OS Lab 7

Aniruddha Amit Dutta

180905488

Roll no – 58

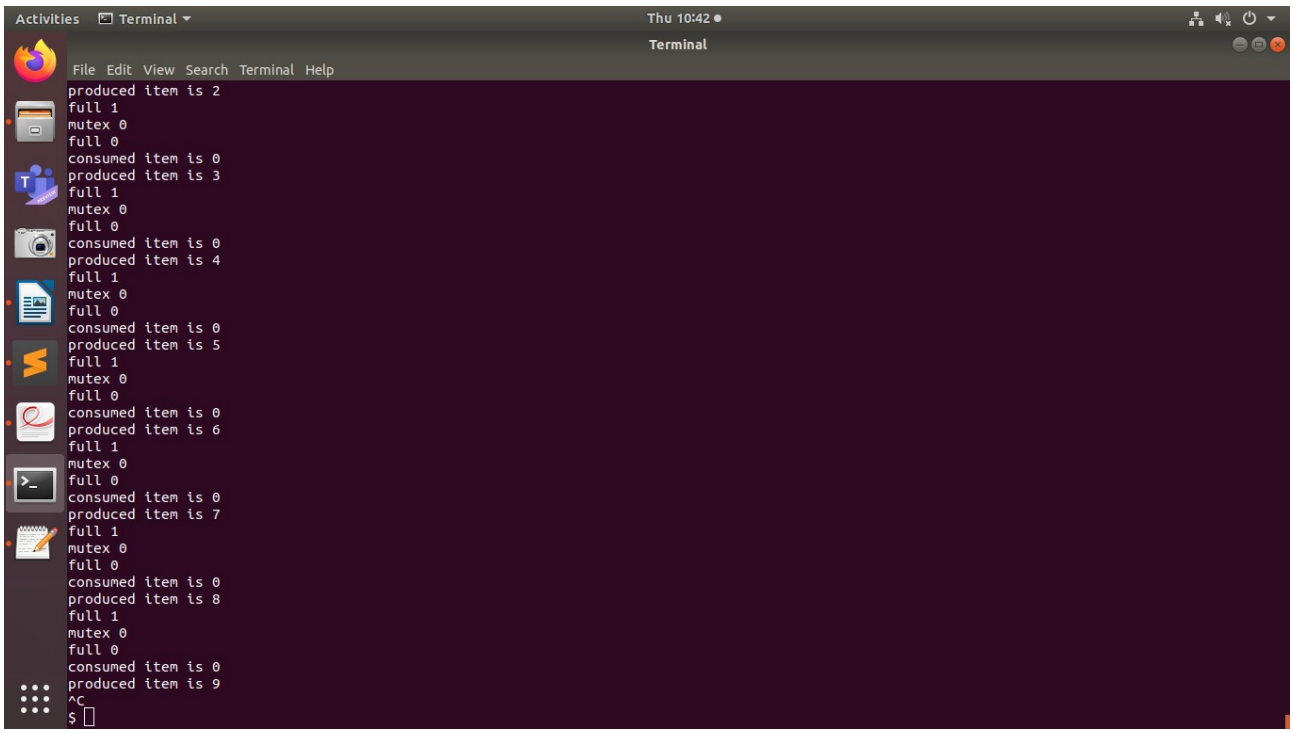
Q1.

```
#include <stdio.h>
#include <pthread.h>
#include <semaphore.h>
#include <unistd.h>
#define SZ 10
#define loop 20
int buf[SZ],f,r;

sem_t full,mutex,empty;

void *produce(void *arg){
	int i;
	for (int i = 0; i < loop; ++i)
	{
		sem_wait(&empty);
		sem_wait(&mutex);
		printf("produced item is %d \n", i);
		buf[(++r)%SZ]=i;
		sleep(1);
		sem_post(&mutex);
		sem_post(&full);


		int value;
	sem_getvalue(&full, &value);
	printf("full %d\n",value);
	// int value2;
	// sem_getvalue(&mutex, &value2);
	// printf("mutex %d\n",value2);
		// printf("full %ld\n", full);
	}

}
```

```
void *consume(void *arg){
	int i,item;

	for (int i = 0; i < loop; ++i)
	{
		sem_wait(&full);
		sem_wait(&mutex);

		int value;
	sem_getvalue(&full, &value);
	printf("full %d\n",value);
		// printf("full %ld\n", full);


		item=buf[(++f)%SZ];
		printf("consumed item is %d \n", item);
		sleep(1);
		sem_post(&mutex);
		sem_post(&empty);

	}
}

int main(){
	pthread_t tid1,tid2;
	sem_init(&mutex,0,1);
	sem_init(&full,0,1);
	sem_init(&empty,0,SZ);
	pthread_create(&tid1,NULL,produce,NULL);
	pthread_create(&tid2,NULL,consume,NULL);
	pthread_join(tid1,NULL);
	pthread_join(tid1,NULL);
}
```

```
mutex 29
$ gcc pc.c -lpthread
$ ./a.out
full 0
consumed item is 0
produced item is 0
full 1
mutex 0
full 0
consumed item is 1
produced item is 1
full 1
mutex 0
full 0
consumed item is 2
produced item is 2
full 1
mutex 0
full 0
consumed item is 3
produced item is 3
full 1
mutex 0
full 0
consumed item is 4
produced item is 4
full 1
mutex 0
full 0
consumed item is 5
produced item is 5
full 1
mutex 0
full 0
consumed item is 6
produced item is 6
full 1
mutex 0
```

```
produced item is 7
full 1
mutex 0
full 0
consumed item is 8
produced item is 8
full 1
mutex 0
full 0
consumed item is 9
produced item is 9
full 1
mutex 0
full 0
consumed item is 10
produced item is 10
full 1
mutex 0
full 0
consumed item is 11
produced item is 11
^C
$ gcc pc.c -lpthread
$ ./a.out
produced item is 0
full 1
mutex 0
full 1
consumed item is 0
full 0
consumed item is 0
produced item is 1
full 1
mutex 0
full 0
consumed item is 0
produced item is 2
full 1
```



Q2.

```
#include<semaphore.h>
#include<stdio.h>
#include<stdlib.h>
#include<unistd.h>
#include<pthread.h>

/*
Problem parameters:

One set of data is shared among a number of processes
Once a writer is ready, it performs its write. Only one writer may write at a time
If a process is writing, no other process can read it
If at least one reader is reading, no other process can write
Readers may not write and only read

*/


sem_t mutex,wrt;
pthread_t tid;
pthread_t writerthreads[100],readerthreads[100];
int readcount = 0;
```

```
void *reader(void* param)
{
   sem_wait(&mutex);
   readcount++;
   if(readcount==1)
      sem_wait(&wrt);
   sem_post(&mutex);
   printf("%d reader in critical sec\n",readcount);
   sleep(1);
   sem_wait(&mutex);
   readcount--;
   if(readcount==0)
   {
      sem_post(&wrt);
   }
   sem_post(&mutex);
   printf("%d reader left\n",readcount);
   return NULL;
}

void *writer(void* param)
{
   printf("writer is trying to enter\n");
   sem_wait(&wrt);
   printf("writer in critical sec\n");
   sleep(1);
   sem_post(&wrt);
   printf("writer left\n");
   return NULL;
}

int main()
{
   int rds,i;
   printf("Enter the number of readers:");
   scanf("%d",&rds);
   printf("\n");
   int n1[rds];
   sem_init(&mutex,0,1);
   sem_init(&wrt,0,1);
   for(i=0;i<rds;i++)
   {
      pthread_create(&writerthreads[i],NULL,reader,NULL);
      pthread_create(&readerthreads[i],NULL,writer,NULL);
   }
```

```
    for(i=0;i<rds;i++)
    {
        pthread_join(writerthreads[i],NULL);
        pthread_join(readerthreads[i],NULL);
    }

}
```

```
$ gcc rw.c -lpthread
$ ./a.out
Enter the number of readers:4

writer is trying to enter
writer in critical sec
writer is trying to enter
writer is trying to enter
writer is trying to enter
writer left
1 reader in critical sec
2 reader in critical sec
3 reader in critical sec
4 reader in critical sec
3 reader left
2 reader left
1 reader left
0 reader left
writer in critical sec
writer left
writer in critical sec
writer left
writer in critical sec
writer left
$
```

Q3.

```
#include <stdio.h>
#include <pthread.h>
#include <semaphore.h>
#include <unistd.h>
#include <unistd.h>
#include <sys/syscall.h>

#ifndef SYS_gettid
#error "SYS_gettid unavailable on this system"
#endif
```

```
#define gettid() ((pid_t)syscall(SYS_gettid))

sem_t mutex1,mutex2;

void *get1then2(){

	pid_t pid1 = getpid();
	printf("tid is %d pid is %d \n", gettid(),pid1 );
	int value1,value2; sem_getvalue(&mutex1, &value1); sem_getvalue(&mutex2,
&value2); printf("mutex1 %d\t",value1);   printf("mutex2 %d\n",value2);
	sem_wait(&mutex1);
	sleep(1); // acquire resource1
	printf("waiting for resource2 \n");
	sem_wait(&mutex2);

	// do the job
	printf("doing work in get1then2 \n");

	sem_post(&mutex2);
	sem_post(&mutex1);
}

void *get2then1(){

	pid_t pid2 = getpid();
	printf("tid is %d pid is %d \n", gettid(),pid2 );
	int value1,value2; sem_getvalue(&mutex1, &value1); sem_getvalue(&mutex2,
&value2); printf("mutex1 %d\t",value1);   printf("mutex2 %d\n",value2);
	sem_wait(&mutex2);
	sleep(1); // acquire resource2
	printf("waiting for resource1 \n");
	sem_wait(&mutex1);

	// do the job
	printf("doing work in get2then1 \n");

	sem_post(&mutex1);
	sem_post(&mutex2);
}

int main(){
	pthread_t tid1,tid2;
	sem_init(&mutex1,0,1);
	sem_init(&mutex2,0,1);
```

```
        pthread_create(&tid1,NULL,get1then2,NULL);
        pthread_create(&tid2,NULL,get2then1,NULL);


        pthread_join(tid1,NULL);
        pthread_join(tid1,NULL);

        printf("this should not be printed if deadlock \n");

        return 0;
}
```

output -

```
aniruddha@aniruddha-G3-3579:~/MIT/semester5/Labs/OS LAB/OS lab 7$ gcc deadlock.c -lpthread
aniruddha@aniruddha-G3-3579:~/MIT/semester5/Labs/OS LAB/OS lab 7$ ./a.out
tid is 5612 pid is 5611
mutex1 1        mutex2 1
tid is 5613 pid is 5611
mutex1 0        mutex2 1
waiting for resource2
waiting for resource1
```

Q4.

```
#include<stdio.h>
#include<stdlib.h>
#include<unistd.h>
#include<pthread.h>
#include<errno.h>
#include<sys/ipc.h>
#include<semaphore.h>
#define N 5

time_t end_time;/*end time*/
sem_t mutex,customers,barbers;/*Three semaphors*/
int count=0;/*The number of customers waiting for haircuts*/

void barber(void *arg);
void customer(void *arg);

int main(int argc,char *argv[])
{
        pthread_t id1,id2;
```

```
	int status=0;
	end_time=time(NULL)+20;/*Barber Shop Hours is 20s*/


	sem_init(&mutex,0,1);sem_init(&customers,0,0);
	sem_init(&barbers,0,1);


	status=pthread_create(&id1,NULL,(void *)barber,NULL);
	if(status!=0)
	perror("barber error!\n");

	/*Customer_thread initialization*/
	status=pthread_create(&id2,NULL,(void *)customer,NULL);
	if(status!=0)
	perror("customers error!\n");

	/*Customer_thread first blocked*/
	pthread_join(id2,NULL);
	pthread_join(id1,NULL);
	exit(0);
}

void barber(void *arg)
{
	while(time(NULL)<end_time || count>0)
	{
		sem_wait(&customers);
		sem_wait(&mutex);
		count--;
		printf("Barber:cut hair,count is:%d.\n",count);
		sem_post(&mutex);
		sem_post(&barbers);
		sleep(3);
	}
}

void customer(void *arg)
{
	while(time(NULL)<end_time)
	{
		sem_wait(&mutex);
		if(count<N)
		{
			count++;
```

```
                printf("Customer:add count,count is:%d\n",count);
                sem_post(&mutex);
                sem_post(&customers);
                sem_wait(&barbers);
            }
            else

                /*If the number is full of customers,just put the mutex lock let
go*/
                sem_post(&mutex);
            sleep(1);
        }
}
```

output-

```
aniruddha@aniruddha-G3-3579:~/Downloads/OS/Lab7$ gcc q4.c -lpthread
aniruddha@aniruddha-G3-3579:~/Downloads/OS/Lab7$ ./a.out
Customer:add count,count is:1
Barber:cut hair,count is:0.
Customer:add count,count is:1
Customer:add count,count is:2
Barber:cut hair,count is:1.
Customer:add count,count is:2
Barber:cut hair,count is:1.
Customer:add count,count is:2
Barber:cut hair,count is:1.
Customer:add count,count is:2
Barber:cut hair,count is:1.
Customer:add count,count is:2
Barber:cut hair,count is:1.
Customer:add count,count is:2
Barber:cut hair,count is:1.
Customer:add count,count is:2
Barber:cut hair,count is:1.
Barber:cut hair,count is:0.
aniruddha@aniruddha-G3-3579:~/Downloads/OS/Lab7$
```