Name: Aman Tiwari

Roll No: A-05

Section: K18MS

Registration No: 11803731

GITHUB link:

Q1)

#include<stdio.h>

#include<unistd.h>

#include<pthread.h>

#include<stdlib.h>

#include<string.h>

int flag[2];

int turn;

const int MAX = 1e9;

int ans = 0;

void lock\_init()

{

flag[0]=flag[1]=0;

turn = 0;

}

void lock(int self)

{

flag[self]=1;

turn = 1-self;

while(flag[1-self]==1 && turn == 1-self);

}

void unlock(int self)

{

flag[self]=0;

}

void\* func(void \*s)

{

int i=0;

int \*limitptr = (int\*) s;

int self = \*limitptr;

printf("Thread %d in queue for critical section\n",self);

lock(self);

printf("Thread %d in critical section\n",self);

for(i=0;i<MAX;i++)

{

ans++;

}

printf("Thread %d done counting\n",self);

printf("Thread %d is exiting critical section\n",self);

unlock(self);

}

int main()

{

pthread\_t p1, p2;

int a=0,b=1;

lock\_init();

pthread\_create(&p1, NULL, func, &a);

pthread\_create(&p2, NULL, func, &b);

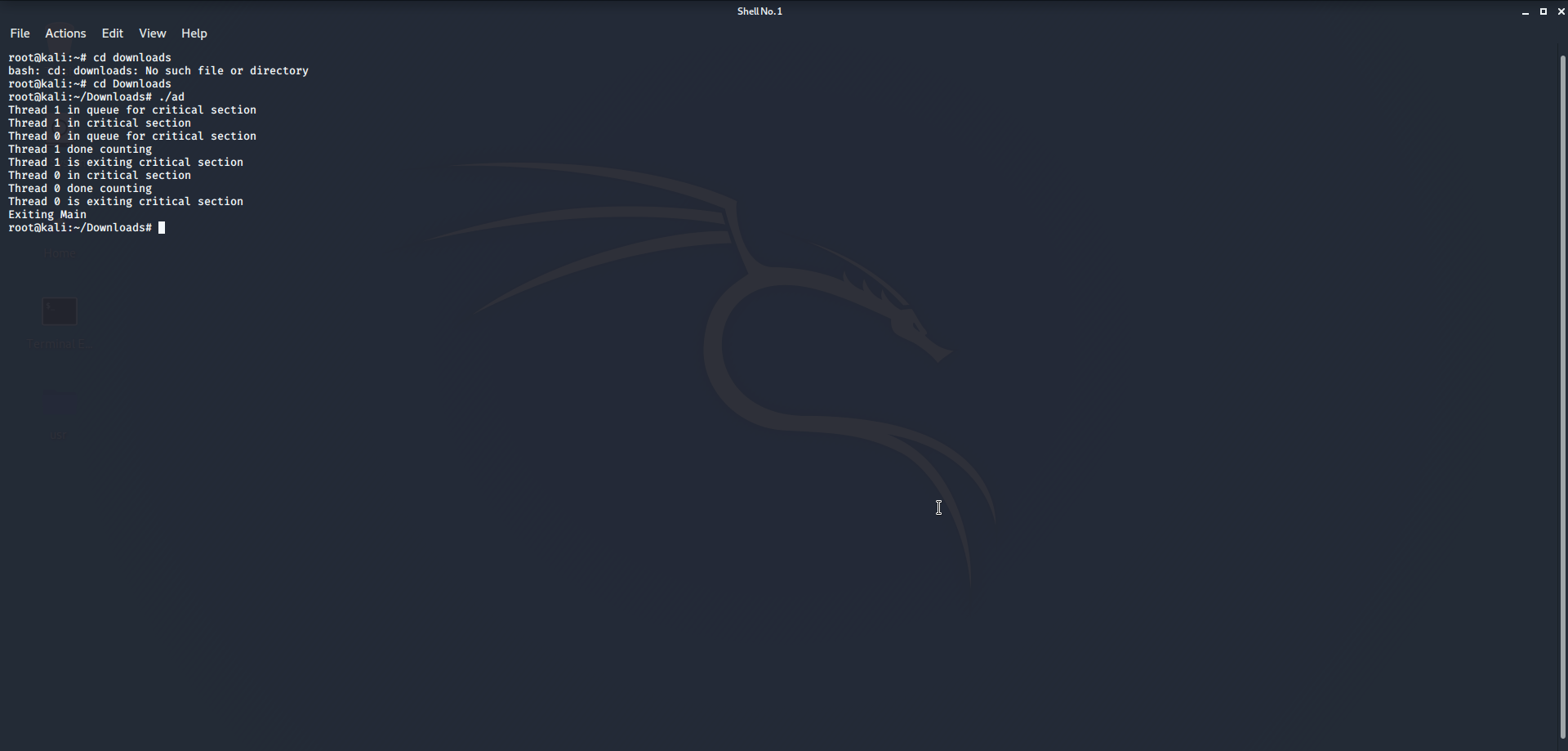
pthread\_join(p1, NULL);

pthread\_join(p2, NULL);

printf("Exiting Main\n");

return 0;

}



Q2)

#include<stdio.h>

#include<pthread.h>

int global[2];

void \*sum\_thread(void \*arg)

{

int \*args\_array;

args\_array = arg;

int n1,n2,sum;

n1=args\_array[0];

n2=args\_array[1];

sum = n1+n2;

printf("Hello!!! Welcome to LPU\n");

printf("Sum = %d\n",sum);

return NULL;

}

int main()

{

printf("First number: ");

scanf("%d",&global[0]);

printf("Second number: ");

scanf("%d",&global[1]);

pthread\_t tid\_sum, tid1;

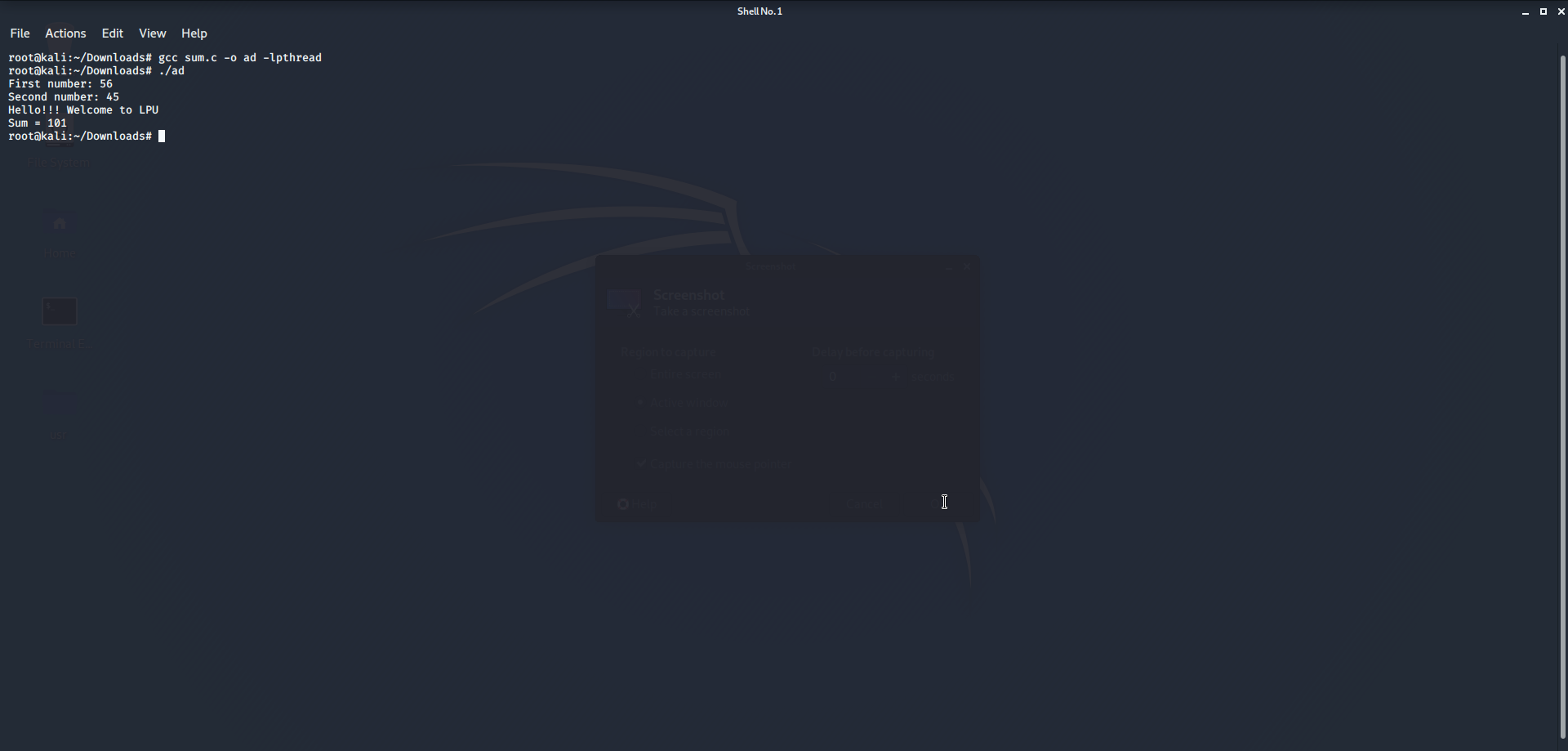
pthread\_create(&tid\_sum,NULL,sum\_thread,global);

pthread\_join(tid\_sum,NULL);

pthread\_create(&tid1, NULL, sum\_thread,NULL);

return 0;

}



Q3)

Process 1

#include <signal.h>

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include <sys/ipc.h>

#include <sys/shm.h>

#include <sys/types.h>

#include <unistd.h>

#define FILLED 0

#define Ready 1

#define NotReady -1

struct memory {

char buff[100];

int status, pid1, pid2;

};

struct memory\* shmptr;

void handler(int signum)

{

if (signum == SIGUSR1) {

printf("Received User2: ");

puts(shmptr->buff);

}

}

int main()

{

int pid = getpid();

int shmid;

int key = 12345;

shmid = shmget(key, sizeof(struct memory), IPC\_CREAT | 0666);

shmptr = (struct memory\*)shmat(shmid, NULL, 0);

shmptr->pid1 = pid;

shmptr->status = NotReady;

signal(SIGUSR1, handler);

while (1) {

while (shmptr->status != Ready)

continue;

sleep(1);

printf("User1: ");

fgets(shmptr->buff, 100, stdin);

shmptr->status = FILLED;

kill(shmptr->pid2, SIGUSR2);

}

shmdt((void\*)shmptr);

shmctl(shmid, IPC\_RMID, NULL);

return 0;

}

Process 2

#include <signal.h>

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include <sys/ipc.h>

#include <sys/shm.h>

#include <sys/types.h>

#include <unistd.h>

#define FILLED 0

#define Ready 1

#define NotReady -1

struct memory {

char buff[100];

int status, pid1, pid2;

};

struct memory\* shmptr;

void handler(int signum)

{

if (signum == SIGUSR2) {

printf("Received From User1: ");

puts(shmptr->buff);

}

}

int main()

{

int pid = getpid();

int shmid;

int key = 12345;

shmid = shmget(key, sizeof(struct memory), IPC\_CREAT | 0666);

shmptr = (struct memory\*)shmat(shmid, NULL, 0);

shmptr->pid2 = pid;

shmptr->status = NotReady;

signal(SIGUSR2, handler);

while (1) {

sleep(1);

printf("User2: ");

fgets(shmptr->buff, 100, stdin);

shmptr->status = Ready;

kill(shmptr->pid1, SIGUSR1);

while (shmptr->status == Ready)

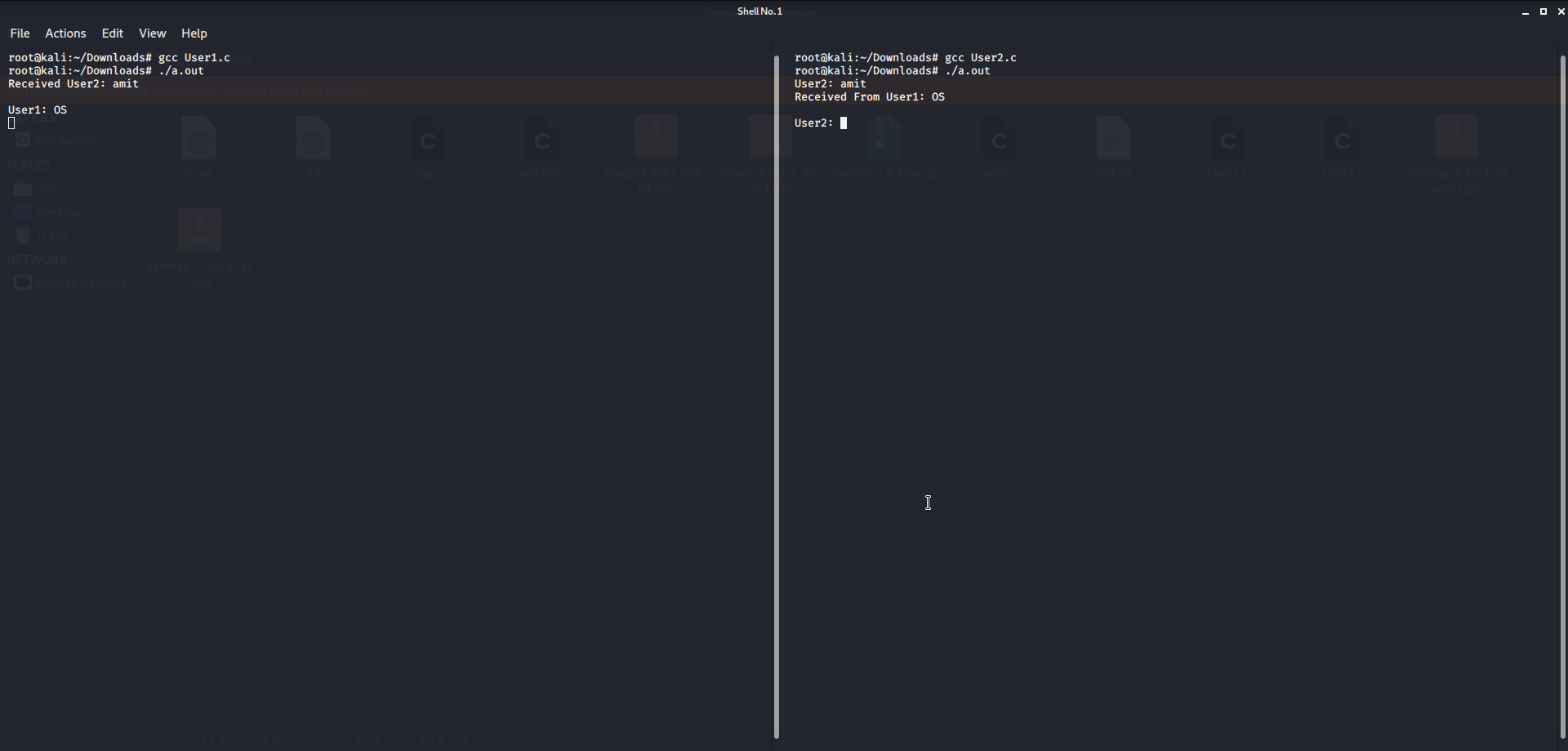
continue;

}

shmdt((void\*)shmptr);

return 0;

}



Q4)

#include<stdio.h>

#include<dirent.h>

#include<stdio.h>

#include<unistd.h>

#include<fcntl.h>

#include<sys/types.h>

#include<sys/stat.h>

#include<stdlib.h>

int main() {

FILE \*fp;

char ch;

long length;

fp = fopen("test.txt", "r");

if (fp == NULL) {

puts("cannot open this file");

exit(1);

}

fseek(fp, 0, SEEK\_END);

length = ftell(fp);

fseek(fp, (length-5), SEEK\_SET);

do {

ch = fgetc(fp);

putchar(ch);

} while (ch != EOF);

fclose(fp);

return(0);

}

