OS LAB 7

REG: 180905478

NAME :SAGNIK CHATTERJEE

ROLL NO: 61

SEC : B

Q1.

#include<stdio.h>

#include<stdlib.h>

#include<pthread.h>

#include<semaphore.h>

#include<unistd.h>

int buf[10], f = 0,r = 0, item, val;

sem\_t mutex, full, empty;

void \*producer(void \*arg){

int i;

for(i = 0;i < 10; i++){

sem\_wait(&empty);

sem\_wait(&mutex);

printf("Produced item is %d\n", i);

buf[(++r) % 10] = i;

sleep(1);

sem\_post(&mutex);

sem\_post(&full);

sem\_getvalue(&full, &val);

printf("Full : %d \n", val);

}

}

void \*consumer(void \*arg){

int i;

for(i = 0; i < 10; i++){

sem\_getvalue(&full, &val);

printf("Full:%d \n", val);

sem\_wait(&full);

sem\_wait(&mutex);

item = buf[(++f) % 10];

printf("Consuming the item %d\n", item);

sleep(1);

sem\_post(&mutex);

sem\_post(&empty);

}

}

int main(int argc, char const \*argv[]){

pthread\_t tid1, tid2;

sem\_init(&mutex, 0, 1);

sem\_init(&full, 0, 1);

sem\_init(&empty, 0, 10);

pthread\_create(&tid1, NULL, producer, NULL);

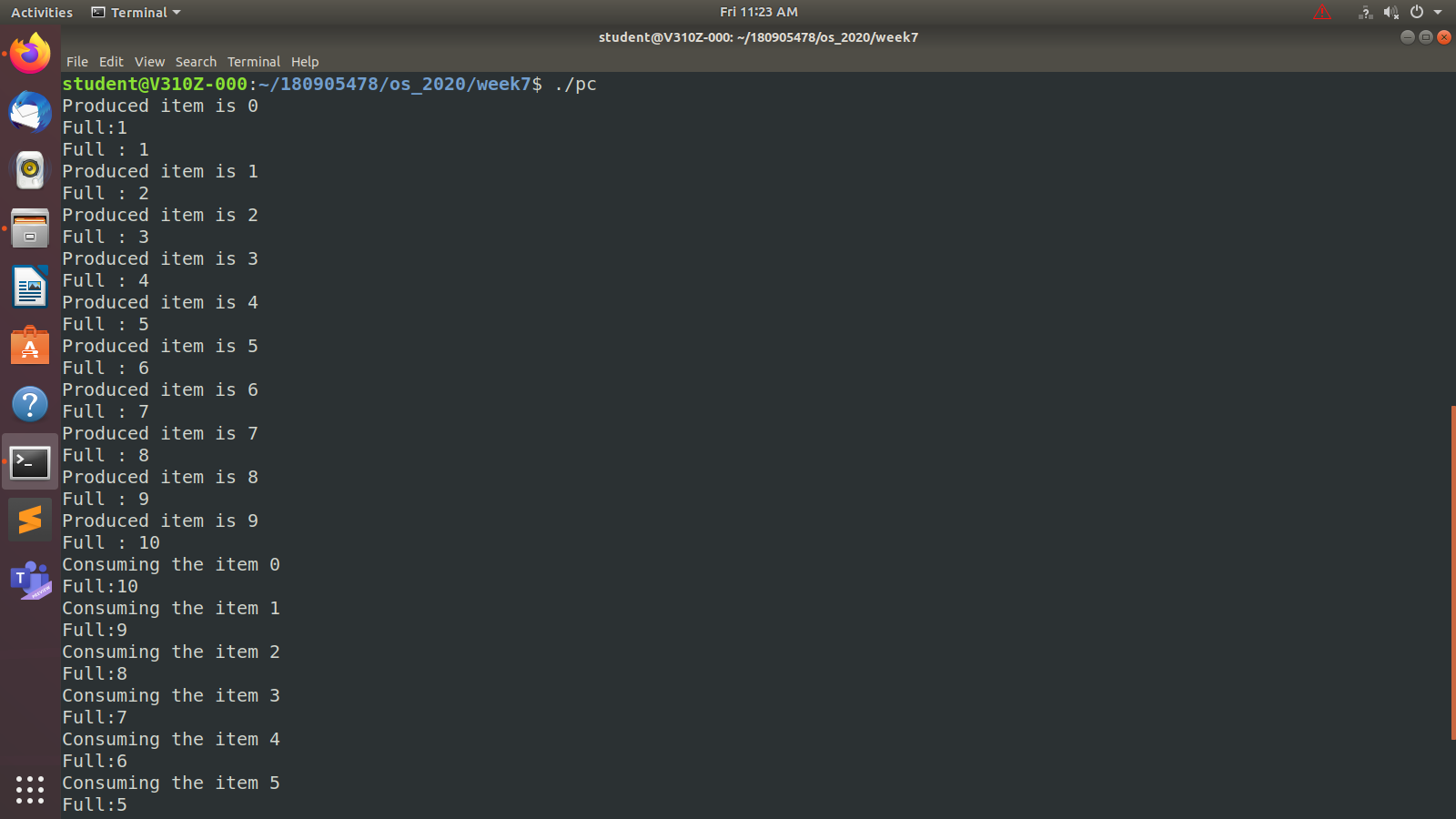
pthread\_create(&tid2, NULL, consumer, NULL);

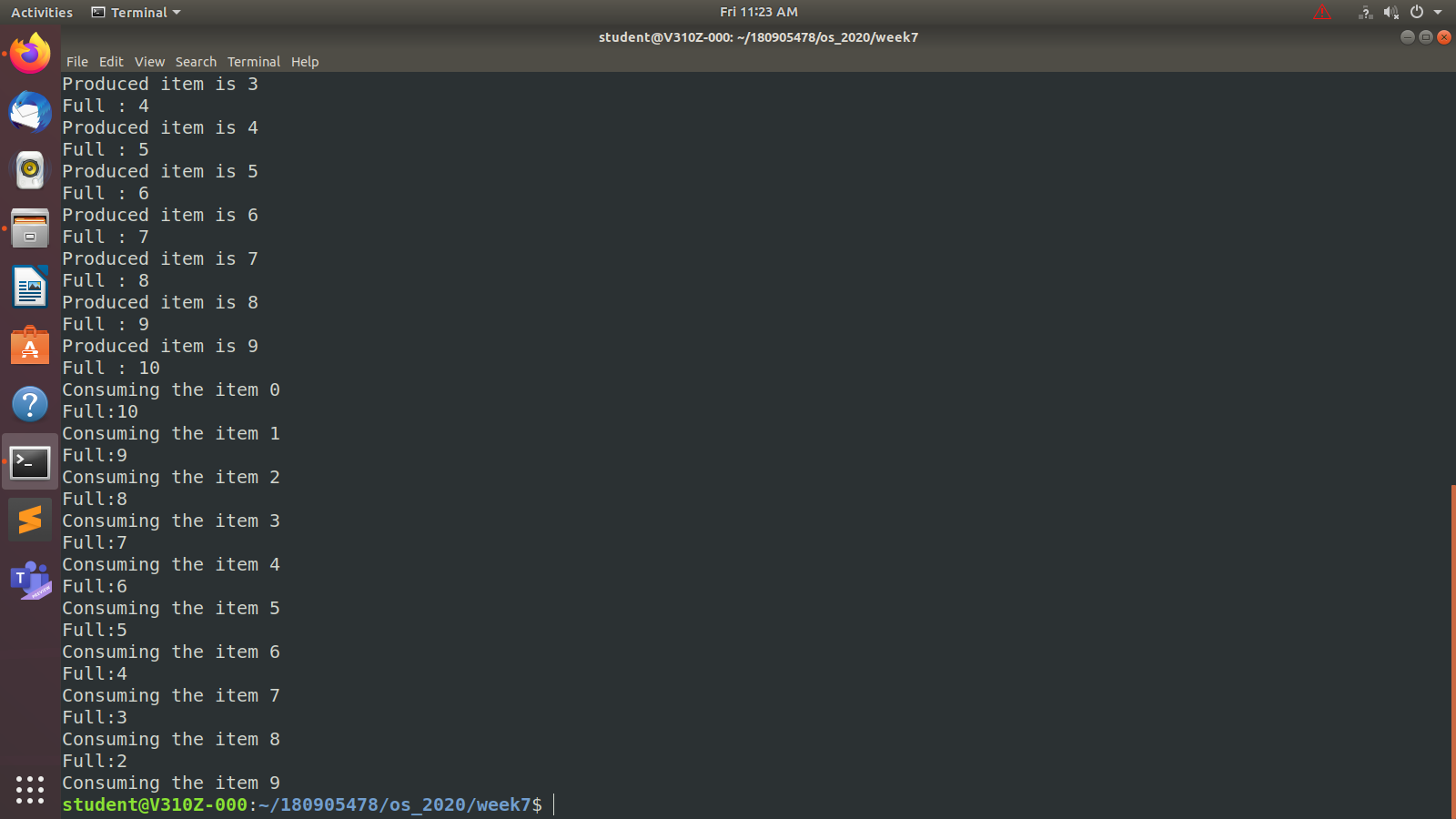
pthread\_join(tid1, NULL);

pthread\_join(tid2, NULL);

return 0;

}





Q2

#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 \*= 2;

printf("Writer %d modified 'cnt' to %d\n", (\*((int \*)wno)), cnt);

sem\_post(&wrt);

}

void \*reader(void \*rno){

pthread\_mutex\_lock(&mutex);

numreader++;

if(numreader == 1)

sem\_wait(&wrt);

pthread\_mutex\_unlock(&mutex);

printf("Reader %d: read 'cnt' as %d\n",\*((int \*)rno),cnt);

pthread\_mutex\_lock(&mutex);

numreader--;

if(numreader == 0)

sem\_post(&wrt);

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, reader, &a[i]);

for(int i = 0; i < 5; i++)

pthread\_create(&write[i], NULL, writer, &a[i]);

for(int i = 0; i < 10; i++)

pthread\_join(read[i], NULL);

for(int i = 0; i < 5; i++)

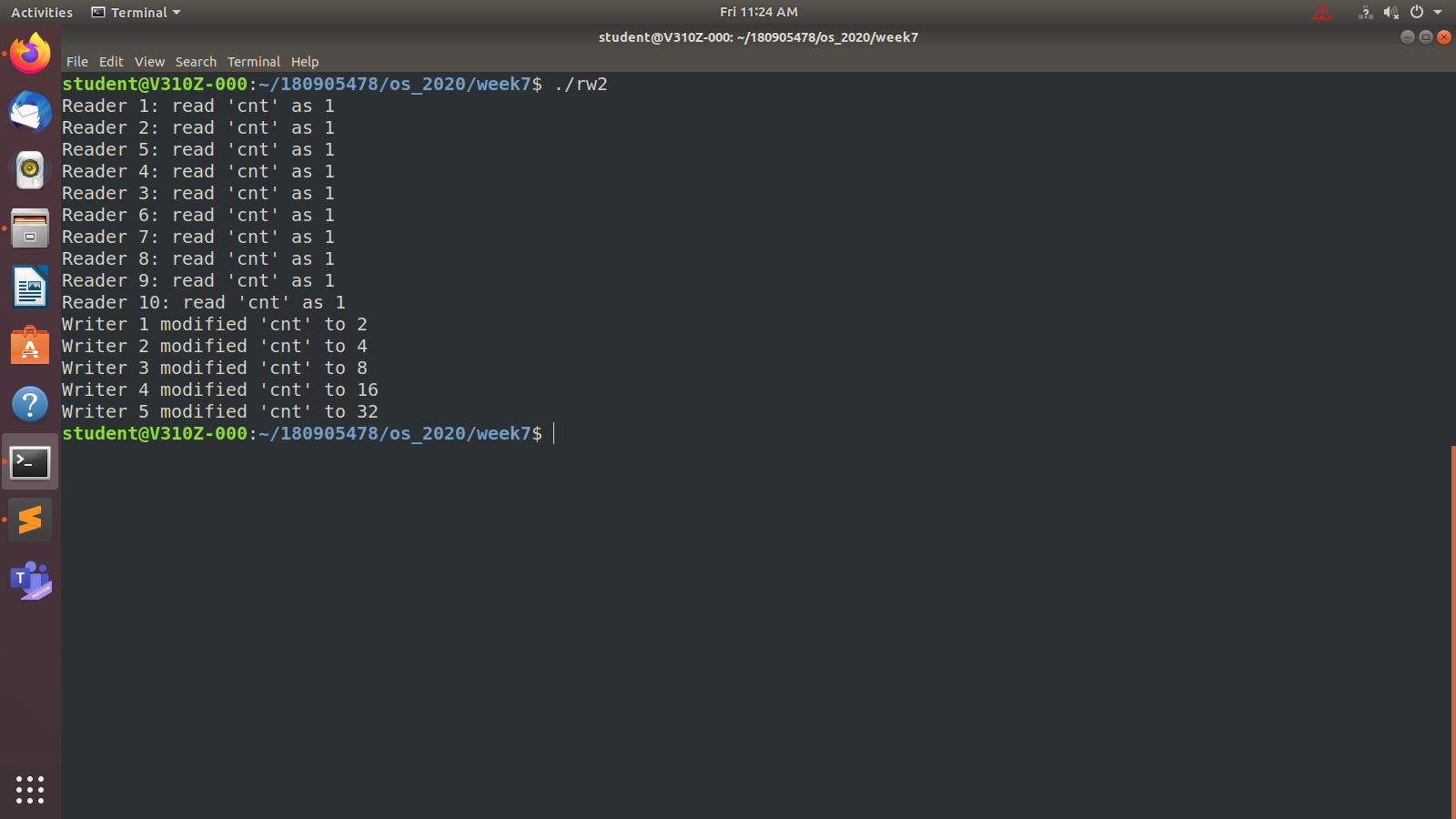
pthread\_join(write[i], NULL);

pthread\_mutex\_destroy(&mutex);

sem\_destroy(&wrt);

return 0;

}



Q3

#include <stdio.h>

#include <unistd.h>

#include <pthread.h>

#include <sys/sem.h>

#define PERMS 0660

int semId;

int initSem(int semId, int semNum, int initValue) {

return semctl(semId, semNum, SETVAL, initValue);

}

int P(int semId, int semNum) {

struct sembuf operationList[1];

operationList[0].sem\_num = semNum;

operationList[0].sem\_op = -1;

operationList[0].sem\_flg = 0;

return semop(semId, operationList, 1);

}

int V(int semId, int semNum) {

struct sembuf operationList[1];

operationList[0].sem\_num = semNum;

operationList[0].sem\_op = 1;

operationList[0].sem\_flg = 0;

return semop(semId, operationList, 1);

}

void\* funcA(void\* nothing) {

printf("Thread A try to lock 0...\n");

P(semId, 0);

printf("Thread A locked 0.\n");

usleep(50\*1000);

printf("Thread A try to lock 1...\n");

P(semId, 1);

printf("Thread A locked 1.\n");

V(semId, 0);

V(semId, 1);

return NULL;

}

void\* funcB(void\* nothing) {

printf("Thread B try to lock 1...\n");

P(semId, 1);

printf("Thread B locked 1.\n");

usleep(5\*1000);

printf("Thread B try to lock 0...\n");

P(semId, 0);

printf("Thread B locked 0.\n");

V(semId, 0);

V(semId, 1);

return NULL;

}

int main(int argc, char\* argv[]) {

int i;

semId = semget(ftok(argv[0], 'A'), 2, IPC\_CREAT | PERMS);

initSem(semId, 0, 1);

initSem(semId, 1, 1);

pthread\_t thread[2];

pthread\_create(&thread[0], NULL, funcA, NULL);

pthread\_create(&thread[1], NULL, funcB, NULL);

for (i = 0 ; i < 2 ; i++) {

pthread\_join(thread[i], NULL);

}

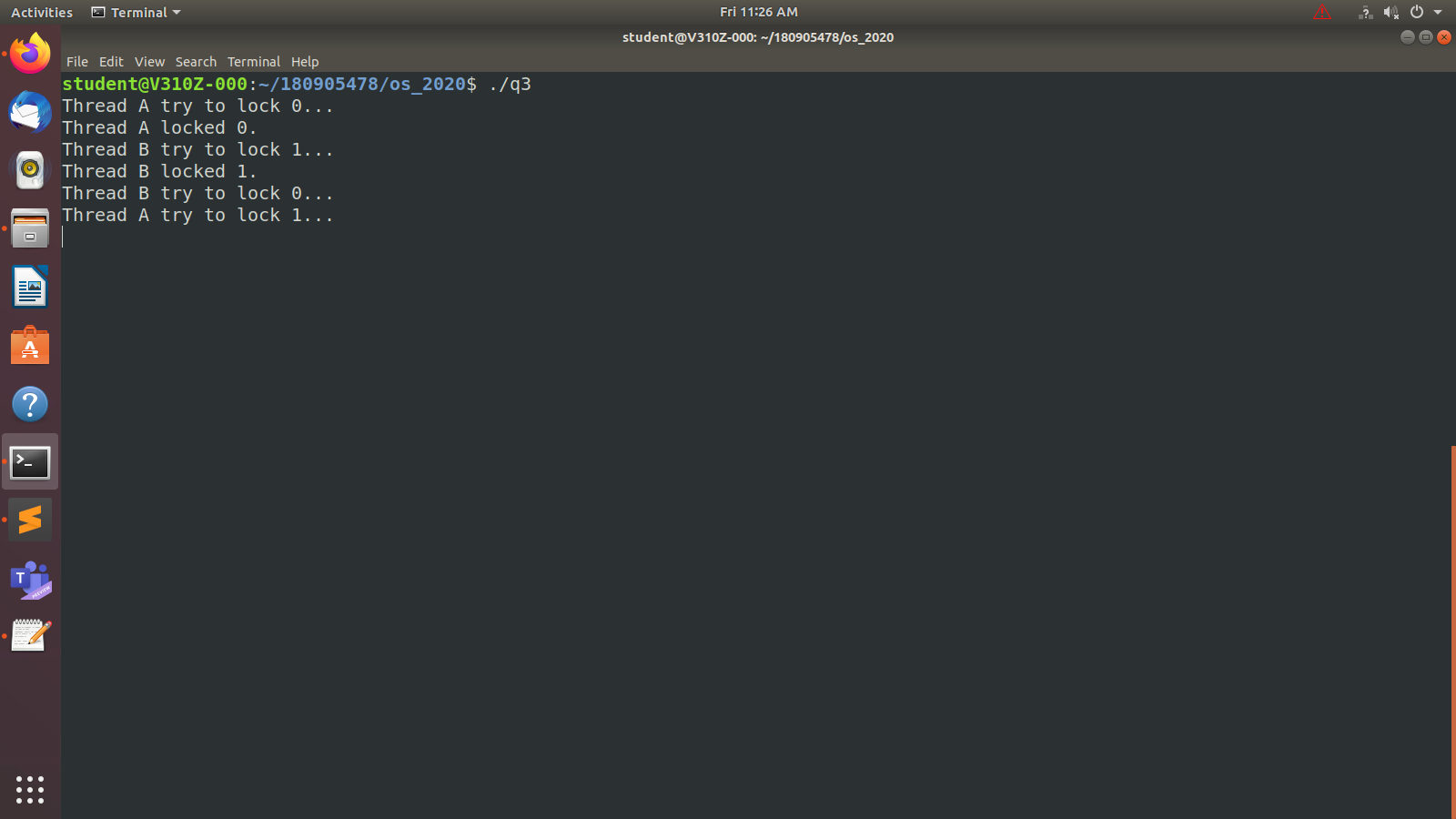
printf("This is not printed in case of deadlock\n");

semctl(semId, 0, IPC\_RMID, 0);

semctl(semId, 1, IPC\_RMID, 0);

return 0;

}



Q4

#include <stdio.h>

#include <unistd.h>

#include <stdlib.h>

#include <pthread.h>

#include <semaphore.h>

#define MAX 20

void \*client(void \*param);

void \*barber(void \*param);

sem\_t chairs\_mutex;

sem\_t sem\_client;

sem\_t sem\_barber;

int num\_chairs;

int clientWait;

int main(int argc, char \*argv[]) {

pthread\_t barberid;

pthread\_t clientids[MAX];

printf("Main thread beginning\n");

int runTime,clients,i;

if (argc != 5){

printf("Please enter 4 arguments: <Program run time> <Number of clients>\n");

printf("<Number of chairs> <Client wait time>\n");

exit(0);

}

runTime = atoi(argv[1]);

clients = atoi(argv[2]);

num\_chairs = atoi(argv[3]);

clientWait = atoi(argv[4]);

sem\_init(&chairs\_mutex,0,1);

sem\_init(&sem\_client,0,0);

sem\_init(&sem\_barber,0,0);

pthread\_create(&barberid, NULL, barber, NULL);

printf("Creating barber thread with id %lu\n",barberid);

for (i = 0; i < clients; i++){

pthread\_create(&clientids[i], NULL, client, NULL);

printf("Creating client thread with id %lu\n",clientids[i]);

}

printf("Main thread sleeping for %d seconds\n", runTime);

sleep(runTime);

printf("Main thread exiting\n");

exit(0);

}

void \*barber(void \*param) {

int worktime;

while(1) {

sem\_wait(&sem\_client);

sem\_wait(&chairs\_mutex);

num\_chairs += 1;

printf("Barber: Taking a client. Number of chairs available = %d\n",num\_chairs);

sem\_post(&sem\_barber);

sem\_post(&chairs\_mutex);

worktime = (rand() % 4) + 1;

printf("Barber: Cutting hair for %d seconds\n", worktime);

sleep(worktime);

}

}

void \*client(void \*param) {

int waittime;

while(1) {

sem\_wait(&chairs\_mutex);

if(num\_chairs <= 0){

printf("Client: Thread %u leaving with no haircut\n", (unsigned int)pthread\_self());

sem\_post(&chairs\_mutex);

}

else{

num\_chairs -= 1;

printf("Client: Thread %u Sitting to wait. Number of chairs left = %d\n",(unsigned int)pthread\_self(),num\_chairs);

sem\_post(&sem\_client);

sem\_post(&chairs\_mutex);

sem\_wait(&sem\_barber);

printf("Client: Thread %u getting a haircut\n",(unsigned int)pthread\_self());

}

waittime = (rand() % clientWait) + 1;

printf("Client: Thread %u waiting %d seconds before attempting next haircut\n",(unsigned int)pthread\_self(),waittime);

sleep(waittime);

}

}

