Mutex e Semafori in Linux (POSIX)

Laboratorio di Reti e Sistemi Distribuiti

April 9, 2025

1 Introduzione

Mutex e semafori sono meccanismi fondamentali per la sincronizzazione dei thread nei sistemi operativi. In Linux, l'interfaccia POSIX fornisce strumenti per utilizzare questi meccanismi in modo portabile ed efficiente, sfruttando le istruzioni atomiche del processore (test-and-set).

2 Mutex (Mutual Exclusion)

Un mutex serve a garantire l'accesso esclusivo a una sezione critica del codice, evitando condizioni di race.

2.1 Funzioni principali

- `pthread_mutex_init`: Inizializza un mutex.
- `pthread_mutex_lock`: Acquisisce un mutex (blocca se già occupato).
- `pthread_mutex_unlock`: Rilascia un mutex.
- `pthread_mutex_destroy`: Dealloca le risorse del mutex.

2.2 Esempio

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

pthread_mutex_t lock;

void *thread_func(void *arg) {
    pthread_mutex_lock(&lock);
    printf("Thread in sezione critica\n");
    pthread_mutex_unlock(&lock);
    return NULL;
}

int main() {
    pthread_t t1, t2;
    pthread_mutex_init(&lock, NULL);

    pthread_create(&t1, NULL, thread_func, NULL);
    pthread_create(&t2, NULL, thread_func, NULL);

    pthread_join(t1, NULL);
    pthread_join(t2, NULL);
```

```
    pthread_mutex_destroy(&lock);
    return 0;
}
```

3 Semafori

I semafori sono contatori sincronizzati usati per controllare l'accesso a risorse condivise.

3.1 Tipi di semafori

- **Semafori POSIX unnamed**: usati tra thread nello stesso processo.
- **Semafori POSIX named**: usati tra processi diversi.

3.2 Funzioni principali (semafori unnamed)

- `sem_init`: Inizializza un semaforo.
- `sem_wait`: Decrementa il semaforo (attende se è zero).
- `sem_post`: Incrementa il semaforo (sblocca eventualmente un thread in attesa).
- `sem_destroy`: Distrugge il semaforo.

3.3 Esempio

```
#include <semaphore.h>
#include <pthread.h>
#include <stdio.h>

sem_t sem;

void *thread_func(void *arg) {
    sem_wait(&sem);
    printf("Thread accede alla risorsa\n");
    sem_post(&sem);
    return NULL;
}

int main() {
    pthread_t t1, t2;
    sem_init(&sem, 0, 1); // 1 = risorsa disponibile

    pthread_create(&t1, NULL, thread_func, NULL);
    pthread_create(&t2, NULL, thread_func, NULL);

    pthread_join(t1, NULL);
    pthread_join(t2, NULL);

    sem_destroy(&sem);
    return 0;
}
```

4 Differenze tra Mutex e Semafori

- **Mutex**: progettato per l'esclusione reciproca, un solo thread può detenere il lock.
- **Semaforo**: può consentire a più thread di accedere a una risorsa (es. contatore maggiore di 1).
- **Uso tipico**: mutex per proteggere sezioni critiche; semafori per gestire pool di risorse o sincronizzazione complessa.