Lecture 38: Concurrent & Distributed Programming

CSC 131
Fall, 2006

Key Concepts in Conflicts

- Critical Section
  - *where two processes can access shared resource*
- Race condition
  - *answer depends on order of execution of other events*
- Mutual exclusion
  - *allow only one process in critical section*
- Deadlock
  - *no process can proceed because cannot obtain needed locks*

Dining Philosophers

Monitor in Java

```
public class Buffer {
    private char[] store = new char[MaxBuffSize];
    private int BufferStart, BufferEnd, BufferSize;

   public Buffer(int size) {
      MaxBufferSize = size;
      BufferEnd := -1;
      BufferStart := 0;
      BufferSize := 0;
   }
```

```
   public synchronized void insert(char ch) {
       try {
          while (BufferSize == MaxBuffSize) wait();
          BufferEnd = (BufferEnd + 1) % MaxBuffSize;
          store[BufferEnd] = ch;
          BufferSize++;
          notifyAll();
      catch(InterruptedException e){
         Thread.currentThread().interrupt();
   }}

   public synchronized char delete() {
      try {
         while (BufferSize == 0) wait();
         char ch = store[BufferStart];
         BufferStart = (BufferStart + 1) % MaxBuffSize;
         BufferSize--;
         notifyAll();
         return ch;
      catch(InterruptedException e){
         Thread.currentThread().interrupt();
         return 'i';
      }
   }
```

More Java

- Threads part of standard library:

```
public class SomeThread extends Thread {
   public SomeThread(...) {
           ...;
      start();
   }

   public void run() {...}
}
```

Avoid inheritance w/Runnable

```
class SomeRunnable implements Runnable {
   public SomeRunnable(...) {...}

   public void run() {...}
}

Thread t = new Thread(new SomeRunnable(...));
t.start();
```

Threads

- Not tied to objects
- Primitives:
  - join() -- *t.join() causes current thread to wait for t to terminate*
    - *Useful if spawning off thread to get value needed in future.*
  - sleep(ms) -- *causes wait of at least ms milliseconds*
- Synchronized blocks:
  - `synchronized(someObj){...}`

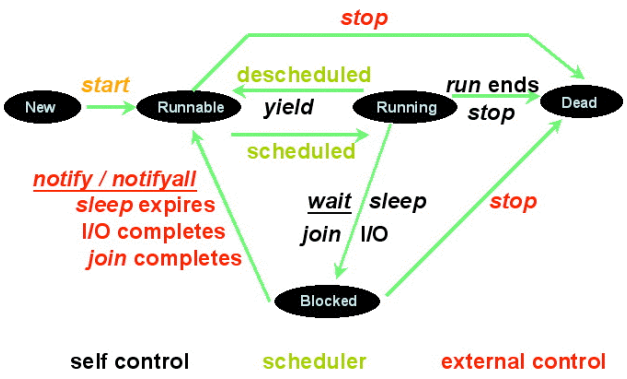
Conditional Waiting

- Every object has a wait set
- `wait()`: release lock & pause until another thread calls notify or notifyAll.
- `notify(), notifyAll()`: wake up waiting threads, which try to grab lock
  - *Can only be used in synchronized code*
  - *Notify wakes up single thread -- arbitrary choice*
  - *NotifyAll wakes up all waiting threads*
  - *Much better than busy-waiting (spin-locks)*

Thread States in Java

- New -- declared, but not yet started
- Runnable -- ready to run
- Running -- currently running
- Blocked -- on I/O, wait on monitor, sleep, join
- Dead -- run has ended

Concurrency in Java



Java 5

- Introduces new higher-level concurrency classes and interfaces
  - Sync -- protocols to acquire and release locks
  - Channels -- protocols to insert and delete objects
  - Executor -- executes Runnable tasks

Message Passing

- Must provide send and receive operations w/ message and receiver/sender.
- Synchronous or asynchronous?
- If asynchronous then use "mailbox" to store
- If synchronous then sender and receiver must "rendezvous" before either can proceed.

Message Passing

- Book covers Actors, objects that can:
  - receive and send messages
  - create new actors
  - specify a replacement behavior
  - asynchronous
  - no guarantee messages delivered in order