**Name : Ayushi Bindal** **Roll no : 1/23/SET/BCS/341** **Class & Section : 4CSF2**

# Experiment – 12

## Aim: Program for Process Synchronization using mutix lock.

vi mutex.c

```
#include <pthread.h>
#include <stdio.h>
#include <unistd.h>
void *fun1();
void *fun2();
int shared = 1;      // Shared variable
pthread_mutex_t l;   // Mutex lock
int main() {
   pthread_t thread1, thread2;
   pthread_create(&thread1, NULL, fun1, NULL);
   pthread_create(&thread2, NULL, fun2, NULL);
   pthread_join(thread1, NULL);
   pthread_join(thread2, NULL);
   printf("Final value of shared is: %d\n", shared);
   return 0;}
void *fun1() {
   int x;
   printf("Thread1 trying to acquire lock\n");
   pthread_mutex_lock(&l);
   printf("Thread1 acquired lock\n");
   x = shared;
   printf("Thread1 reads the shared variable as:
%d\n", x);
   x++;
   printf("Local updation by thread1: %d\n", x);
   sleep(1);
   shared = x;
   printf("Value of shared variable updated by
Thread1 is: %d\n", shared);
   pthread_mutex_unlock(&l);
   printf("Thread1 released the lock");
}
void *fun2() {
   int y;
   printf("Thread2 trying to acquire lock\n");
   pthread_mutex_lock(&l);
   printf("Thread2 acquired lock\n");
   y = shared;
   printf("Thread2 reads the shared variable as:
%d\n", y);
   y++;
   printf("Local updation by thread2: %d\n", y);
   sleep(1);
   shared = y;
   printf("Value of shared variable updated by
Thread2 is: %d\n", shared);
   pthread_mutex_unlock(&l);
   printf("Thread2 released the lock");}
gcc mutex.c -o mutex -lpthread
./mutex
```

```
localhost:~/Ayushi# vi mutex.c
#include <pthread.h>
#include <stdio.h>
#include <unistd.h>
void *fun1();
void *fun2();
int shared = 1;      // Shared variable
pthread_mutex_t l;   // Mutex lock
int main() {
    pthread_t thread1, thread2;
    pthread_create(&thread1, NULL, fun1, NULL);
    pthread_create(&thread2, NULL, fun2, NULL);
    pthread_join(thread1, NULL);
    pthread_join(thread2, NULL);
    printf("Final value of shared is: %d\n", shared);
    return 0;
}
void *fun1() {
    int x;
    printf("Thread1 trying to acquire lock\n");
    pthread_mutex_lock(&l); // Thread acquires lock
    printf("Thread1 acquired lock\n");
    x = shared;
    printf("Thread1 reads the shared variable as: %d\n", x);
    x++;
    printf("Local updation by thread1: %d\n", x);
    sleep(1); // Thread1 is preempted by Thread2
    shared = x;  // Thread1 updates the value of shared
    printf("Value of shared variable updated by Thread1 is: %d\n", shared);
    pthread_mutex_unlock(&l);
    printf("Thread1 released the lock");
}
void *fun2() {
    int y;
    printf("Thread2 trying to acquire lock\n");
    pthread_mutex_lock(&l); // Thread acquires lock
    printf("Thread2 acquired lock\n");
    y = shared;
    printf("Thread2 reads the shared variable as: %d\n", y);
    y++;
    printf("Local updation by thread2: %d\n", y);
    sleep(1); // Thread2 is preempted by Thread1
    shared = y;  // Thread2 updates the value of shared
    printf("Value of shared variable updated by Thread2 is: %d\n", shared);
    pthread_mutex_unlock(&l);
    printf("Thread2 released the lock");
}
localhost:~/Ayushi# gcc mutex.c -o mutex -lpthread
localhost:~/Ayushi# ./mutex
Thread2 trying to acquire lock
Thread2 acquired lock
Thread2 reads the shared variable as: 1
Local updation by thread2: 2
Thread1 trying to acquire lock
Value of shared variable updated by Thread2 is: 2
Thread2 released the lockThread1 acquired lock
Thread1 reads the shared variable as: 2
Local updation by thread1: 3
Value of shared variable updated by Thread1 is: 3
Thread1 released the lockFinal value of shared is: 3
localhost:~/Ayushi#
```