

OS Assignment

MD ASIBUR RAHMAN AKASH

Roll: 19 Reg:11800427

Section: K18PT Group: 1

Q1. Write a program to sync 5 readers and 3 writers such that multiple readers should be allowed to read at same time but reader and writer cannot simultaneously enter into critical section. Program should be implemented with the help of thread and use semaphore for synchronization.

Q2. Write program to create two threads and both the thread are sharing a common variable whose value is initialized by 2. First thread should increment the shared value by 1 and the other thread should decrement by 1. In order to avoid context switching and to attain consistency use Mutex Lock. Create a scenario such that after execution, the value of the shared variable should be consistent.

```
#include<pthread.h>
```

```
#include<semaphore.h>
```

```
#include<stdio.h>

pthread_mutex_t mutex;

int num =2;

void * increment(){
    int x;
    pthread_mutex_lock(&mutex);
    x=num;
    x++;
    num=x;
    pthread_mutex_unlock(&mutex);
}

void * decrement(){
    int y;
    pthread_mutex_lock(&mutex);
    y = num;
    y--;
    num =y;
    pthread_mutex_unlock(&mutex);
}

void main(){
    pthread_t t1,t2;
    pthread_create(&t1,NULL,*increment,NULL);
```

```

pthread_create(&t2,NULL,*decrement,NULL);

printf(" initial value before %d\n",num);

pthread_join(t1,NULL);

pthread_join(t2,NULL);

printf("final value after %d\n",num);

```

The screenshot shows the OnlineGDB interface with a C program that demonstrates thread synchronization using pthreads. The program includes `pthread.h` and `semaphore.h`. It defines a mutex and a semaphore, both initialized to 1. Two threads are created: `increment` and `decrement`. The `increment` thread locks the semaphore, increments a counter, and unlocks the semaphore. The `decrement` thread locks the semaphore, decrements the counter, and unlocks the semaphore. The program prints the initial value (2) and the final value (2) after both threads complete. The console output shows 'initial value before 2' and 'final value after 2'. The program finishes with exit code 20.

Q3. Write a program to create three threads and three binary semaphore S1 ,S2 and S3 that are initialized with value 1. Threads should acquire semaphore in such a manner that deadlock is achieved.

```
#include<pthread.h>
#include<stdio.h>
#include<semaphore.h>
int num=1;
int x;
void *thread1();
void *thread2();
void *thread3();
sem_t s1,s2,s3;
void main(){

    sem_init(&s1,0,1);
    sem_init(&s2,0,1);
    sem_init(&s3,0,1);
    pthread_t p1,p2,p3;
    pthread_create(&p1,NULL,&thread1,NULL);
    pthread_create(&p2,NULL,&thread2,NULL);
    pthread_create(&p3,NULL,&thread3,NULL);
    printf("before the value is %d \n",num);
    pthread_join(p1,NULL);
    pthread_join(p2,NULL);
    pthread_join(p3,NULL);
```

```
        printf("after the value is %d\n",num);
    }
void *thread1(){
    sem_wait(&s1);
    sem_wait(&s2);
    x=num;
    x++;
    num=x;
    sem_post(&s2);
}
void *thread2(){
    sem_wait(&s2);
    sem_wait(&s3);
    x=num;
    num=x;
    sem_post(&s3);
}
void *thread3(){
    sem_wait(&s3);
    sem_wait(&s1);
    x=num;
    x--;
```

```

num=x;

sem_post(&s1);

}

```

The screenshot shows the OnlineGDB web interface. On the left is a sidebar with navigation links like 'Welcome, Akash', 'Create New Project', 'My Projects', 'Learn Programming', 'Programming Questions', 'Jobs', and 'Logout'. The main area displays a C program named 'main.c' with the following code:

```

1 #include<pthread.h>
2 #include<stdio.h>
3 #include<semaphore.h>
4 int num=1;
5 int x;
6 void *thread1();
7 void *thread2();
8 void *thread3();
9 sem_t s1,s2,s3;
10 void main(){
11
12     sem_init(&s1,0,1);
13     sem_init(&s2,0,1);
14     sem_init(&s3,0,1);
15     pthread_t p1,p2,p3;
16     pthread_create(&p1,NULL,&thread1,NULL);
17     pthread_create(&p2,NULL,&thread2,NULL);
18     pthread_create(&p3,NULL,&thread3,NULL);
19     printf("before the value is %d \n",num);
20     pthread_join(p1,NULL);
21     pthread_join(p2,NULL);
22     pthread_join(p3,NULL);
23     printf("after the value is %d\n",num);
24 }
25 void *thread1(){
26     sem_wait(&s1);
27     sem_wait(&s2);

```

Below the code editor, the 'Input' pane shows the output: 'before the value is 1' followed by 'deadlock' and a series of dots, indicating the program has hung.

Q4. Write a program to implement producer consumer problem such that Maximum number of items produced by a producer are five. Producer should not produce any item if the buffer is full and should say" BUFFER IS ALREADY FULL" and the consumer should not consume if the buffer is empty. Problem should be implemented with the help of thread, semaphore and mutex lock.

```
#include<stdio.h>
```

```
#include<stdlib.h>
```

```
int mutex=1,full=0,empty=3,x=0;
```

```
int main()
```

```
{
```

```
    int n;
```

```
    void producer();
```

```
    void consumer();
```

```
    int wait(int);
```

```
    int signal(int);
```

```
    printf("\n1.Producer\n2.Consumer\n3.Exit");
```

```
    while(1)
```

```
    {
```

```
        printf("\nEnter your choice:");
```

```
        scanf("%d",&n);
```

```
        switch(n)
```

```
        {
```

```
            case 1:    if((mutex==1)&&(empty!=0))
```

```

        producer();
    else
        printf("Buffer is full!!");
    break;
case 2: if((mutex==1)&&(full!=0))
        consumer();
    else
        printf("Buffer is empty!!");
    break;
case 3:
    exit(0);
    break;
}
}

return 0;
}

int wait(int s)
{
    return (--s);
}

```



```
int signal(int s)
```

```
{
```

```
    return(++s);
```

```
}
```

```
void producer()
```

```
{
```

```
    mutex=wait(mutex);
```

```
    full=signal(full);
```

```
    empty=wait(empty);
```

```
    x++;
```

```
    printf("\nProducer produces the item %d",x);
```

```
    mutex=signal(mutex);
```

```
}
```

```
void consumer()
```

```
{
```

```
    mutex=wait(mutex);
```

```
    full=wait(full);
```

```
    empty=signal(empty);
```

```
    printf("\nConsumer consumes item %d",x);
```

```

X--;

mutex=signal(mutex);

}

```

OnlineGDB beta
online compiler and debugger for c/c++

Welcome, Akash

Create New Project

My Projects

Learn Programming

Programming Questions

Jobs new

Logout

Segment Send data to any tool without having to implement a new API every time. ads served ethically

About • FAQ • Blog • Terms of Use • Contact Us • GDB Tutorial • Credits • Privacy

main.c

```

1 #include<stdio.h>
2 #include<stdlib.h>
3
4 int mutex=1,full=0,empty=3,x=0;
5
6 int main()
7 {
8     int n;
9     void producer();
10    void consumer();
11    int wait(int);
12    int signal(int);
13    printf("\n1.Producer\n2.Consumer\n3.Exit");
14    while(1)
15    {
16        printf("\nEnter your choice:");
17        scanf("%d",&n);
18        switch(n)
19        {
20            case 1: if((mutex==1)&&(empty!=0))
21                    producer();
22                    else printf("Buffer is full!!");
23                    break;
24            case 2: if((mutex==1)&&(full!=0))
25                    consumer();
26                    else
27

```

Input

```

Producer produces the item 2
Enter your choice:1

Producer produces the item 3
Enter your choice:1
Buffer is full!!
Enter your choice:

```