Recitation 11
11-17-2014
Today

- Synchronization
  - Semaphore
  - Condition Variables
  - Producer-Consumer example
**Condition Variables**

- Problem: Make a thread wait until some condition is satisfied \((\text{Say } x == y)\). \((x,y – \text{shared variables among threads})\)

- **Avoid Busy Waiting** \((\text{while } (x != y); ):\)
  - Consumes unnecessary CPU cycles
  - Depending on Scheduling if other threads never get a chance to execute, shared variables may not change and thread may busy wait forever
Condition Variables

• Correct way:
  
  ```
  while(true)
  ```
  
  • **Lock** a mutex
  • Test condition \((x == y)\)
  • If true, **unlock** mutex and **exit** loop
  • If false, **suspend** the thread and **unlock** the mutex

Reference: RR §13.4
Conditional Variables

• To create a conditional variable:

```c
pthread_cond_t condvar = PTHREAD_COND_INITIALIZER;
```

• To wait for a conditional variable:

```c
pthread_cond_wait(&condvar, &mutex);
```
This will unlock the mutex, wait for a signal.

• To signal a thread:

```c
pthread_cond_signal(&condvar);
```
This will send a signal to a thread waiting on condvar. But won’t release the mutex.
Semaphores

Semaphores are another synchronization method, besides Mutex and Cond. Variables.

A semaphore is a special kind of integer: it can be incremented or decremented *atomically*. 
Semaphores

If the value of the semaphore is 0, it cannot be decremented. Threads that try to decrement it when its value is 0 go to sleep

- Increment: int sem_post(sem_t *sem);

- Decrement: int sem_wait(sem_t *sem);
POSIX Semaphores

sem_t access_lock;
sem_init(&access_lock, 0, 2);
// thread code…
sem_wait(&access_lock);
// critical section…
sem_post(&access_lock);
Signature

#include <semaphore.h>

int sem_init(sem_t *sem, int pshared, unsigned int value);

Link with -lrt or -pthread.

Initializes the unnamed semaphore at the address pointed to by sem.

value argument specifies the initial value for the semaphore. 

pshared indicates whether this semaphore is to be shared between the threads of a process, or between processes.

If pshared has the value 0, then the semaphore is shared between the threads of a process.
Sample code

• Code semaphore.c

```c
void* func ( void * ptr )
{
    int x = *((int *) ptr);
    sem_wait(&mutex);       //Down semaphore
    //begin critical section
    counter++;
    //end critical section
    sem_post(&mutex);       // up semaphore
}
```
Producer-Consumer Exercise

Create the producer-consumer relationship with some queue size

- The produced item should be the printout “Item produced!”

- The consumption of an item prints out “Item consumed!”
Producer-Consumer Exercise

Use the provided code Pcsem.c as skeleton
Questions?