

# 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

# Producers Consumers **using a condition variable**

```
cond_init(not_full)
cond_init(not_empty)
```

```
void producer () {
    while(1){
        item := produce()
        acquire(mutex)
        while(full(buffer))
            cond_wait(not_full, mutex)
        write(buffer, item)
        cond_signal(not_empty)
        release(mutex)
    }
}
```

```
void consumer () {
    while(1){
        acquire(mutex)
        while(empty(buffer))
            cond_wait(not_empty, mutex)
        item := read(buffer)
        cond_signal(not_full)
        release(mutex)
        consume(item)
    }
}
```