Processes & Threads

As we know…

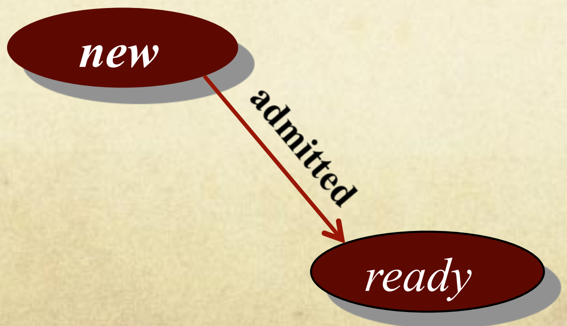
- Computer system essentially used to execute/run programs
  - May run several different programs
    - E.g., e-mail client, browser, editor, music player
  - May run multiple instances of same program
    - E.g., Multiple instances of browser, editor
- Need some way to represent running programs internally

Process

- Abstraction for a *running program instance*
  - Represents an *activity* of some kind – hence the name!
  - Used by OS to manage concurrently running programs
- A *process* is not equivalent to a *program*
  - TextEdit → program
  - Running instance of TextEdit → process
- More to *process* than just a program
  - Has program, data, state information…
  - Owns resources (memory, etc.)

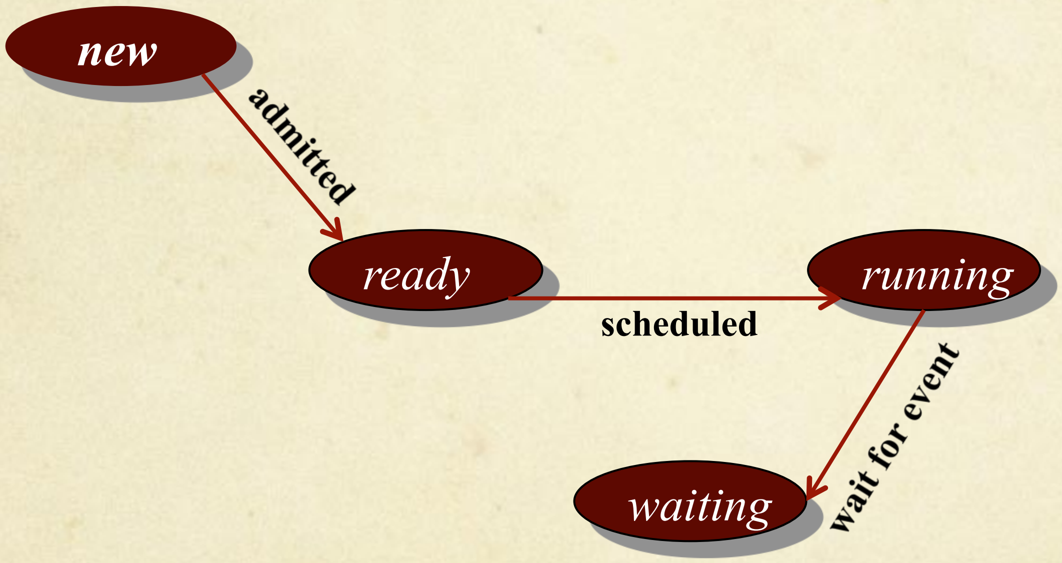
Program execution

- To execute a program, corresponding *process* must be *created*
  - All processes can be created when system starts
    - Okay for embedded system like a dish washer
    - Not so great for general purpose personal computer
  - Of course, some processes are created when system starts
  - Others are created as requested/needed by user/system
- After creation, process becomes *active* or *ready* for execution



Program execution

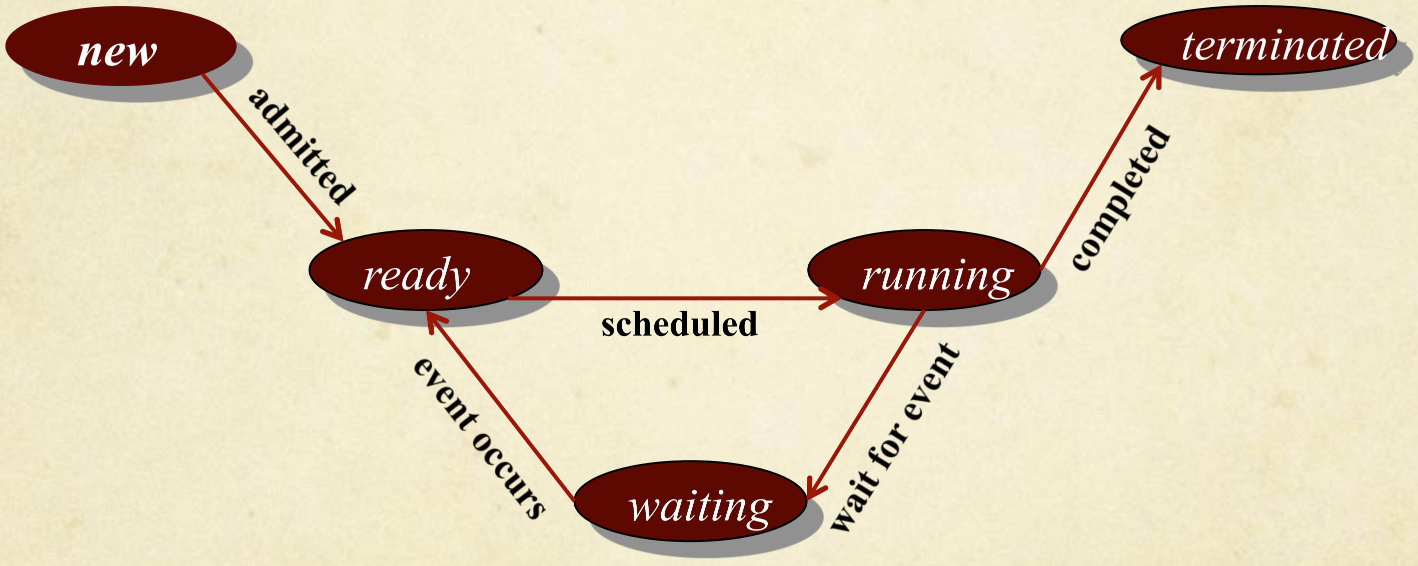
- If processor is free, ready process can be *dispatched/scheduled*



- Process may sometimes have to *wait* for event (e.g. I/O)
  - → Process gets *blocked*

Program execution

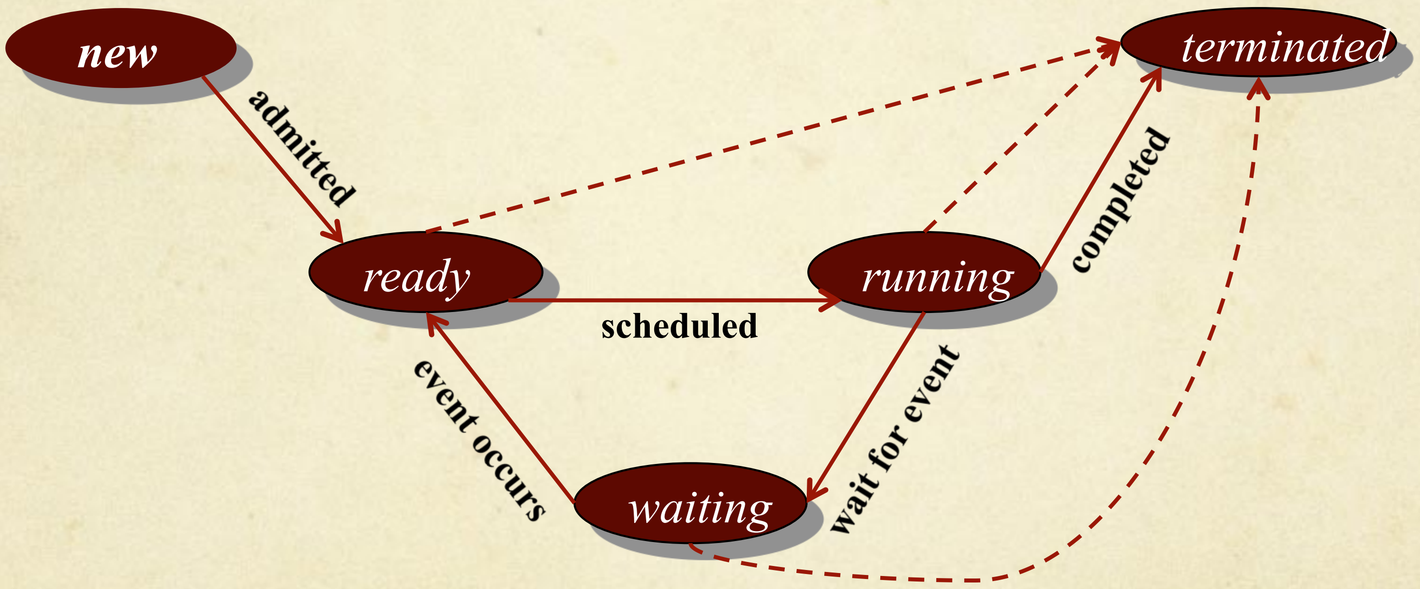
- Once event being waited for occurs, process is *ready* again
  - This *ready* – *running* – *waiting* cycle can repeat



- Once process is complete, it may be *terminated*

Program execution

- Process may be killed explicitly or terminated due to error



More about processes…

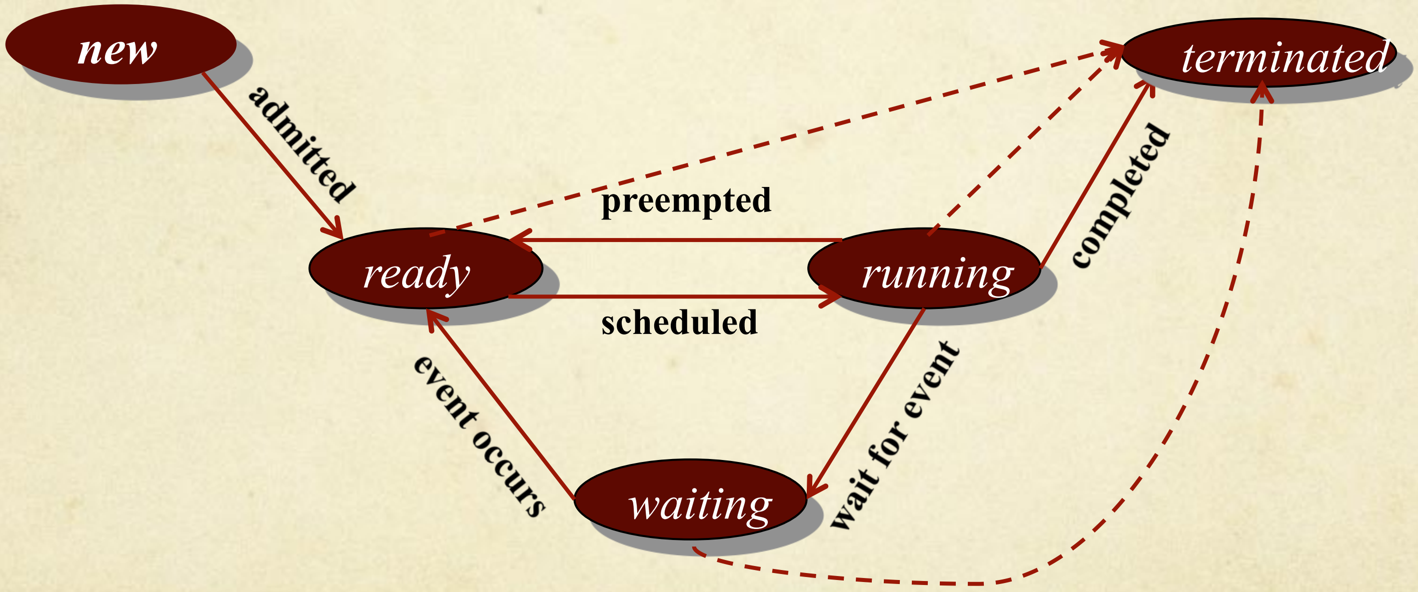
- Multiple processes can be simultaneously *active/ready*

- Only *one* process can actually *run* on a processor at a time
    - For now, let us assume single processor system

- OS switches between multiple processes as appropriate
    - → *Multiprogramming*

This introduces more concepts…

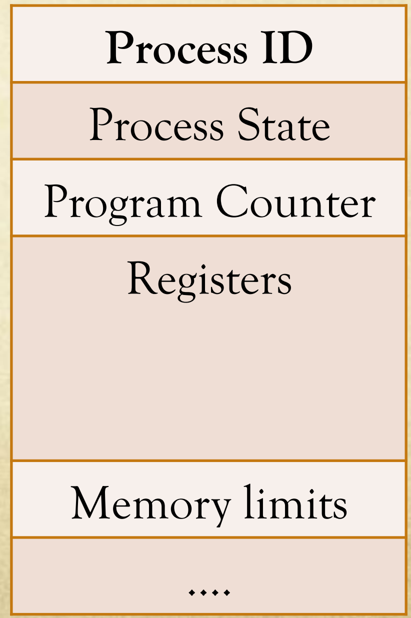
- Decide what process to run when → *scheduling policy*
  - Brings up another possible scenario – *preemption*



- *We will discuss scheduling policies in detail later*

To manage multiple processes…

- …information about each process must be maintained
  - *Process control block* used for this
    - Process ID
    - Process State (ready, running etc.)
    - Program Counter – address of next instruction to be executed
    - Registers – general purpose registers, stack pointer etc.
    - Scheduling information
    - Memory management information
    - Accounting information – time limits, etc.
    - …


Process ID
Process State
Program Counter
Registers
Memory limits
….