Expr 8: Semaphore

Semaphore code:

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

#define BUFFER_SIZE 5
#define PRODUCE_COUNT 10

int buffer[BUFFER_SIZE];
int in = 0, out = 0;

sem_t empty, full, mutex;

void *producer(void *arg) {
    for (int i = 0; i < PRODUCE_COUNT; i++) {
        int item = i + 1;

        sem_wait(&empty);
        sem_wait(&mutex);

        buffer[in] = item;
        printf("Producer produced: %d\n", item);
        in = (in + 1) % BUFFER_SIZE;

        sem_post(&mutex);
        sem_post(&full);

        sleep(1);
    }
    pthread_exit(NULL);
}

void *consumer(void *arg) {
    for (int i = 0; i < PRODUCE_COUNT; i++) {
        sem_wait(&full);
        sem_wait(&mutex);

        int item = buffer[out];
        printf("Consumer consumed: %d\n", item);
        out = (out + 1) % BUFFER_SIZE;

        sem_post(&mutex);
        sem_post(&empty);

        sleep(1);
```

```
    }
    pthread_exit(NULL);
}

int main() {
    pthread_t prodThread, consThread;

    sem_init(&empty, 0, BUFFER_SIZE);
    sem_init(&full, 0, 0);
    sem_init(&mutex, 0, 1);

    pthread_create(&prodThread, NULL, producer, NULL);
    pthread_create(&consThread, NULL, consumer, NULL);

    pthread_join(prodThread, NULL);
    pthread_join(consThread, NULL);

    sem_destroy(&empty);
    sem_destroy(&full);
    sem_destroy(&mutex);

    return 0;
}
```

Output:

```
kfl02@fedora:~/exp8$ ./pc
Producer produced: 1
Consumer consumed: 1
Producer produced: 2
Consumer consumed: 2
Producer produced: 3
Consumer consumed: 3
Producer produced: 4
Consumer consumed: 4
Producer produced: 5
Consumer consumed: 5
Producer produced: 6
Consumer consumed: 6
Producer produced: 7
Consumer consumed: 7
Producer produced: 8
Consumer consumed: 8
Producer produced: 9
Consumer consumed: 9
Producer produced: 10
Consumer consumed: 10
```

Result:

Thus the Semaphore Code is implemented in fedora using the C language