OS_LAB_09_Assignment

CE_054

Aim :- Study of semaphores and implementation of producer-consumer problem.

Theory :-

1. sem_init()

=> To initialize an unnamed semaphore

Synopsis:

#include <semaphore.h>

int sem_init(sem_t * sem, int pshared, unsigned value);

Arguments:

Sem => A pointer to the sem_t object for the semaphore that you want to initialize.

Pshared => Nonzero if you want the semaphore to be shared between processes via shared memory.

Value => The initial value of the semaphore. A positive value (i.e. greater than zero) indicates an unlocked semaphore, and a value of 0 (zero) indicates a locked semaphore. This value must not exceed SEM_VALUE_MAX.

Description:

The sem_init() function initializes the unnamed semaphore referred to by the sem argument. The initial counter value of this semaphore is specified by the value argument.

You should allocate synchronization objects only in normal memory mappings. On certain processors (e.g. some PPC ones), atomic operations such as calls to pthread_mutex_lock() will cause a fault if the control structure is allocated in uncached memory.

You can use the initialized semaphore in subsequent calls to sem_wait(), sem_trywait(), sem_post(), and sem_destroy(). An initialized semaphore is valid until it's destroyed by the sem_destroy() function, or until the memory where the semaphore resides is released.

If the pshared argument is nonzero, then the semaphore can be shared between processes via shared memory. Any process can then use sem with the sem_wait(), sem_trywait(), sem_post() and sem_destroy() functions.

Return value:

0 => Success. The semaphore referred to by sem is initialized.

-1 => An error occurred (errno is set).

2. sem_wait()

=>Wait on a named or unnamed semaphore

Synopsis:

#include <semaphore.h>

int sem_wait(sem_t * sem);

Arguments:

Sem => A pointer to the sem_t object for the semaphore that you want to wait on.

Description:

The sem_wait() function decrements the semaphore referred to by the sem argument. If the semaphore value is not greater than zero, then the calling process blocks until it can decrement the counter, or the call is interrupted by signal.

Some process should eventually call sem_post() to increment the semaphore.

Return Value:

0 => The semaphore was successfully decremented.

-1 => The state of the semaphore is unchanged (errno is set).

3. sem_post()

=> Increment a named or unnamed semaphore

Synopsis:

#include <semaphore.h>

int sem_post(sem_t * sem);

Arguments:

Sem => A pointer to the sem_t object for the semaphore whose value you want to increment.

Description:

The sem_post() function increments the semaphore referenced by the sem argument. If any processes are currently blocked waiting for the semaphore, then one of these processes will return successfully from its call to sem_wait.

The process to be unblocked is determined in accordance with the scheduling policies in effect for the blocked processes. The highest priority waiting process is unblocked, and if there is more than one highest priority process blocked waiting for the semaphore, then the highest priority process that has been waiting the longest is unblocked.

The sem_post() function is reentrant with respect to signals, and can be called from a signal handler.

Return Value:

0 => Success.

-1 => An error occurred (errno is set).

Program :-

1. Write a program to implement solution of bounded buffer producer consumer problem using semaphores.

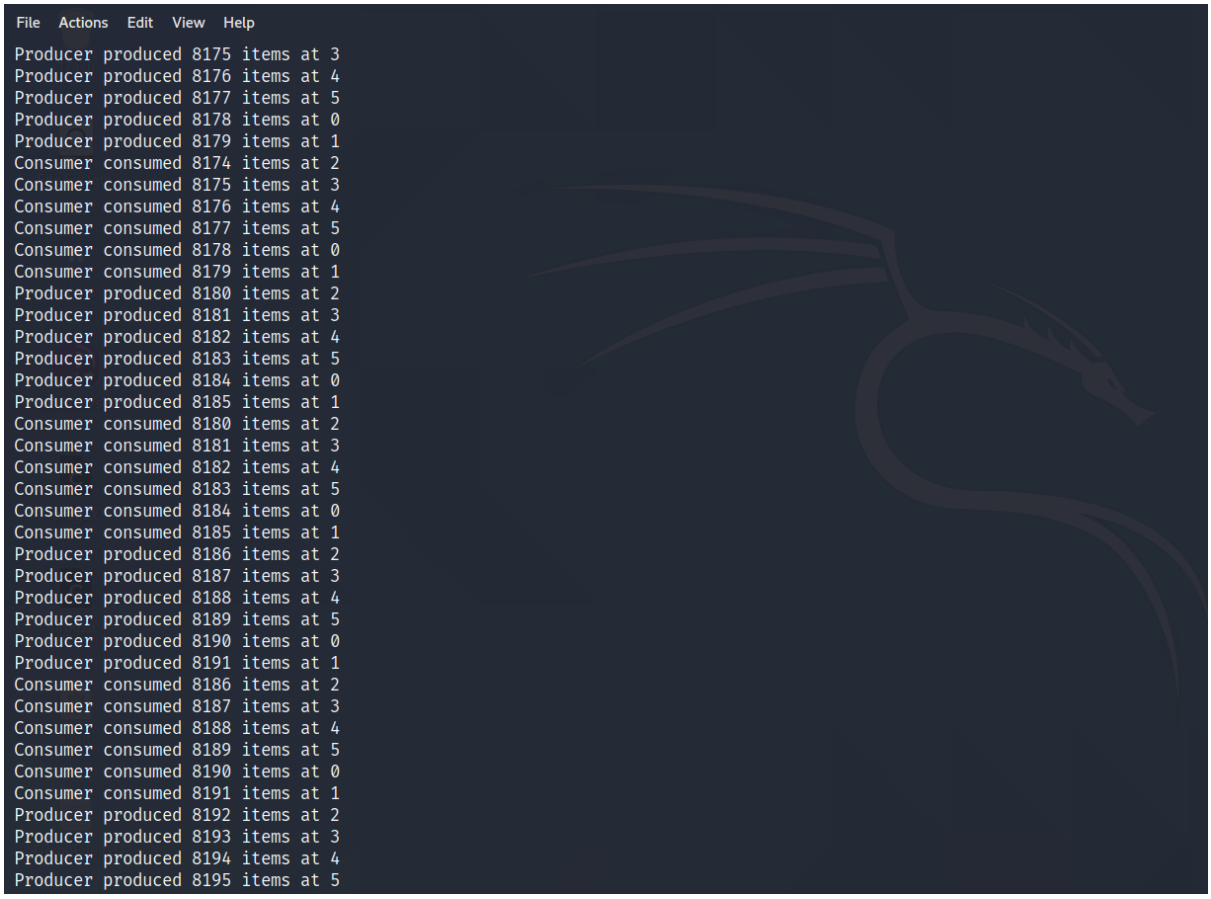
Code :-

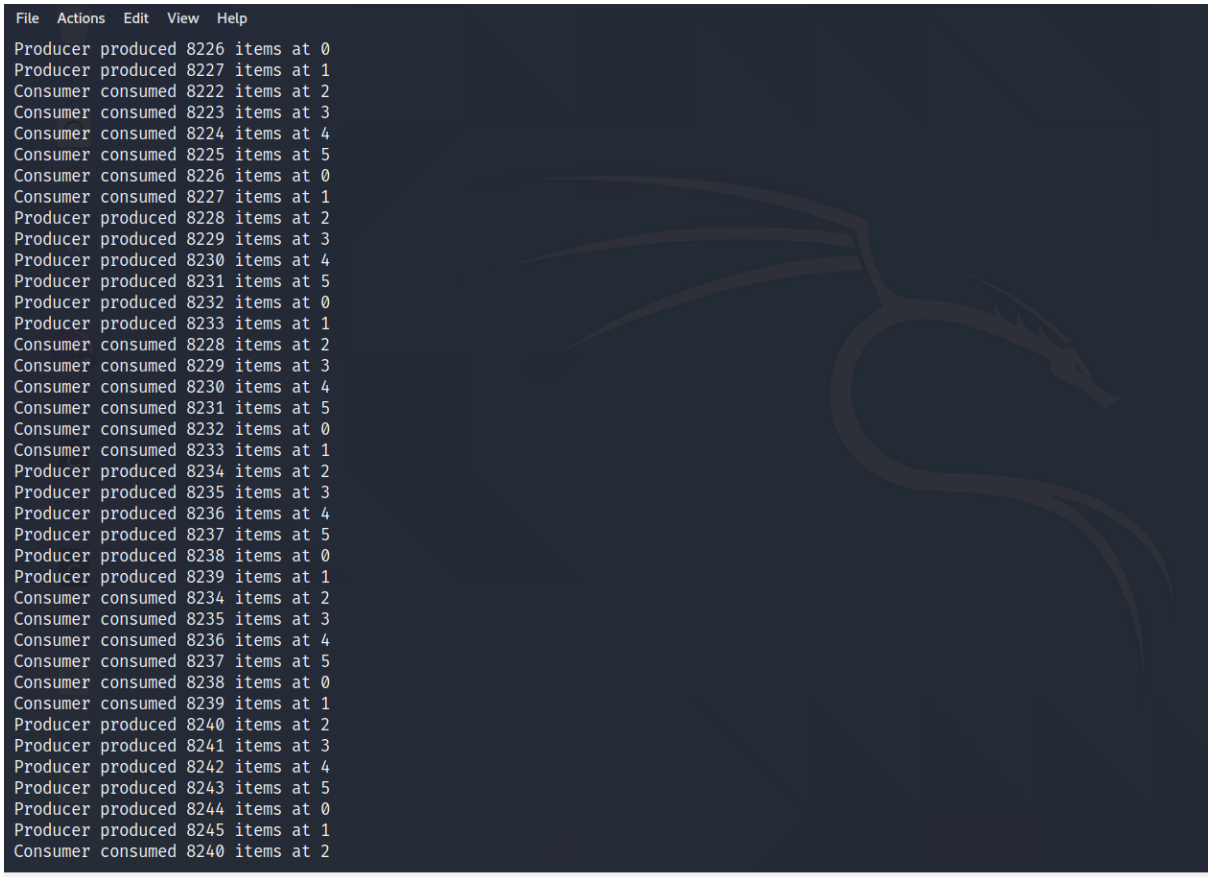
```c
// Author : Dhruv B Kakadiya
#include<stdio.h>
#include<pthread.h>
#include<semaphore.h>
#define PC_BUFFER 6

sem_t mut_sema, empty_sema, full_sema;
int pcbuff[PC_BUFFER];
void* producer(void *args);
void* consumer(void *args);
```

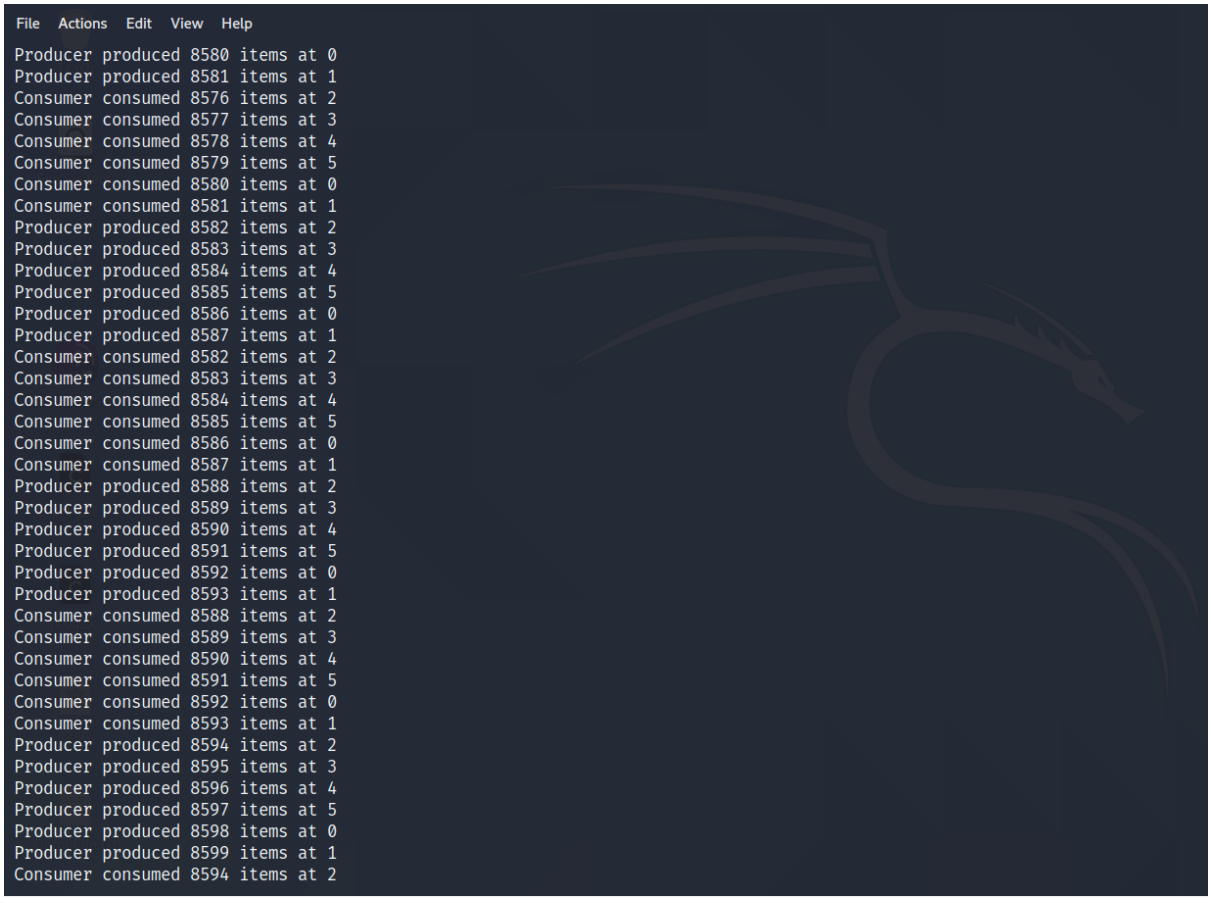
```c
int main()
{
    pthread_t pth, cth;
    if (sem_init(&mut_sema, 0, PC_BUFFER) < 0)
    {
        printf("---Error in mut_sema---");
    }
    if (sem_init(&empty_sema, 0, PC_BUFFER) < 0)
    {
        printf("---Error in empty_Sema---");
    }
    if (sem_init(&full_sema, 0, 0) < 0)
    {
        printf("--Error in full_sema");
    }
    pthread_create(&pth, NULL, producer, NULL);
    pthread_create(&cth, NULL, consumer, NULL);
    pthread_join(pth, NULL);
    pthread_join(cth, NULL);
}
void* producer(void* args)
{
    int i = 0;
    int j = 0;
    while(1)
    {
        sem_wait(&empty_sema);
        sem_wait(&mut_sema);
        printf("Producer produced %d items at %d\n", i, j);
        i++;
        j = i % PC_BUFFER;
        sem_post(&mut_sema);
        sem_post(&full_sema);
    }
}

void* consumer(void* args)
{
    int i = 0;
    int j = 0;
    while(1)
    {
        sem_wait(&full_sema);
        sem_wait(&mut_sema);
        printf("Consumer consumed %d items at %d\n", i, j);
        i++;
        j = i % PC_BUFFER;
```

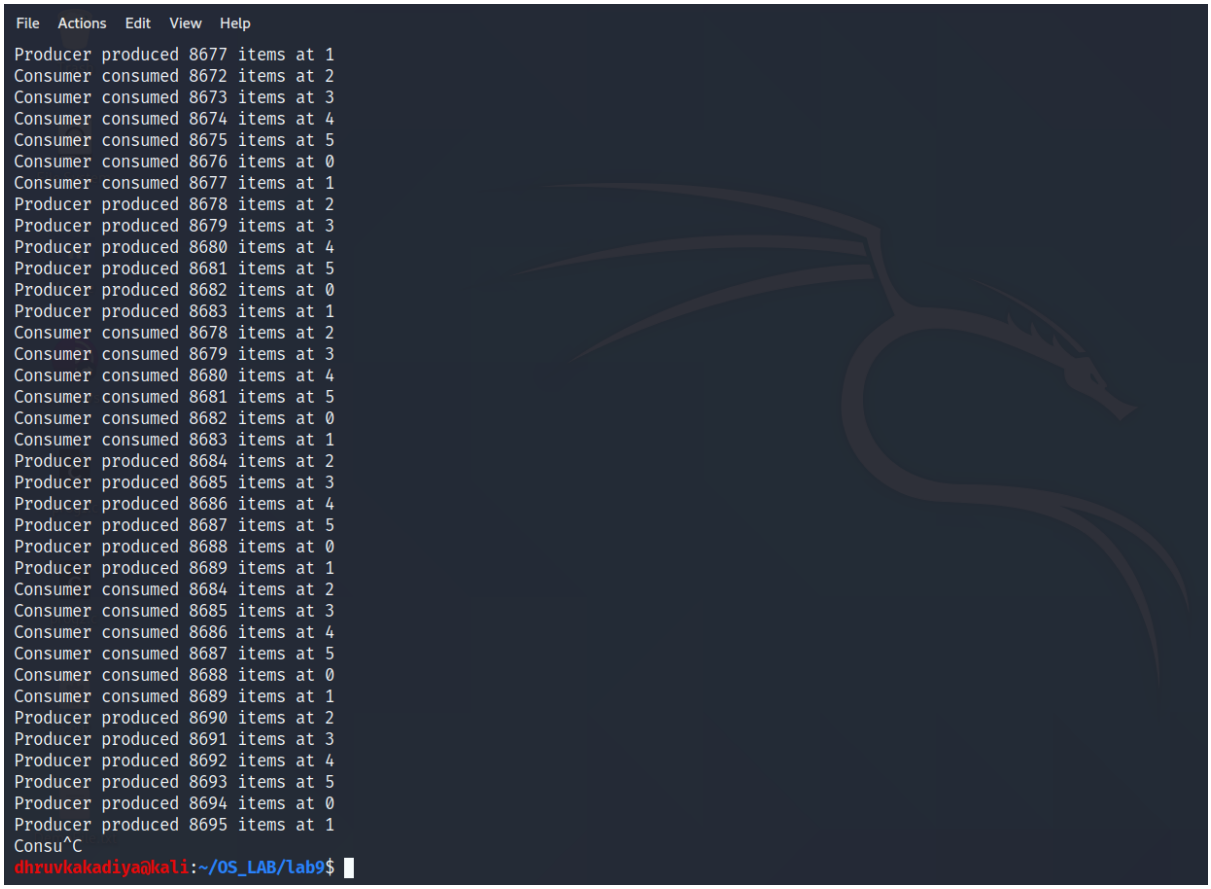
```
        sem_post(&mut_sema);
        sem_post(&empty_sema);
    }
}
```

Output :-




File Actions Edit View Help
Producer produced 8226 items at 0
Producer produced 8227 items at 1
Consumer consumed 8222 items at 2
Consumer consumed 8223 items at 3
Consumer consumed 8224 items at 4
Consumer consumed 8225 items at 5
Consumer consumed 8226 items at 0
Consumer consumed 8227 items at 1
Producer produced 8228 items at 2
Producer produced 8229 items at 3
Producer produced 8230 items at 4
Producer produced 8231 items at 5
Producer produced 8232 items at 0
Producer produced 8233 items at 1
Consumer consumed 8228 items at 2
Consumer consumed 8229 items at 3
Consumer consumed 8230 items at 4
Consumer consumed 8231 items at 5
Consumer consumed 8232 items at 0
Consumer consumed 8233 items at 1
Producer produced 8234 items at 2
Producer produced 8235 items at 3
Producer produced 8236 items at 4
Producer produced 8237 items at 5
Producer produced 8238 items at 0
Producer produced 8239 items at 1
Consumer consumed 8234 items at 2
Consumer consumed 8235 items at 3
Consumer consumed 8236 items at 4
Consumer consumed 8237 items at 5
Consumer consumed 8238 items at 0
Consumer consumed 8239 items at 1
Producer produced 8240 items at 2
Producer produced 8241 items at 3
Producer produced 8242 items at 4
Producer produced 8243 items at 5
Producer produced 8244 items at 0
Producer produced 8245 items at 1
Consumer consumed 8240 items at 2


File Actions Edit View Help
Producer produced 8580 items at 0
Producer produced 8581 items at 1
Consumer consumed 8576 items at 2
Consumer consumed 8577 items at 3
Consumer consumed 8578 items at 4
Consumer consumed 8579 items at 5
Consumer consumed 8580 items at 0
Consumer consumed 8581 items at 1
Producer produced 8582 items at 2
Producer produced 8583 items at 3
Producer produced 8584 items at 4
Producer produced 8585 items at 5
Producer produced 8586 items at 0
Producer produced 8587 items at 1
Consumer consumed 8582 items at 2
Consumer consumed 8583 items at 3
Consumer consumed 8584 items at 4
Consumer consumed 8585 items at 5
Consumer consumed 8586 items at 0
Consumer consumed 8587 items at 1
Producer produced 8588 items at 2
Producer produced 8589 items at 3
Producer produced 8590 items at 4
Producer produced 8591 items at 5
Producer produced 8592 items at 0
Producer produced 8593 items at 1
Consumer consumed 8588 items at 2
Consumer consumed 8589 items at 3
Consumer consumed 8590 items at 4
Consumer consumed 8591 items at 5
Consumer consumed 8592 items at 0
Consumer consumed 8593 items at 1
Producer produced 8594 items at 2
Producer produced 8595 items at 3
Producer produced 8596 items at 4
Producer produced 8597 items at 5
Producer produced 8598 items at 0
Producer produced 8599 items at 1
Consumer consumed 8594 items at 2


File Actions Edit View Help
Producer produced 8677 items at 1
Consumer consumed 8672 items at 2
Consumer consumed 8673 items at 3
Consumer consumed 8674 items at 4
Consumer consumed 8675 items at 5
Consumer consumed 8676 items at 0
Consumer consumed 8677 items at 1
Producer produced 8678 items at 2
Producer produced 8679 items at 3
Producer produced 8680 items at 4
Producer produced 8681 items at 5
Producer produced 8682 items at 0
Producer produced 8683 items at 1
Consumer consumed 8678 items at 2
Consumer consumed 8679 items at 3
Consumer consumed 8680 items at 4
Consumer consumed 8681 items at 5
Consumer consumed 8682 items at 0
Consumer consumed 8683 items at 1
Producer produced 8684 items at 2
Producer produced 8685 items at 3
Producer produced 8686 items at 4
Producer produced 8687 items at 5
Producer produced 8688 items at 0
Producer produced 8689 items at 1
Consumer consumed 8684 items at 2
Consumer consumed 8685 items at 3
Consumer consumed 8686 items at 4
Consumer consumed 8687 items at 5
Consumer consumed 8688 items at 0
Consumer consumed 8689 items at 1
Producer produced 8690 items at 2
Producer produced 8691 items at 3
Producer produced 8692 items at 4
Producer produced 8693 items at 5
Producer produced 8694 items at 0
Producer produced 8695 items at 1
Consu^C
dhruvkakadiya@kali:~/OS_LAB/lab9$