Experiment 12

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

vi mutex.c

```
#include<pthread.h>
#include<stdio.h>
#include<unistd.h>
void *fun1();
void *fun2();
int shared=1;
pthread_mutex_t l;
int main(){
pthread_t thread1,thread2;
pthread_mutex_init(&l,NULL);
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);
pthread_mutex_destroy(&l);
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\n");
return NULL;}
void *fun2(){
int x;
printf("Thread2 trying to acquire lock\n");
pthread_mutex_lock(&l);
printf("Thread2 acquired lock\n");
x=shared;
printf("Thread2 reads the shared variable as:
%d\n",x);x++;
printf("Local updation by Thread2: %d\n",x);
sleep(1);
shared=x;
printf("Value of shared variable updated by
Thread2 is: %d\n",shared);
pthread_mutex_unlock(&l);
printf("Thread2 released the lock\n");
return NULL;}
:wq
```

```
gcc mutex.c -o mutex -lpthread
./mutex
```

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