Condition Variables and Semaphores

Prof. José F. Martínez
Recap: Readers and Writers

\textit{enter\textsubscript{w}}:
\begin{verbatim}
lock(m);
while (nw > 0 || nr > 0) {
    unlock(m);
    while (nw > 0 || nr > 0);
    lock(m);
}
nw=1;
unlock(m);
\end{verbatim}

\textit{enter\textsubscript{r}}:
\begin{verbatim}
lock(m);
while (nw) {
    unlock(m);
    while (nw);
    lock(m);
}
nr=nr+1;
unlock(m);
\end{verbatim}
Outline

- Condition variables
  - Definition
  - Usage patterns and examples

- Semaphores
  - Definition
  - Producer-consumer example
Condition Variables

A condition variable $c$ has two basic operations:

- $\text{wait}(l,c)$: wait on a condition $c$ using lock $l$
- $\text{signal}(l,c)$: signal condition

$\text{wait}(l,c)$ is a blocking operation
**Condition Variables**

- `wait(l,c)`
  - Waits for a condition to be signaled
  - While it is waiting, the lock is released
  - When we continue after the wait, the lock has been re-acquired

- `signal(l,c)`
  - Signal the condition
  - In this version, also release the lock; the next use of the released lock is a process that was previously blocked on the wait.
    - In some implementations, signals do no release the lock
Basic Usage Pattern

// acquire a lock
lock(l);

... if (test) {
    // wait for a condition
    wait(l,c);
}

... // release a lock
unlock(l)

// acquire a lock
lock(l);

... if (other test) {
    // signal condition
    signal(l,c);
}
else {
    // release a lock
    unlock(l);
}
Locks + Condition Variables

- Condition variable is used with a lock
  - `wait(l, c)` assumes that it has the lock when called

- In this version, a program never signals a condition unless some process is waiting
  - `waiting(l, c)`: returns true or false depending on whether or not there is a process blocked on the condition—you must hold the lock when this is executed.
  - More common implementations may simply ignore a signal when there is no thread/process waiting
Example: Readers and Writers

```c
enter_r:
    lock(m);
    while (nw) {
        unlock(m);
        while (nw);
        lock(m);
    }
    nr=nr+1;
    unlock(m);
```
Example: Readers and Writers

```
enter_r:
    lock(m);
    while (nw) {
        unlock(m);
        while (nw);
        lock(m);
    }
    nr=nr+1;
    unlock(m);

enter_r:
    lock(m);
    if (nw) {
        wait(m, cr);
    }
    nr=nr+1;
    if (waiting(m, cr)) {
        signal(m, cr);
    } else {
        unlock(m);
    }
```
Example: Readers and Writers

```c
exit_w:
    nw=0;

exit_w:
    lock(m);
    nw=0;
    if (waiting(m,cr)) {
        signal(m,cr);
    }
    else {
        unlock(m);
    }
```
Semaphores

- A semaphore is a non-negative integer, with the following operations
  - \( P(s) \): if \( s > 0 \), decremented by 1, otherwise wait
  - \( V(s) \): increment \( s \) by 1 and wake up one of the waiting processes (if any)
  - \( P(s) \) and \( V(s) \) must be executed atomically

- A semaphore can be used to control access to a critical section

\[
P(s) \quad \text{Initial value of } s \text{ determines the maximum no. of processes in the critical section}
\]

\[
\text{<critical section>}
\]

\[
V(s)
\]
Producer-Consumer Example

producer:
while (1) {
    item=produce();
    fifo_put(item);
}

consumer:
while (1) {
    item=fifo_get();
    consume(item);
}
Read when FIFO is not empty

producer:
while (1) {
    item=produce();
    fifo_put(item);
}

consumer:
while (1) {
    item=fifo_get();
    consume(item);
}
Write when FIFO is not full

producer:
while (1) {
    item = produce();
    fifo_put(item);
}

consumer:
while (1) {
    item = fifo_get();
    consume(item);
}