Lecture Topics: 11/1

- Processes
  - processes vs. programs
- Process Management
  - process control blocks

Processes

- A *program* is passive
  - just a file on the disk with code that is potentially runnable
- A *process* is active
  - one instance of a program in execution
  - also called *job*, *task*, *sequential process*
- At any given time, there may be many processes running the same program
  - even though they run the same program, they are separate and independent

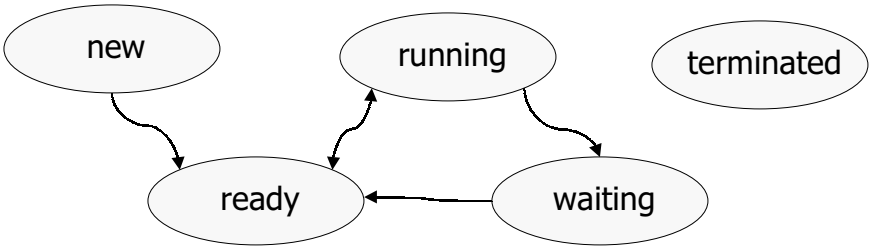
What's in a Process?

- A process consists of at least:
  - the code for the running program
  - the data for the running program
  - an execution stack tracing the state of procedure calls made
  - the Program Counter, indicating the next instruction
  - a set of general-purpose registers with current values
  - a set of operating system resources (open files, connections to other programs, etc.)

Process State

- Each process has an execution state that indicates what it is currently doing:
  - ready: waiting to be assigned to the CPU
  - running: executing instructions on the CPU
  - waiting: waiting for an event, e.g., I/O completion
- As a program executes, it moves from state to state

Process State Changing



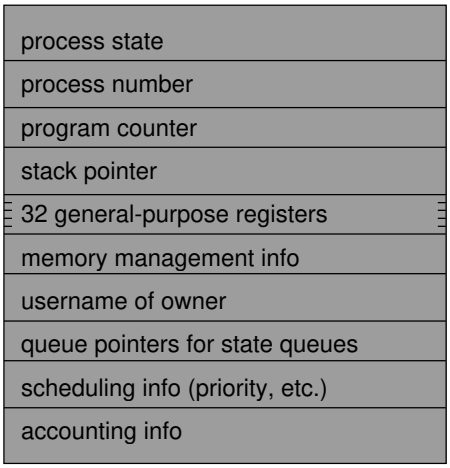
Processes move from state to state as a result of actions they perform (e.g., system calls), OS actions (rescheduling) and external actions (interrupts)

Process Data Structures

- At any time, there are many processes, each in its particular state.
- The OS must have data structures representing each process
  - this data structure is called the PCB (*Process Control Block*)
- The PCB contains all of the info about a process, plus hardware state (PC, regs) when the process is not running

PCB

PCB contains lots of information, e.g.:



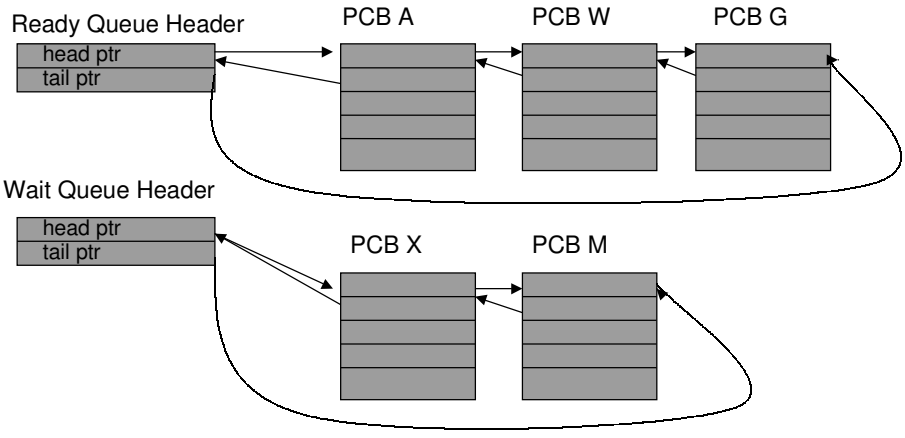
PCBs and Hardware State

- When a process is running its program counter, stack pointer, registers, etc., are loaded on the CPU (i.e., the processor hardware registers contain the values)
- When the OS is ready to start executing a new process, it loads the hardware registers from the values stored in that process' PCB.
- The process of switching the CPU from one process to another is called a context switch. Timesharing systems may do 100s or 1000s of context switches a second!

State Queues

- The OS maintains a collection of queues that represent the state of all processes in the system.
- There is typically one queue for each state, e.g., ready, waiting for I/O, etc.
- Each PCB is queued onto a state queue according to its current state.
- As a process changes state, its PCB is unlinked from one queue and linked onto another.

State Queues



There may be many wait queues, one for each type of wait (specific device, timer, message,…).

PCBs and State Queues

- PCBs are data structures, dynamically allocated in OS memory.
- When a process is created, a PCB is allocated to it, initialized, and placed on the correct queue.
- As the process computes, its PCB moves from queue to queue.
- When the process is terminated, its PCB is deallocated.

Creating a Process

- One process can create other processes to do its work. These are *child processes* and the creator is the *parent*.
- When a child is created, the parent may either wait for it to finish its task, or continue in parallel.
- In Unix, subprocesses are created by a call to `fork()`, a Unix system call.

Cooperating Processes

- Processes can be independent or they can cooperate to accomplish one job.
- Cooperating processes can be used:
  - to gain speedup by overlapping activities or performing work in parallel
  - to better structure an application
  - to share information between jobs
- Processes can form a pipeline where each produces work for the next stage that consumes it, and so on.