Monitors

CS377 - Parallel Programming

Marc L. Smith

Monitors

Monitor structure:

```
monitor name {
   declarations of permanent (static) variables
   initialization code -- executes first
   procedures (methods)
}
```

Program structure:

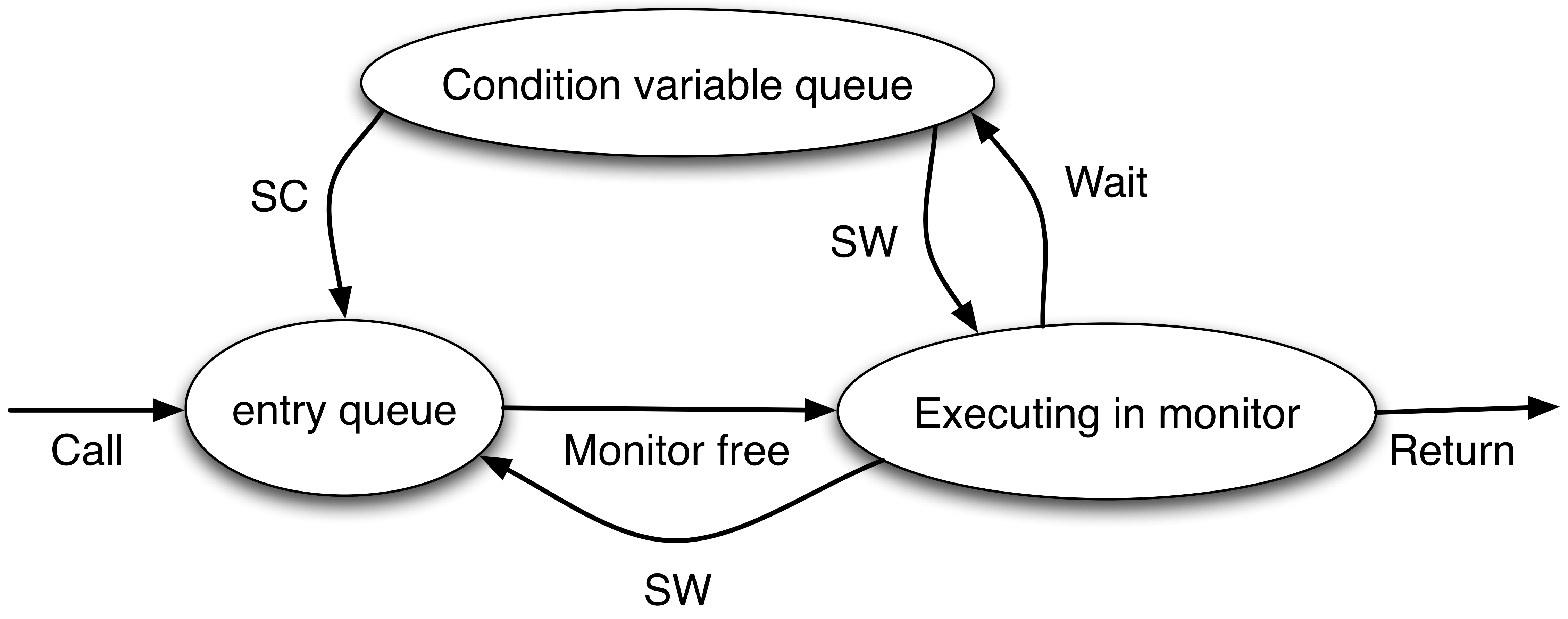
```
      monitor1 ... monitorM

process1      ...      processN
```

- processes interact indirectly by using the same monitor
- processes call monitor procedures
- at most one call active in a monitor at a time -- by definition
- explicit signaling using condition variables
- monitor invariant: predicate about local state that is true when no call is active

Condition Variables

```
cond cv;          # queue of delayed processes; initially empty

wait(cv);         # block on cv's queue AND release monitor lock

signal(cv);       # awaken one process on cv's queue, if there is one

questions about signal:
   which one to awaken?  default is oldest (FIFO queue)
   who executes next?  the signaled process?  or the signaler?

signaling disciplines:
   signal and continue (SC) -- signaler goes next; used in Java, Unix, Pthreads
   signal and wait (SW) -- signaled process goes next; used in Hoare's paper

SW is preemptive; SC is not

state diagram for synchronization in monitors: see next slide!
```

State diagram for synchronization in monitors *



* MPD text, Figure 5.1, p. 209

We can implement semaphores using a monitor...

Let’s look at two examples

Semaphore Example 1

```
monitor Semaphore {
   int s = 0;      ## s >= 0
   cond pos;       # signaled when s > 0

   procedure Psem() {
      while (s == 0) wait(pos);
      s = s-1;
   }

   procedure Vsem() {
      s = s+1;
      signal(pos);
   }
}
```

Works for both SC and SW

How?

Not FIFO for SC

Why?

Semaphore Example 2

```
monitor FIFOsemaphore {
   int s = 0;        ## s >= 0
   cond pos;         # signaled when s > 0

   procedure Psem() {
      if (s == 0)
         wait(pos);
      else
         s = s-1;
   }

   procedure Vsem() {
      if (empty(pos))
         s = s+1;
      else
         signal(pos);
   }
}
```

Uses passing the condition

FIFO for both SC and SW