Announcements

- Program #2
  - Due next Thursday (3/3/16)
Writers Have Priority

**reader**

repeat
  P(z);
  P(rsem);
  P(x);
  readcount++; if (readcount == 1) then
    P(wsem);
    V(x);
    V(rsem);
  V(z);
  readunit;
  P(x);
  readcount--; if readcount == 0 then
    V(wsem);
    V(y);
  forever

**writer**

repeat
  P(y);
  writecount++;
  if writecount == 1 then
    P(rsem);
    V(y);
    P(wsem);
  writeunit
  V(wsem);
  P(y);
  writecount--; if (writecount == 0) then
    V(rsem);
    V(y);
  forever;
Notes on readers/writers with writers getting priority

Semaphores\( x, y, z, w\text{sem}, r\text{sem} \) are initialized to 1

\[
\begin{align*}
P(z); \\
P(r\text{sem}); \\
P(x); \\
\text{readcount}++; \\
\text{if (readcount==1) then} \\
P(w\text{sem}); \\
V(x); \\
V(r\text{sem}); \\
V(z);
\end{align*}
\]

Readers queue up on semaphore \( z \); this way only a single reader queues on \( r\text{sem} \). When a writer signals \( r\text{sem} \), only a single reader is allowed through.
Sample Synchronization Problem

- **Class Exercise:**
  - *CMSC 412 Midterm #1 (Spring 1998) Q#3*

- **Went over master solution**

- **Variables:**
  - Semaphore mutex = 1
  - Semaphore writer = 0
  - Semaphore reader = 0
  - int nReader = 0
  - int nWriter = 0
  - int wReader = 0
  - int wWriter = 0
Sample Synchronization Problem

- **Class Exercise:**
  - **CMSC 412 Midterm #1 (Spring 1998) Q#3**
  
- Solve a variation of the readers-writers problem, in which multiple writers can write at the same time. Specifically, there are readers and writers. Up to 5 reads at the same time are allowed, but only one write at the same time are allowed. A read and a write at the same time is not allowed. Provide a solution using semaphores with the following properties:
  - no busy waiting.
  - starvation-free (i.e. a continuous stream of readers does not starve writers, and vice versa) is desirable but not compulsory (but you will lose some points).
  - you cannot use process ids and you cannot have a separate semaphore for every process.