CIS 452 - Operating Systems Concepts

Nathan Bowman

Images taken from Silberschatz book

Deadlock and Priority Inversion

Two of the issues that can arise during process synchronization are *deadlock* and *priority inversion*

Deadlock occurs when two or more processes are waiting on an event that can only be caused by one of the waiting processes

This results in all of the processes waiting forever

Two processes are sharing semaphores S and Q. How can the following go wrong?

```
P0                P1

wait(S);          wait(Q);
wait(Q);          wait(S);

...

signal(S);        signal(Q);
signal(Q);        signal(S);
```

We will discuss more about deadlocks later

Priority inversion occurs when a lower-priority process causes a higher-priority processs to wait longer than it should

Assume there are three processes, each at a different priority level: L(ow), M(edium), and H(igh)

L is modifying kernel data needed by H, so H must wait. This itself is not priority inversion

Process L is then preempted when process M wants to run

L was still holding the lock when preempted, so H cannot run

M has indirectly been scheduled ahead of H, which is priority inversion

This may not sound that bad, but it can be. Process H was high priority for a reason. Also, there could be more than one medium-priority process

How to avoid priority inversion?

This situation cannot happen if there are just two priorities, but that is insufficient for most systems

Priority-inheritance protocol -- any process accessing a resource needed by higher-priority process inherits higher priority until resource is released

In our example, L would no longer be preempted by M because H is waiting for L

Once L finishes, its priority reverts, and H can be scheduled instead of M