Lab2: Semaphores and Monitors

Credits

- Material in this slide set is from G. Andrews, “Foundation of Multithreaded, Parallel, and Distributed Programming”, Addison Wesley, 2000.

Semaphores

- Used for
  - Mutual Exclusion (homework 2)
  - Barriers
    - Signaling events (lab 2)
- Low level mechanism
  - Easy to make mistakes
  - Hard to understand

Monitors

- Program modules that provide more structure than semaphores
  - Abstract objects that are well encapsulate
  - Only one process or thread can be in the monitor at one time
  - Use condition variables

Monitors

- Active processes
  - Interact by calling procedure in the same monitor
- Passive monitors
  - Provide methods to manipulate data but no access to internal data structures

Monitors

- Benefits
  - Implementation independent
    - Only visible effects matter
  - Callers independent
    - Monitor writers can change the implementation as long as the same effects are maintained

Monitors

- Provide mutual exclusion implicitly
  - Only one instance of the monitor can be active at one time
    - 2 calls to two different procedures in the same monitor? NO!
    - 2 calls to the same procedure in the same monitor? NO!

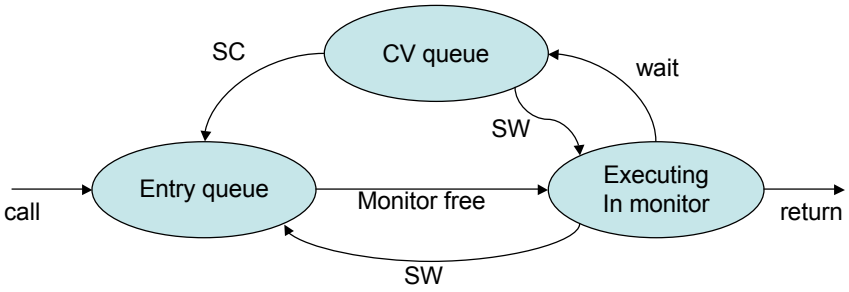
Monitors

- Condition Variables
  - Delay a process
    - Monitor's state fails to satisfy Boolean condition
  - Awaken a process
    - Awaken a process or processes if the condition becomes true

Is this possible?

Monitors

- Signaling Disciplines



From G. Andrews, “Foundation of Multithreaded, Parallel, and Distributed Programming”, Addison Wesley, 2000

Monitors

- Signal and continue: signalers continues and the signaled process executes at some later time
  - Non-preemptive
- Signal and wait: signaler waits and the signaled process executes now
  - Preemptive

Monitors

- Signal and Wait in Unix?