

//Lab7

q1)

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

int buff[10], f = 0, r = 0;
sem_t mutex, full, empty;

void* produce(void* arg)
{
    int i;
    for(i = 0; i < 15; i++)
    {
        sem_wait(&empty);
        sem_wait(&mutex);
        printf("produced item is %d \n", i);
        buff[(++r) % 10] = i;
        sleep(1);
        sem_post(&mutex);
        sem_post(&full);
        //printf("full %u \n", full);
    }
}

void* consume(void* arg)
{
    int item, i;
    for(i = 0; i < 15; i++)
    {
        sem_wait(&full);
        //printf("full %u \n", full);
        sem_wait(&mutex);
        item = buff[(++f) % 10];
        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, 0);
    sem_init(&empty, 0, 10);
    pthread_create(&tid1, NULL, produce, NULL);
```

```

pthread_create(&tid2, NULL, consume, NULL);
pthread_join(tid1, NULL);
pthread_join(tid2, NULL);
}

```

The screenshot shows a Sublime Text editor window with a C program. The program uses pthreads and semaphores to implement a producer-consumer problem. The terminal window shows the output of the program, which includes the initial counts, the production of 15 items, and the consumption of 16 items.

```

1 #include<stdio.h>
2 #include<pthread.h>
3 #include<semaphore.h>
4 #include<unistd.h>
5 #include<stdlib.h>
6
7 int buff[10], f = 0, r = 0;
8 sem_t mutex, full, empty;
9
10 void* produce(void* arg)
11 {
12     int i;
13     for(i = 0; i < 15; i++)
14     {
15         sem_wait(&empty);
16         sem_wait(&mutex);
17         printf("produced item is %d \n", i);
18         buff[(f++) % 10] = i;
19         sleep(1);
20         sem_post(&mutex);
21         sem_post(&full);
22         //printf("full %u \n", full);
23     }
24 }
25
26 void* consume(void* arg)
27 {
28     int item, i;
29     for(i = 0; i < 15; i++)
30     {
31         sem_wait(&full);
32         //printf("full %u \n", full);
33         sem_wait(&mutex);
34         item = buff[f];
35         printf("consumed item is %d \n", item);
36         f++;
37         sem_post(&mutex);
38     }
39 }
40
41 int main()
42 {
43     pthread_t tid1, tid2;
44     pthread_create(&tid1, NULL, produce, NULL);
45     pthread_create(&tid2, NULL, consume, NULL);
46     pthread_join(tid1, NULL);
47     pthread_join(tid2, NULL);
48     return 0;
49 }

```

```

180905402@prg22: ~/5thSemLabs/os/lab8
writer count is: 15
count is: 15 and readcount is 1
writer count is: 16
180905402@prg22:~/5thSemLabs/os/lab8$ clear

180905402@prg22:~/5thSemLabs/os/lab8$ ./p1
produced item is 0
produced item is 1
produced item is 2
produced item is 3
produced item is 4
produced item is 5
produced item is 6
produced item is 7
produced item is 8
produced item is 9
consumed item is: 0
consumed item is: 1
consumed item is: 2
consumed item is: 3
consumed item is: 4
consumed item is: 5
consumed item is: 6
consumed item is: 7

```

q2)

```

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

```

```

int readcount;
sem_t mutex, wrt;
// mutex is used to ensure mutual exclusion
// when readcount is updated

```

```

int count = 1;

```

```

void* reader(void* arg)
{
    sem_wait(&mutex);
    readcount++;
    if(readcount == 1)
    {
        sem_wait(&wrt);
    }
    sem_post(&mutex);
}

```

```

printf("count is: %d and readcount is %d \n",count, readcount);

sem_wait(&mutex);
readcount--;
if(readcount == 0)
{
    sem_post(&wrt);
}
sem_post(&mutex);
}

void* writer(void* arg)
{

    sem_wait(&wrt);
    count++;
    printf("writer count is: %d \n", count);
    sem_post(&wrt);
}

int main()
{
    int readcount = 0;
    pthread_t tid1, tid2;
    sem_init(&mutex, 0, 1);
    sem_init(&wrt, 0, 1);

    for(int i = 0; i < 15; i++)\
    {
        pthread_create(&tid1, NULL, reader, NULL);
        pthread_create(&tid2, NULL, writer, NULL);
        pthread_join(tid1, NULL);
        pthread_join(tid2, NULL);
    }

    sem_destroy(&mutex);
    sem_destroy(&wrt);
}

```

The screenshot shows a Sublime Text editor window with a C file named `p2.c`. The code implements a reader-writer synchronization using semaphores. It includes headers for `stdio.h`, `pthread.h`, `semaphore.h`, `unistd.h`, and `stdlib.h`. It defines a shared `readcount` and a `mutex` for mutual exclusion. The `reader` function increments `readcount` and prints the current state. The `main` function creates two threads, `tid1` and `tid2`, which execute `func1` and `func2` respectively. A terminal window on the right shows the output of the program, displaying alternating counts for the writer and the readers.

```
1 #include<stdio.h>
2 #include<pthread.h>
3 #include<semaphore.h>
4 #include<unistd.h>
5 #include<stdlib.h>
6
7 int readcount;
8 sem_t mutex, wrt;
9 // mutex is used to ensure mutual exclusion
10 // when readcount is updated
11
12 int count = 1;
13
14 void* reader(void* arg)
15 {
16
17     sem_wait(&mutex);
18     readcount++;
19     if(readcount == 1)
20     {
21         sem_wait(&wrt);
22     }
23     sem_post(&mutex);
24
25     printf("count is: %d and readcount is %d \n", count, readcount);
26
27     sem_wait(&mutex);
28     readcount--;
29     if(readcount == 0)
30     {
31         sem_post(&wrt);
32     }
33 }
```

```
180905402@prg22:~/5thSemLabs/os/lab8$ ./p2
writer count is: 2
count is: 2 and readcount is 1
count is: 2 and readcount is 1
writer count is: 3
count is: 3 and readcount is 1
writer count is: 4
count is: 4 and readcount is 1
writer count is: 5
count is: 5 and readcount is 1
writer count is: 6
count is: 6 and readcount is 1
writer count is: 7
count is: 7 and readcount is 1
writer count is: 8
count is: 8 and readcount is 1
writer count is: 9
count is: 9 and readcount is 1
writer count is: 10
count is: 10 and readcount is 1
writer count is: 11
count is: 11 and readcount is 1
writer count is: 12
```

q3)

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

```
int shared;
sem_t sem1,sem2;
int value1, value2;
```

```
void* func1()
{
    sem_wait(&sem1);
    sem_getvalue(&sem1, &value1);
    printf("In function 1 with value of sem1: %d \n", value1);
    sem_wait(&sem2);
}
```

```
void* func2()
{
    sem_wait(&sem2);
    sem_getvalue(&sem2, &value2);
    printf("In function 1 with value of sem2: %d \n", value2);
    sem_wait(&sem1);
}
```

```
void main()
{
    pthread_t tid1,tid2;
```

```

sem_init(&sem1,0,1);
sem_init(&sem2,0,1);
pthread_create(&tid1,NULL,func1,NULL);
pthread_create(&tid2,NULL,func2,NULL);
pthread_join(tid1,NULL);
pthread_join(tid2,NULL);
}

```

The screenshot shows a Sublime Text editor window with a C program. The program uses two semaphores, sem1 and sem2, to coordinate two threads, func1 and func2. The main function creates these threads and joins them. The terminal window shows the compilation of p3.c to p3 and its execution. The output shows that both functions wait on their respective semaphores and then print their values, which are 0.

```

1 #include<stdio.h>
2 #include<pthread.h>
3 #include<semaphore.h>
4 #include<stdlib.h>
5 #include<unistd.h>
6
7 int shared;
8 sem_t sem1,sem2;
9 int value1, value2;
10
11 void* func1()
12 {
13     sem_wait(&sem1);
14     sem_getvalue(&sem1, &value1);
15     printf("In function 1 with value of sem1: %d \n", value1);
16     sem_wait(&sem2);
17 }
18
19 void* func2()
20 {
21     sem_wait(&sem2);
22     sem_getvalue(&sem2, &value2);
23     printf("In function 1 with value of sem2: %d \n", value2);
24     sem_wait(&sem1);
25 }
26
27 void main()
28 {
29     pthread_t tid1,tid2;
30     sem_init(&sem1,0,1);
31     sem_init(&sem2,0,1);
32     pthread_create(&tid1,NULL,func1,NULL);
33     pthread_create(&tid2,NULL,func2,NULL);
34 }

```

```

180905402@prg22: ~/5thSemLabs/os/lab8
180905402@prg22:~/5thSemLabs/os/lab8$ gcc p3.c -o p3 -lpthread
180905402@prg22:~/5thSemLabs/os/lab8$ ./p3
In function 1 with value of sem1: 0
In function 1 with value of sem2: 0

```

q4)

```

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

```

```

int shared;
sem_t b, c;
sem_t seat;
int seats = 5;

```

```

void* barber()
{
    while(1)
    {
        printf("idle barber \n");
        sem_wait(&c);
        sem_wait(&seat);
        printf("barber is now working \n");
        seats++;
        sem_post(&seat);
        sem_post(&c);
    }
}

```

```

    }
}

void* customer()
{
    while(1)
    {
        if(seats > 0)
        {
            seats--;
            sem_post(&c);

            sem_wait(&b);
            sem_wait(&seat);
            printf("customer's chance for a haircut has finally come\n");
            sleep(2);
            seats++;
            sem_post(&seat);
            sem_post(&b);
            printf("Customer has finished his haircut\n");
        }
        else
        {
            printf("shop full\n");
            sleep(2);
        }
    }
}

void main()
{
    pthread_t tid1,tid2;
    sem_init(&b,0,1);
    sem_init(&c,0,1);
    sem_init(&seat,0,1);
    pthread_create(&tid1,NULL,barber,NULL);
    pthread_create(&tid2,NULL,customer,NULL);
    pthread_join(tid1,NULL);
    pthread_join(tid2,NULL);
}

```

Activities Terminal Mon 15:21
~/5thSemLabs/os/lab8/p4.c - Sublime Text (UNREGISTERED)

File Edit Selection Find View Goto Tools Project Preferences Help

p1.c x p2.c x p3.c x p4

```
49 }
50 }
51
52 void main()
53 {
54
55     pthread_t tid1,tid2;
56     sem_init(&b,0,1);
57     sem_init(&c,0,1);
58     sem_init(&seat,0,1);
59     pthread_create(&tid1,NULL,barber,NULL);
60     pthread_create(&tid2,NULL,customer,NULL);
61     pthread_join(tid1,NULL);
62     pthread_join(tid2,NULL);
63 }
```

180905402@prg22: ~/5thSemLabs/os/lab8\$./p4

File Edit View Search Terminal Help

```
180905402@prg22:~/5thSemLabs/os/lab8$ ./p4
idle barber
customer's chance for a haircut has finally come
Customer has finished his haircut
barber is now working
idle barber
barber is now working
idle barber
barber is now working
idle barber
barber is now working
idle barber
customer's chance for a haircut has finally come
Customer has finished his haircut
customer's chance for a haircut has finally come
Customer has finished his haircut
barber is now working
idle barber
customer's chance for a haircut has finally come
Customer has finished his haircut
customer's chance for a haircut has finally come
Customer has finished his haircut
customer's chance for a haircut has finally come
```

Line 63, Column 2 P main Tab Size: 4 C

S