Lab Assignment No: 03

NAME: Vivek Vikram Pundkar ROLLNO: 77

CLASS: C BRANCH: ENTC BATCH: 3

DATE OF PERFORMANCE: 23-09-2021

Function

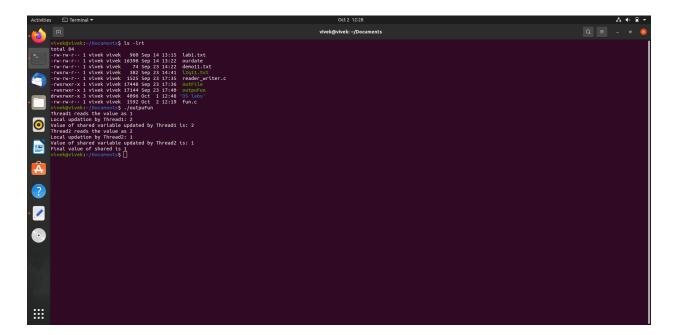
Code:

```
#include<pthread.h>
#include<stdio.h>
#include<semaphore.h>
#include<unistd.h>
void *fun1();
void *fun2();
int shared=1; //shared variable
sem t s; //semaphore variable
int main()
sem init(&s,0,1); //initialize semaphore variable - 1st argument is address of variable, 2nd is number of
processes sharing semaphore, 3rd argument is the initial value of semaphore variable
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); //prints the last updated value of shared variable
}
void *fun1()
        int x;
        sem wait(&s); //executes wait operation on s
        x=shared;//thread1 reads value of shared variable
        printf("Thread1 reads the value as %d\n",x);
        x++; //thread1 increments its value
        printf("Local updation by Thread1: %d\n",x);
        sleep(1); //thread1 is preempted by thread 2
        shared=x; //thread one updates the value of shared variable
```

```
printf("Value of shared variable updated by Thread1 is: %d\n", shared);
sem_post(&s);

void *fun2()
{
    int y;
    sem_wait(&s);
    y=shared;//thread2 reads value of shared variable
    printf("Thread2 reads the value as %d\n",y);
    y--; //thread2 increments its value
    printf("Local updation by Thread2: %d\n",y);
    sleep(1); //thread2 is preempted by thread 1
    shared=y; //thread2 updates the value of shared variable
    printf("Value of shared variable updated by Thread2 is: %d\n",shared);
    sem_post(&s);
}
```

Output:



Reader Writer

Code:

```
#include <pthread.h>
#include <semaphore.h>
#include <stdio.h>
sem t wrt;
pthread mutex t mutex;
int cnt = 1;
int numreader = 0;
void *writer(void *wno)
       sem wait(&wrt);
       cnt = cnt*2;
       printf("Writer %d modified cnt to %d\n",(*((int *)wno)),cnt);
       sem post(&wrt);
void *reader(void *rno)
       // Reader acquire the lock before modifying numreader
       pthread mutex lock(&mutex);
       numreader++;
       if(numreader == 1) {
       sem wait(&wrt); // If this id the first reader, then it will block the writer
       pthread mutex unlock(&mutex);
       // Reading Section
       printf("Reader %d: read cnt as %d\n",*((int *)rno),cnt);
       // Reader acquire the lock before modifying numreader
       pthread mutex lock(&mutex);
       numreader--;
       if(numreader == 0) {
       sem post(&wrt); // If this is the last reader, it will wake up the writer.
       }
       pthread mutex unlock(&mutex);
}
int main()
        pthread t read[10],write[5];
       pthread mutex init(&mutex, NULL);
       sem init(&wrt,0,1);
```

```
int a[10] = {1,2,3,4,5,6,7,8,9,10};

for(int i = 0; i < 10; i++) {
  pthread_create(&read[i], NULL, (void *)reader, (void *)&a[i]);
  }
  for(int i = 0; i < 5; i++) {
    pthread_create(&write[i], NULL, (void *)writer, (void *)&a[i]);
  }

for(int i = 0; i < 10; i++) {
    pthread_join(read[i], NULL);
  }
  for(int i = 0; i < 5; i++) {
    pthread_join(write[i], NULL);
  }

  pthread_mutex_destroy(&mutex);
  sem_destroy(&wrt);

  return 0;</pre>
```

Output:

}