OS LAB-1BM21CS203

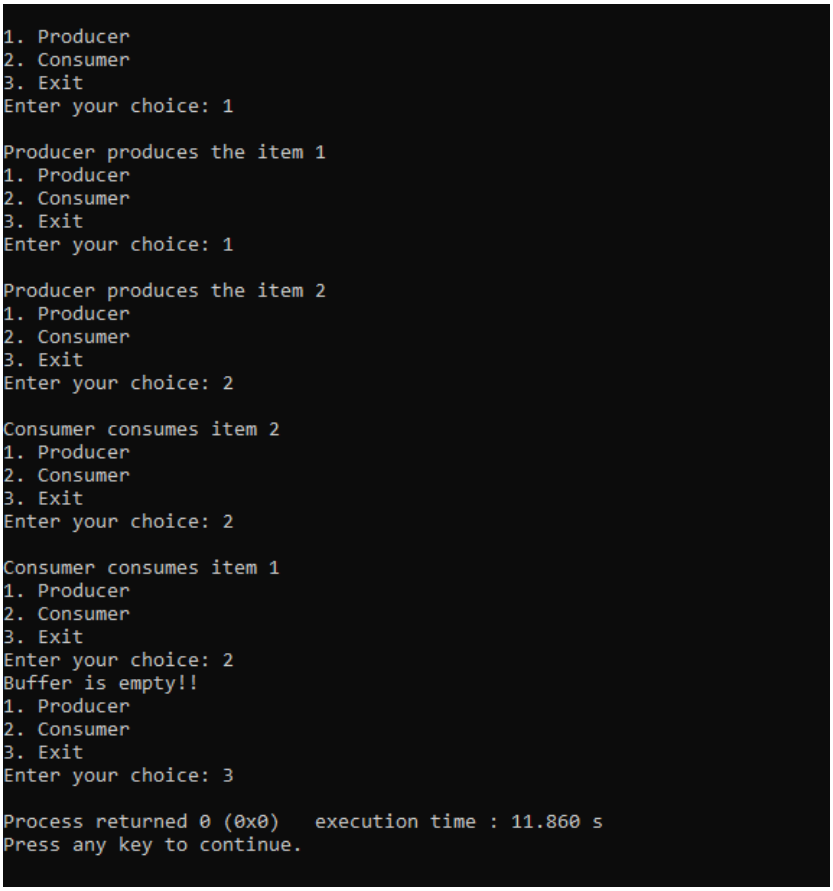
-Shivani Sathyanarayanan

1. Write a C program to simulate producer-consumer problem using semaphores.

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

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

int wait(int);
int signal(int);
void producer();
void consumer();

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);
}

int main() {
   int n;
   while (1) {
```

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

OUTPUT:



2. Write a C program to simulate the concept of Dining-Philosophers problem.

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

#define N 5
#define THINKING 2
#define HUNGRY 1
#define EATING 0
#define LEFT (phnum + 4) % N
#define RIGHT (phnum + 1) % N

int state[N];
int phil[N] = { 0, 1, 2, 3, 4 };
```

```
sem_t mutex;
sem_t S[N];

void test(int phnum)
{
    if (state[phnum] == HUNGRY
            && state[LEFT] != EATING
            && state[RIGHT] != EATING) {

            state[phnum] = EATING;

            sleep(2);

            printf("Philosopher %d takes fork %d and %d\n",
                                    phnum + 1, LEFT + 1, phnum + 1);

            printf("Philosopher %d is Eating\n", phnum + 1);


            sem_post(&S[phnum]);
    }
}


void take_fork(int phnum)
{

    sem_wait(&mutex);

    state[phnum] = HUNGRY;
```

```
    printf("Philosopher %d is Hungry\n", phnum + 1);

    test(phnum);

    sem_post(&mutex);

    sem_wait(&S[phnum]);

    sleep(1);
}

void put_fork(int phnum)
{

    sem_wait(&mutex);

    state[phnum] = THINKING;

    printf("Philosopher %d putting fork %d and %d down\n",
            phnum + 1, LEFT + 1, phnum + 1);
    printf("Philosopher %d is thinking\n", phnum + 1);

    test(LEFT);
    test(RIGHT);
```

```
    sem_post(&mutex);
}

void* philosopher(void* num)
{

    while (1) {

            int* i = num;

            sleep(1);

            take_fork(*i);

            sleep(0);

            put_fork(*i);
    }
}

int main()
{

    int i;
    pthread_t thread_id[N];

    sem_init(&mutex, 0, 1);

    for (i = 0; i < N; i++)
```

```
            sem_init(&S[i], 0, 0);

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

            pthread_create(&thread_id[i], NULL,
                                    philosopher, &phil[i]);

            printf("Philosopher %d is thinking\n", i + 1);
    }

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

            pthread_join(thread_id[i], NULL);
}
```

Output:

```
Philosopher 1 is thinking
Philosopher 2 is thinking
Philosopher 3 is thinking
Philosopher 4 is thinking
Philosopher 5 is thinking
Philosopher 1 is Hungry
Philosopher 3 is Hungry
Philosopher 5 is Hungry
Philosopher 4 is Hungry
Philosopher 4 takes fork 3 and 4
Philosopher 4 is Eating
Philosopher 2 is Hungry
Philosopher 2 takes fork 1 and 2
Philosopher 2 is Eating
Philosopher 4 putting fork 3 and 4 down
Philosopher 4 is thinking
Philosopher 5 takes fork 4 and 5
Philosopher 5 is Eating
Philosopher 2 putting fork 1 and 2 down
Philosopher 2 is thinking
Philosopher 3 takes fork 2 and 3
Philosopher 3 is Eating
Philosopher 5 putting fork 4 and 5 down
Philosopher 5 is thinking
Philosopher 1 takes fork 5 and 1
Philosopher 1 is Eating
Philosopher 4 is Hungry
Philosopher 2 is Hungry
Philosopher 3 putting fork 2 and 3 down
Philosopher 3 is thinking
Philosopher 4 takes fork 3 and 4
Philosopher 4 is Eating
Philosopher 5 is Hungry
Philosopher 1 putting fork 5 and 1 down
Philosopher 1 is thinking
Philosopher 2 takes fork 1 and 2
Philosopher 2 is Eating
Philosopher 3 is Hungry
Philosopher 4 putting fork 3 and 4 down
```