Conditional Variables

- Condition variable are a synchronization construct with simple operations:
  - `wait` -- means that the process invoking this operation is suspended until another process/thread invokes `signal`
  - `signal` operation resumes exactly one suspended process/thread. If no process/thread is suspended, then the signal operation has no effect
  - `broadcast` which wakes up all suspended/processes/threads
**Conditional Variables (cont’d)**

**wait (CV*, Lock*)**

If called with lock held: sleep, atomically releasing lock. Atomically reacquire lock before returning.

**signal (CV*, Lock*)**

wake up one waiter, if any

**broadcast (CV*, Lock*)**

wake up all waiters, if any.

---

**Note:** Some implementations may not need locks here.
Conditional Variables

• *Condition variables* allow *explicit* event notifications
  
  • Associated with a mutex to prevent races on event conditions.

```
condition:
acquire (lock);  acquire (lock);
  if (...) wait (CV, lock);  if (...) signal (CV, lock);
release (lock);  release (lock);
```
Posix condition variables

#include<pthread.h>

int pthread_cond_signal (pthread_cond_t*cond);
int pthread_broadcast (pthread_cond_t*cond);
int pthread_cond_wait (pthread_cond_t*cond, //also a timed wait
    pthread_mutex_t*mutex);
int pthread_cond_destroy (pthread_cond_t*cond);
int pthread_cond_init (pthread_cond_t*cond,const
    pthread_condattr_t*attr);

pthread_cond_t cond = PTHREAD_COND_INITIALIZER;
Example #1: Bounded-Buffer

• There is a finite-sized buffer that producer threads want to add items to ... and consumer threads want to remove items from ... repeatedly

• Two kinds of synchronization needed:
  • Me—to protect integrity of the buffer
  • Correctness—producer must block if buffer is full and consumer must block if buffer is empty...
Example

pthread_mutex_t ring_access = PTHREAD_MUTEX_INITIALIZER;

pthread_cond_t buffer_full = PTHREAD_COND_INITIALIZER;

pthread_cond_t buffer_empty = PTHREAD_COND_INITIALIZER;
void buffer_insert (char*item){
    pthread_mutex_lock (&ring_access);
    while (count == RINGSIZE)
        pthread_cond_wait(&buffer_empty,&ring_access);
    Buffer [in]=item; // in, out initialized to 0
    in=(in +1)%RINGSIZE,
    count++,
    pthread_cond_signal (&buffer_full);
    pthread_mutex_unlock (&ring_access);
}
Remove

• Do “remove”
Example #2: Barrier

• Barrier: synchronization construct
• Using one synch construct to implement another

init (int how_many_thread)

checkin ()

• called by all threads
• blocks all threads until last one checks in
Example #3: License Management
Next Time

• More CVs
• Lab #4
• Exam Review