```
#include <Pthread.h>

Pthread_t th1;

Pthread_create (&th1, NULL, threadProc, (void *)&arg)

Pthread_join (th1, NULL);

void threadProc (void * ptr)

{
    int * num = (int *) ptr

    cout << *num
    =
    <
    -

}
```

Mutex → `#include <pthread.h>`

`pthread_mutex_t lock; // global`

init once in main → `Pthread_mutex_init (&lock, NULL);`

lock use in CS → `Pthread_mutex_lock (&lock)`

unlock use in once done → `Pthread_mutex_unlock (&lock)`

CS

delete once done → `Pthread_mutex_destroy (&lock)`

Semaphore → #include <semaphore.h>

sem_t lock; // global

init once in main → sem_init(&lock, 0, 1)

↳ 0 → semaphore is for thread (initial value)

1 + —"— process

before CS → sem_wait(&lock);

after CS → sem_post(&lock);

destroy in → sem_destroy(lock);