Lezione 10 - 17/5/21 (Mutex)

Mutex

```
int pthread_mutex_init(pthread_mutex_t *restrict mutex, const pthread_mutexattr_t *restrict attr)
int pthread_mutex_destroy(pthread_mutex_t *mutex)
```

```
#include <pthread.h> //createMutex.c
pthread_mutex_t lock;
int main(void){
  pthread_mutex_init(&lock, NULL); // Create mutex with default attrs
  pthread_mutex_destroy(&lock); // Destroy mutex (it can be re-init)
}
```

Lock & Unlock

```
int pthread_mutex_lock(pthread_mutex_t *mutex)
int pthread_mutex_unlock(pthread_mutex_t *mutex)
```

```
#include <pthread.h>
pthread_mutex_t lock;
void * thread(void *){
  pthread_mutex_lock(&lock); /* Waits for mutex to be unlocked, then
   locks it becoming the owner */
  pthread_mutex_unlock(&lock); //Unlock mutex allowing others to lock it
}
```

Esempio completo

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

pthread_mutex_t lock;
pthread_t tid[2];
int counter = 0;

void* thr1(void* arg){
```

```
    pthread_mutex_lock(&lock);
    counter = 1;
    printf("Thread 1 has started with counter %d\n",counter);
    for (long unsigned i = 0; i < (0x00FF0000); i++);
    counter += 1;
    pthread_mutex_unlock(&lock);
    printf("Thread 1 expects 2 and has: %d\n", counter);
}

void* thr2(void* arg){
    pthread_mutex_lock(&lock);
    counter = 10;
    printf("Thread 2 has started with counter %d\n",counter);
    for (long unsigned i = 0; i < (0xFFF0000); i++);
    counter += 1;
    pthread_mutex_unlock(&lock);
    printf("Thread 2 expects 11 and has: %d\n", counter);
}
int main(void){
    pthread_mutex_init(&lock, NULL);
    pthread_create(&(tid[0]), NULL, thr1,NULL);
    pthread_create(&(tid[1]), NULL, thr2,NULL);
    pthread_join(tid[0], NULL);
    pthread_join(tid[1], NULL);
    pthread_mutex_destroy(&lock);
}
```

Tipi di mutex

- **PTHREAD_MUTEX_NORMAL**: deadlock detection
  - ***Ribloccare quando bloccato*** → deadlock
  - ***Sbloccare quando bloccato da altri*** → undefined
  - ***Sbloccare quando sbloccato*** → undefined
- **PTHREAD_MUTEX_ERRORCHECK**: error checking
  - ***Ribloccare quando bloccato*** → error
  - ***Sbloccare quando bloccato da altri*** → error
  - ***Sbloccare quando sbloccato*** → error
- **PTHREAD_MUTEX_RECURSIVE**: multiple locks
  - ***Ribloccare quando bloccato*** → increase lock count → richiede stesso numero di sbloccaggi
  - ***Sbloccare quando bloccato da altri*** → error

    - ***Sbloccare quando sbloccato*** → error
- **PTHREAD_MUTEX_DEFAULT**:
    - ***Ribloccare quando bloccato*** → undefined
    - ***Sbloccare quando bloccato da altri*** → undefined
    - ***Sbloccare quando sbloccato*** → undefined

Esempio mutex ricorsivo

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

pthread_mutex_t lock;
pthread_t tid[2];
int counter = 0;

void* thr1(void* arg){
    pthread_mutex_lock(&lock);
    pthread_mutex_lock(&lock);
    counter = 1;
    printf("Thread 1 has started with counter %d\n",counter);
    for (long unsigned i = 0; i < (0x00FF0000); i++);
    counter += 1;
    pthread_mutex_unlock(&lock);
    printf("Thread 1 expects 2 and has: %d\n", counter);
    pthread_mutex_unlock(&lock);
}

void* thr2(void* arg){
    pthread_mutex_lock(&lock); pthread_mutex_lock(&lock);
    counter = 10;
    printf("Thread 2 has started with counter %d\n",counter);
    for (long unsigned i = 0; i < (0xFFF0000); i++);
    counter += 1;
    pthread_mutex_unlock(&lock); pthread_mutex_unlock(&lock);
    printf("Thread 2 expects 11 and has: %d\n", counter);
}
void main(){
    pthread_mutexattr_t attr;
    pthread_mutexattr_settype(&attr,PTHREAD_MUTEX_RECURSIVE);
    pthread_mutex_init(&lock, &attr);
    pthread_create(&(tid[0]), NULL, thr1,NULL);
    pthread_create(&(tid[1]), NULL, thr2,NULL);
    pthread_join(tid[0], NULL); pthread_join(tid[1], NULL);
    pthread_mutex_destroy(&lock);
}
```