Today

• Synchronization
  – Condition Variables
  – Hello World exercise
  – Producer-Consumer example
Condition Variables

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

- Avoid Busy Waiting ($while (x != y);$):
  - 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.
Example

• Fix hello_world_ex.c to print “Hello, World” using two threads
• “Hello” thread has to go first and then signal “world” thread to print
• Hello_world.c shows the completed version
Hello World Synchronization

• We need
  – a shared “flag” variable to synchronize
  – a mutex lock to protect it
  – a conditional variable to wake up the world thread when it’s its turn to print
Hello World Synchronization

• Hello thread
  – Print “Hello”
  – Set flag to 1
  – Signal world thread that it can print

• World thread
  – Check flag
  – If not set, suspend on conditional variable
  – Print “world” when signalled
Hello World Synchronization

• Insert a delay before printing “Hello” to make sure the world thread waits.
• What happens if you don’t initialize the conditional variable?
• Make sure you unlock the mutex before you signal a suspended thread! What happens if you don’t?
Producer-Consumer

- Consumer has to wait while the buffer is empty, producer has to wait while buffer is full.
- `busywait.c` contains a busy wait, `condvar.c` shows how to fix this with conditional variables.
Questions?