Operating Systems
COMS W4118
Lecture 16

Alexander Roth

2015 – 03 – 24

1 Semaphore Uses: Execution Order

- Semaphore allows to different threads to wait and post separately.
- Can be used when you want to impose order.
- Mutex just makes sure that threads cannot overlap.
- Cannonical example is the producer-consumer model.

2 Conditions Vs. Semaphore

- Semaphores are sticky, they have a number.
- Every thread that calls `sem_post` will increment the semaphore.
- Every thread that calls `sem_wait` will decrement the semaphore.
- There is no waiting.
- The only time you block with a semaphore is when the semaphore goes to 0, the number must be positive.
- The ordering doesn't matter in terms of coordination. Things can get lost.
- Semaphores have memory
- Condition variables do not. They only send a signal that is expected to be caught by a

3 `sem_post`

- Locks the mutex within the semaphore.
- Increments the count of the semaphore.
- Alert the threads to the increment with `pthread_cond_signal`.
- Unlock the mutex within the semaphore.

4 `sem_wait`

- Lock the mutex
- Check that the count is not zero.
- If the count is zero set the wait, come back and check later.
- Otherwisre, decrement the count.
- Unlock the mutex within the semaphore.

5 Mesa Semantics

- Modern operating systems use this type of competition variables.
- When you come out a wait, you have to reacquire the lock; otherwise another process can come in and take this lock.
- **Hoare Semantics:** Suspends the signaling thread, and immediately transfers control to the woken thread.
- **Mesa Semantics:** `signal()` moves a signle wiating thread from the blocked state to a runnable state, then the signaling thread continues until it exists the monitor.
- Typically, you will signal before you unlock the mutex.

6 System Calls

- The kernel address space contains every process mapped to it.
- Every process has the top first gigabyte mapped to the kernel.
- Virtual address spaces can be mapped to single physical addresses.
- If you call a system call that is executed on behalf of a process, there is a little bit of the stack allocated for each process.

- A small amount of stack is allocated in the kernel for any process calls that would appear.
- System calls elevate privilege of user process.
- The CPU switches into privilege mode, which allows all instructions to become available.
- Thus, any user can inject code into the system.
- System calls must verify every single parameter to make sure it is not doing anything illegal.
- You should not be able to pass any pointer to write because you can overwrite memory you weren't supposed to touch.

6.1 System Call Dispatch

- A user process invokes a system call.
- There is some facility that allows the CPU to switch from the user mode to kernel mode.
- There are three ways that a running process can be interrupted.
    1. Hardware Interrupt - Sent from device to processor (timers, etc.)
    2. Software Interrupt - Instruction loaded by processor, you are interrupting yourself to do something else. This is the only way to jump into the kernel and make system calls.
    3. Processor - Exceptions (invalid instructions)