(Good) Producer consumer **using a lock**

```
lock := init()
```

```
void producer () {
  while(1){
     item := produce()
     acquire(lock)
     while(!emtpy(buffer)){
          release(lock)
          yield();
          acquire(lock)
     }
     write(buffer, item)
     release(lock)
  }
}
```

```
void consumer () {
  while(1){
     acquire(lock)
     while(emtpy(buffer)){
          release(lock)
          yield();
          acquire(lock)
     }
     item := read(buffer)
     release(lock)
     consume(item)
  }
}
```

Another Synchronization Construct
Condition Variable

A condition variable supports three operations

- **`cond_wait(cond, lock)`**
  unlock the lock and sleep until `cond` is signaled
  then re-acquire `lock` before resuming execution
- **`cond_signal(cond)`**
  signal the condition `cond` by waking up the next thread
- **`cond_broadcast(cond)`**
  signal the condition `cond` by waking up all threads