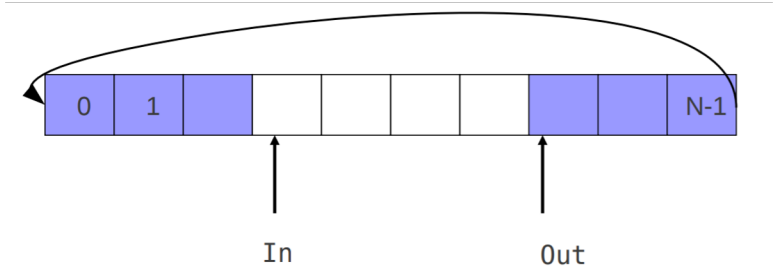
ICS53 - winter 2022

The Producer-Consumer Problem & POSIX Threads Condition Variables

Producer Consumer Problem

- Multiple producer-threads.
- Multiple consumer-threads.
- One bounded buffer with N entries.
- All threads modify the same buffer.
- Requirements:
  - No production when all N entries are full.
  - No consumption when no entry is full.
  - Only one thread should modify the buffer at any time.



Pthread Mutex

- A mutex is a MUTual EXclusion device, and is useful for protecting shared data structures from concurrent modifications, and implementing critical sections and monitors.
- A mutex has two possible states:
  - **unlocked (not owned by any thread)**
  - **locked (owned by one thread)**
    - A mutex can never be owned by two different threads simultaneously.
    - A thread attempting to lock a mutex that is already locked by another thread is suspended until the owning thread unlocks the mutex first.

pthread_mutex_lock and pthread_mutex_unlock

```
#include <pthread.h>

int pthread_mutex_lock( pthread_mutex_t* mutex );
 int pthread_mutex_unlock( pthread_mutex_t* mutex );
```

- pthread_mutex_lock locks the given mutex.
  - If the mutex is currently unlocked, it becomes locked and owned by the calling thread, and pthread_mutex_lock returns immediately.
  - If the mutex is already locked by another thread, pthread_mutex_lock suspends the calling thread until the mutex is unlocked.
- pthread_mutex_unlock unlocks the given mutex.
  - The mutex is assumed to be locked and owned by the calling thread on entrance to pthread_mutex_unlock.

pthread_cond_wait

#include <pthread.h>

int pthread_cond_wait(pthread_cond_t *_cond_, pthread_mutex_t *_mutex_);

- *pthread_cond_wait()* atomically releases *mutex* and causes the calling thread to block on the condition variable *cond*;
  - atomically here means "atomically with respect to access by another thread to the mutex and then the condition variable".

pthread_cond_signal and pthread_cond_brodcast

#include <pthread.h>

int pthread_cond_signal(pthread_cond_t *cond);
int pthread_cond_broadcast(pthread_cond_t *cond);

- *pthread_cond_signal()* unblocks at least one of the threads that are blocked on the specified condition variable *cond* (if any threads are blocked on *cond*).
- *pthread_cond_broadcast()* unblocks all threads currently blocked on the specified condition variable *cond*.

Basic Producer Consumer Problem Using Condition Variables Sudocode

Producer:

```
while (true) {

  pthread_mutex_lock (&mutex);

  while (/*buffer is full*/){

     pthread_cond_wait(&notfull,&mutex);
}

 /*produces an item*/

  pthread_cond_signal(&notempty);

  pthread_mutex_unlock(&mutex);

}
```

Consumer:

```
while (true) {

  pthread_mutex_lock (&mutex);

  while (/*buffer is empty*/){

     pthread_cond_wait(&notempty,&mutex);
}

 /*consumes an item*/

  pthread_cond_signal(&notfull);

  pthread_mutex_unlock(&mutex);

}
```