Practical-9

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

Explanation:

sem_init function:

#include <semaphore.h>

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

sem_init() initializes the unnamed semaphore at the address pointed to by *sem*. The *value* argument specifies the initial value for the semaphore.

The *pshared* argument indicates whether this semaphore is to be shared between the threads of a process, or between processes.

If *pshared* has the value 0, then the semaphore is shared between the threads of a process, and should be located at some address that is visible to all threads (e.g., a global variable, or a variable allocated dynamically on the heap).

If *pshared* is nonzero, then the semaphore is shared between processes, and should be located in a region of shared memory.

sem_init() returns 0 on success; on error, -1 is returned, and errno is set to indicate the error.

E.g.

```
sem_t mutex;
sem_init(&mutex,0,1);
```

sem_wait function:

#include <semaphore.h>

int sem_wait(sem_t *_sem_);

sem_wait() decrements (locks) the semaphore pointed to by *sem*. If the semaphore's value is greater than zero, then the decrement proceeds, and the function returns, immediately. If the semaphore currently has the value zero, then the call blocks until either it becomes possible to perform the decrement.

This function returns 0 on success; on error, the value of the semaphore is left unchanged, -1 is returned, and *errno* is set to indicate the error.

E.g.

```
sem_t mutex;
sem_wait(&mutex);
```

sem_post function:

#include <semaphore.h>

int sem_post(sem_t *_sem_);

sem_post() increments (unlocks) the semaphore pointed to by *sem*. If the semaphore's value consequently becomes greater than zero, then another process or thread blocked in a sem_wait call will be woken up and proceed to lock the semaphore.

sem_post() returns 0 on success; on error, the value of the semaphore is left unchanged, -1 is returned, and errno is set to indicate the error.

E.g.

```
sem_t mutex;
sem_post(&mutex);
```

Program:

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