Sincronizzazione tra thread

Leonardo Bizzoni

November 5, 2023

1 Lock Mutex

```
#include <pthread.h>
#include <sched.h>
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>

pthread_mutex_t mutex;

int count = 0;

void *incrementer();
void *printer();

int main(void) {
  pthread_t t1, t2;

  if (pthread_create(&t1, NULL, &incrementer, NULL) != 0) {
    perror("[incrementer thread]");
  }
  if (pthread_create(&t2, NULL, &printer, NULL) != 0) {
    perror("[printer thread]");
  }

  if(pthread_join(t1, NULL) != 0) {
     perror("[incrementer thread]");
  }
  if(pthread_join(t2, NULL) != 0) {
```

```
     perror("[printer thread]");
  }
}

void *incrementer() {
  static int local_count = 0;

  /* Non avendo ancora il lock su ‘count‘ */
  /* non posso accedervi nemmeno in letture */
  while (local_count < 100) {
    if (pthread_mutex_lock(&mutex)) {
       continue;
    }
    /* Inizio sezione critica */

    local_count = count;
    count++;
    if (local_count + 1 != count) {
      fprintf(stderr, "Violazione della sezione critica!\n");
      exit(EXIT_FAILURE);
    }
    pthread_mutex_unlock(&mutex);

    /* Fine sezione critica */
    sched_yield();
  }
  return NULL;
}

void *printer() {
  static int local_count = 1;

  while (local_count > 0) {
    if (pthread_mutex_lock(&mutex)) {
       continue;
    }

    local_count = count;
    printf("Il valore di 'count' è: %d\n", local_count);
    count--;
```

```
    pthread_mutex_unlock(&mutex);
    sched_yield();
  }
  return NULL;
}
```

2 Semafori

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

#define GIOCATORI 10
#define POSTI 5

const u_int16_t tempo_in_partita[GIOCATORI] = {
  1000, 2000, 1500, 5000, 6000, 750, 2300, 2222, 3000, 450
};

sem_t semaphore;

void *gioca(void *);

int main(void) {
  /* 0 indica che il semaforo è tra thread mentre un valore */
  /* diverso da 0 indica che è un semaforo tra processi. */
  /* L'ultimo argomento è il numero di thread/processi consentiti */
  /* contemporaneamente. */
  sem_init(&semaphore, 0, POSTI);
  pthread_t threads[GIOCATORI];

  for (u_int64_t i = 0; i < GIOCATORI; i++) {
    if (pthread_create(&threads[i], NULL, &gioca, (void *)i) != 0) {
      perror("[incrementer thread]");
    }
```

```
  }

  for (int i = 0; i < GIOCATORI; i++) {
    if (pthread_join(threads[i], NULL) != 0) {
      perror("[printer thread]");
    }
  }
}

void *gioca(void *num) {
  sem_wait(&semaphore);

  u_int64_t num_giocatore = (u_int64_t)num;
  printf("Entra in campo giocatore: %lu\n", num_giocatore);
  usleep(tempo_in_partita[num_giocatore]);
  printf("Esce dal campo giocatore: %lu\n", num_giocatore);

  sem_post(&semaphore);

  return NULL;
}
```