CS 361 – Lab 11

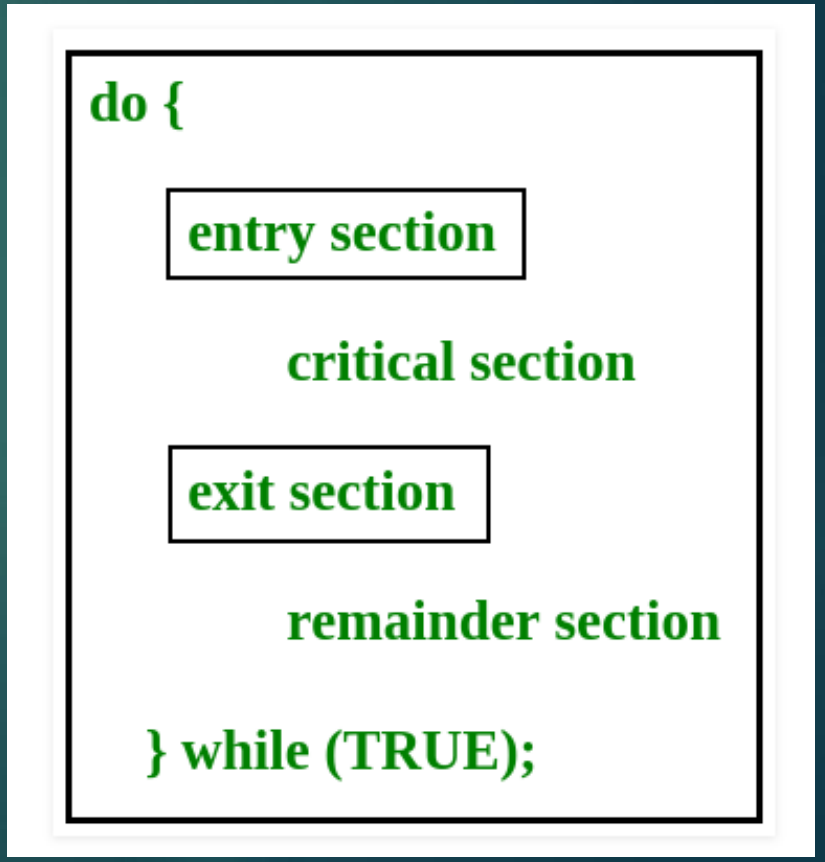
Producer-Consumer Problem

MONDAY, APRIL 22ND 2019.

Semaphores

- Critical region is a code segment that can only be accessed by one process at a time; contains shared variables that need to be synchronized for consistency
- Solution: mutual exclusion
  - Semaphores (Ch. 12.5.2 in book)

- Semaphore is a non-negative integer variable; accessed through wait()(to decrement) and signal()(to increment) operations
  - Binary Semaphore (aka mutex)
  - Counting Semaphore



Producer-Consumer Problem

- The producer consumer problem is a synchronization problem. There is a fixed size buffer and the producer produces items and enters them into the buffer. The consumer removes the items from the buffer and consumes them.
- A producer should not produce items into the buffer when the consumer is consuming an item from the buffer and vice versa. So the buffer should only be accessed by the producer or consumer at a time.
- The producer consumer problem can be resolved using semaphores.

Producer Process Pseudocode

```
do {
        .
        . PRODUCE ITEM
        .
        wait(empty);
        wait(mutex);
        .
        . PUT ITEM IN BUFFER
        .
        signal(mutex);
        signal(full);

} while(1);
```

Consumer Process Pseudocode

```
do {

        wait(full);
        wait(mutex);
        .          .
        .   REMOVE ITEM FROM BUFFER
        .
        signal(mutex);
        signal(empty);
        .
        . CONSUME ITEM
        .
} while(1);
```

- **int pthread_create(pthread_t *thread, const pthread_attr_t *attr,**

*void *(*start_routine)(void*), void *arg);*

  - used to create a new thread
- **int pthread_join(pthread_t thread, void **value_ptr);**
  - suspends execution of the calling thread until the target thread terminates
- **int pthread_cond_wait(pthread_cond_t *cond, pthread_mutex_t *mutex);**
  - used to block on a condition variable
- **int pthread_cond_signal(pthread_cond_t *cond);**
  - used to unblock threads blocked on a condition variable

- **int pthread_mutex_lock(pthread_mutex_t *mutex);**
- **int pthread_mutex_unlock(pthread_mutex_t *mutex);**
- The mutex object referenced by mutex shall be locked by calling pthread_mutex_lock(). If the mutex is already locked, the calling thread shall block until the mutex becomes available.
- The pthread_mutex_unlock() function shall release the mutex object referenced by mutex.
- **void pthread_exit(void *value_ptr);**
  - thread termination

Task

- Write C code for the consumer() thread in the given producer-consumer program (prodCons.c in github). The program creates an N number of producer and consumer threads. The job of the producer will be to generate a random number and place it in a bound-buffer. The role of the consumer will be to remove items from the bound-buffer and print them to the screen. .