadio *s, int porta, int tipo) {
      pthread_mutex_lock (&s->lock);
      /* controlla le condizioni di accesso:*/
      while ( (s->DENTRO==MAXC) /* se lo stadio e` pieno */ ||
              (s->DENTRO_CORR_OUT[porta][other(tipo)]>0) /* se ci sono tifosi di
                  altro tipo in dir opposta*/ ||
              (s->sospOUT[porta][other(tipo)]>0) /*se c'e` qualcuno (del tipo
                  opposto) in attesa di uscire */ ||
              ((tipo==ITA) && s->sospIN[other(porta)][STRA]>0)) /* ci sono
                  stranieri in attesa di entrare dall'altra porta*/
            { s->sospIN[porta][tipo]++;
              pthread_cond_wait (&s->codaIN[porta][tipo], &s->lock);
              s->sospIN[porta][tipo]--;
      }
      /* ingresso nel corridoio: */
      s->DENTRO++;
      s->DENTRO_CORR_IN[porta][tipo]++;
      pthread_cond_signal(&s->codaIN[porta][tipo]); /* segnalo il successivo */
      pthread_mutex_unlock (&s->lock);
}
```

```
void entra_corridoio_OUT(stadio *s, int porta, int tipo) {
      pthread_mutex_lock (&s->lock);
      /* controlla le condizioni di accesso:*/
      while ( s->DENTRO_CORR_IN[porta][other(tipo)]>0)
            /* ci sono tifosi di altro tipo in dir opposta*/
      {     s->sospOUT[porta][tipo]++;
            pthread_cond_wait (&s->codaOUT[porta][tipo], &s->lock);
            s->sospOUT[porta][tipo]--;
      }
      /* ingresso nel corridoio: */
      s->DENTRO--;
      s->DENTRO_CORR_OUT[porta][tipo]++;
      pthread_cond_signal(&s->codaOUT[porta][tipo]); /* segnalo il successivo */
      if ((tipo==STRA)&& (DENTRO==MAXC-1))
      {     pthread_cond_signal(&s->codaIN[porta][STRA]);
            /* segnalo processi stranieri in ingresso sulla stessa porta */
            pthread_cond_signal(&s->codaIN[other(porta)][STRA]);
            /* segnalo processi stranieri in ingresso sull'altra porta */
            pthread_cond_signal(&s->codaIN[other(porta)][ITA]);
            /* segnalo processi italiani in ingresso sull'altra porta */
      }
      else if ((tipo==ITA)&& (DENTRO==MAXC-1))
      {     pthread_cond_signal(&s->codaIN[other(porta)][STRA]);
            /* segnalo processi stranieri in ingresso sull'altra porta */
            pthread_cond_signal(&s->codaIN[porta][ITA]);
            /* segnalo processi italiani in ingresso sulla stessa porta */
            pthread_cond_signal(&s->codaIN[other(porta)][ITA]);
            /* segnalo processi italiani in ingresso sull'altra porta */
      }
      pthread_mutex_unlock (&s->lock);
}

void esci_corridoio_IN(stadio *s, int porta, int tipo) {
      pthread_mutex_lock (&s->lock);
            /* uscita dal corridoio: */
      s->DENTRO_CORR_IN[porta][tipo]--;
      pthread_cond_signal(&s->codaOUT[porta][other(tipo)]);
            /* sveglio processi di tipo opposto in dir opposta*/
      pthread_mutex_unlock (&s->lock);
}

void esci_corridoio_OUT(stadio *s, int porta, int tipo) {
      pthread_mutex_lock (&s->lock);
      s->DENTRO_CORR_OUT[porta][tipo]--;
      pthread_cond_signal(&s->codaIN[porta][other(tipo)]);
      pthread_mutex_unlock (&s->lock);
}

void *thread_Italiano(void * arg) { /*thread italiano*/
      int porta;
      porta=atoi((char *)arg);
      sleep(1);
      entra_corridoio_IN(&S,  porta, ITA);
      sleep(1); /* attraversamento corridoio... */
      esci_corridoio_IN(&S,  porta, ITA);
```

```
      sleep(1); /* vede la partita... */
      entra_corridoio_OUT(&S,  porta, ITA);
      sleep(1); /* attraversamento corridoio... */
      esci_corridoio_OUT(&S,  porta, ITA);
}

void *thread_Straniero(void * arg) {  /*thread straniero*/
      int porta;
      porta=atoi((char *)arg);
      sleep(1);
      entra_corridoio_IN(&S,  porta, STRA);
        sleep(1); /* attraversamento corridoio... */
      esci_corridoio_IN(&S,  porta, STRA);
        sleep(1); /* vede la partita... */
      entra_corridoio_OUT(&S,  porta, STRA);
        sleep(1); /* attraversamento corridoio... */
      esci_corridoio_OUT(&S,  porta, STRA);
}

void init (stadio *p) { /*inizializzazione del monitor */
      int i, j;
      pthread_mutex_init (&p->lock, NULL);
      p->DENTRO=0;
      for (i=0; i<Nporte; i++)
         for(j=0; j<Ntipitifosi; j++)
         {  pthread_cond_init (&p->codaIN[i][j], NULL);
            pthread_cond_init (&p->codaOUT[i][j], NULL);
            p->sospIN[i][j]=0;
            p->sospOUT[i][j]=0;
            p->DENTRO_CORR_IN[i][j]=0;
            p->DENTRO_CORR_OUT[i][j]=0;
         }
}

/* programma di test: */

main (){
  pthread_t th_I[MAXT][Nporte], th_S[MAXT][Nporte];
  int NI[Nporte], NS[Nporte], i;
  void *retval;
  init (&S);
  /* Creazione threads: */
  printf("\nquanti ITALIANI sulla porta 0? ");
  scanf("%d", &NI[0]);
  printf("\nquanti ITALIANI sulla porta 1? ");
  scanf("%d", &NI[1]);
  printf("\nquanti STRANIERI sulla porta 0? ");
  scanf("%d", &NS[0]);
  printf("\nquanti STRANIERI sulla porta 1? ");
  scanf("%d", &NS[1]);
  /*Creazione italiani*/
  for (i=0; i<NI[0]; i++)
          pthread_create (&th_I[i][0], NULL, thread_Italiano, "0");
  for (i=0; i<NI[1]; i++)
```

```
            pthread_create (&th_I[i][1], NULL, thread_Italiano, "1");
    /*Creazione stranieri*/
    for (i=0; i<NI[0]; i++)
            pthread_create (&th_S[i][0], NULL, thread_Straniero, "0");
    for (i=0; i<NI[1]; i++)
            pthread_create (&th_S[i][1], NULL, thread_Straniero, "1");
    /* Attesa teminazione threads creati: */
    for (i=0; i<NI[0]; i++)
            pthread_join(th_I[i][0], &retval);
    for (i=0; i<NI[1]; i++)
            pthread_join(th_I[i][1], &retval);
    for (i=0; i<NS[0]; i++)
            pthread_join(th_S[i][0], &retval);
    for (i=0; i<NS[1]; i++)
            pthread_join(th_S[i][1], &retval);
     return 0;
}
```