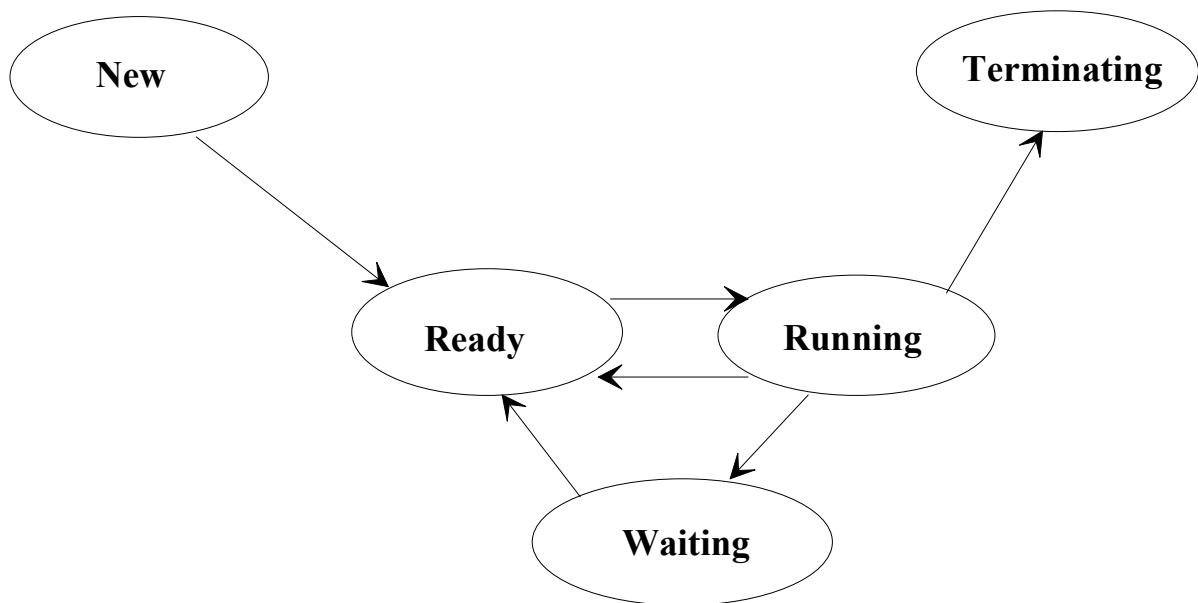


Class Notes Aug 28 2006.

We have two sorts of entities in our computer now, the operating system and processes. Bizarrely enough, the OS doesn't do anything except respond to interrupts. In that sense, the OS is completely passive. It provides services that are invoked with interrupts. The only thing that can do anything is a process.

If we look at the lifetime of a process, it will look as follows (this is in the book). These states are normally recorded in the PCB. Each state normally represents a different queue within the OS's data space.



For uniprocessors, the Ready and the Running state do not have to be explicitly different, however for multiprocessors it is essential to distinguish between the two. When a process requests I/O, the following steps take place:

0. The process is in the running state.
1. System Call
2. OS starts I/O
3. Process is placed in waiting state.
4. Other stuff goes on for a while.
5. An I/O interrupt occurs.
6. The OS processes the interrupt
7. The OS places the process in the ready state.
8. Eventually the process enters the running state again.

Suppose process A is running, process B is waiting for I/O. We can switch from one process to another as follows. This is called a context switch.

1. An I/O interrupt occurs
2. Current context is saved.
3. I/O interrupt is processed.
4. Process B is made ready.
5. Scheduler runs.
6. Process B is selected.
7. Process B's context is loaded.

What exactly is a context anyway? It is the contents of all registers in the computer.

In addition to the other stuff that we placed in the PCB we also have a copy of all the registers. These are saved when a process is interrupted, and restored when a process enters the running state.

What creates a process?

How does a process die?

How do we communicate between processes?

How do we handle shared data structures?