## **Experiment 12**

**Aim:** Write a program for Process Synchronization using mutex lock.

```
vi mutex.c
                                                          localhost:~/aatif# vi mutex.c
#include<pthread.h>
                                                          #include<pthread.h>
                                                          #include<stdio.h>
#include<stdio.h>
                                                          #include<unistd.h>
#include<unistd.h>
                                                          void *fun1();
void *fun2();
void *fun1();
void *fun2();
                                                          int shared=1;
                                                          pthread_mutex_t 1;
int shared=1;
                                                          int main(){
pthread mutex tl;
                                                          pthread_t thread1, thread2;
int main(){
                                                          pthread_mutex_init(&1,NULL);
pthread t thread1,thread2;
                                                          pthread_create(&thread1,NULL,fun1,NULL);
pthread mutex init(&l,NULL);
                                                          pthread_create(&thread2,NULL,fun2,NULL);
                                                         pthread_join(thread1,NULL);
pthread_join(thread2,NULL);
pthread create(&thread1,NULL,fun1,NULL);
pthread create(&thread2,NULL,fun2,NULL);
                                                          printf("Final value of shared is: %d\n", shared);
pthread join(thread1, NULL);
                                                          pthread_mutex_destroy(&1);
pthread join(thread2,NULL);
                                                           return 0;
printf("Final value of shared is: %d\n",shared);
                                                          void *fun1(){
pthread mutex destroy(&l);
                                                          int x;
return 0;}
                                                          printf("Thread1 trying to acquire lock\n");
void *fun1(){
                                                          pthread_mutex_lock(&1);
                                                           printf("Thread1 acquired lock\n");
int x:
                                                          x=shared;
printf("Thread1 trying to acquire lock\n");
                                                           printf("Thread1 reads the shared variable as: %d\n",x);
pthread mutex lock(&l);
printf("Thread1 acquired lock\n");
                                                          printf("Local updation by Thread1: %d\n",x);
                                                          sleep(1);
x=shared;
                                                          shared=x;
printf("Thread1 reads the shared variable as:
                                                          printf("Value of shared variable updated by Thread1 is: %d\n", shared);
%d\n'',x);
                                                          pthread_mutex_unlock(&1);
x++;
                                                          printf("Thread1 released the lock\n");
return NULL;
printf("Local updation by Thread1: %d\n",x);
sleep(1);
                                                           /oid *fun2(){
shared=x;
                                                          int x;
printf("Value of shared variable updated by
                                                           printf("Thread2 trying to acquire lock\n");
Thread1 is: %d\n",shared);
                                                           othread_mutex_lock(&1);
pthread mutex unlock(&l);
                                                           printf("Thread2 acquired lock\n");
printf("Thread1 released the lock\n");
                                                           (=shared;
printf("Thread2 reads the shared variable as: %d\n",x);
return NULL;}
void *fun2(){
                                                           printf("Local updation by Thread2: %d\n",x);
printf("Thread2 trying to acquire lock\n");
                                                           rintf("Value of shared variable updated by Thread2 is: %d\n",shared);
                                                           thread_mutex_unlock(&1);
pthread mutex lock(&l);
                                                           rintf("Thread2 released the lock\n");
eturn NULL;
printf("Thread2 acquired lock\n");
x=shared:
printf("Thread2 reads the shared variable as:
                                                          localhost:~/aatif# gcc mutex.c -o mutex -lpthread
                                                          localhost:~/aatif# ./mutex
Thread2 trying to acquire lock
%d\n'',x);x++;
printf("Local updation by Thread2: %d\n",x);
                                                          Thread2 acquired lock
Thread2 reads the shared variable as: 1
Local updation by Thread2: 2
Thread1 trying to acquire lock
sleep(1);
shared=x:
printf("Value of shared variable updated by
                                                          Value of shared variable updated by Thread2 is: 2
Thread2 is: %d\n",shared);
                                                           Thread2 released the lock
                                                           Thread1 acquired lock
pthread mutex unlock(&l);
                                                          Thread1 reads the shared variable as: 2 Local updation by Thread1: 3
printf("Thread2 released the lock\n");
                                                          Value of shared variable updated by Thread1 is: 3
Thread1 released the lock
return NULL;}
                                                           inal value of shared is: 3
```

gcc mutex.c -o mutex -lpthread

./mutex