Process Scheduling

- Maximize CPU use, quickly switch processes onto CPU for time sharing
- **Process scheduler** selects among available processes for next execution on CPU
- Maintains **scheduling queues** of processes
  - **Job queue** – set of all processes in the system
  - **Ready queue** – set of all processes residing in main memory, ready and waiting to execute
  - **Device queues** – set of processes waiting for an I/O device
  - Processes migrate among the various queues

Schedulers

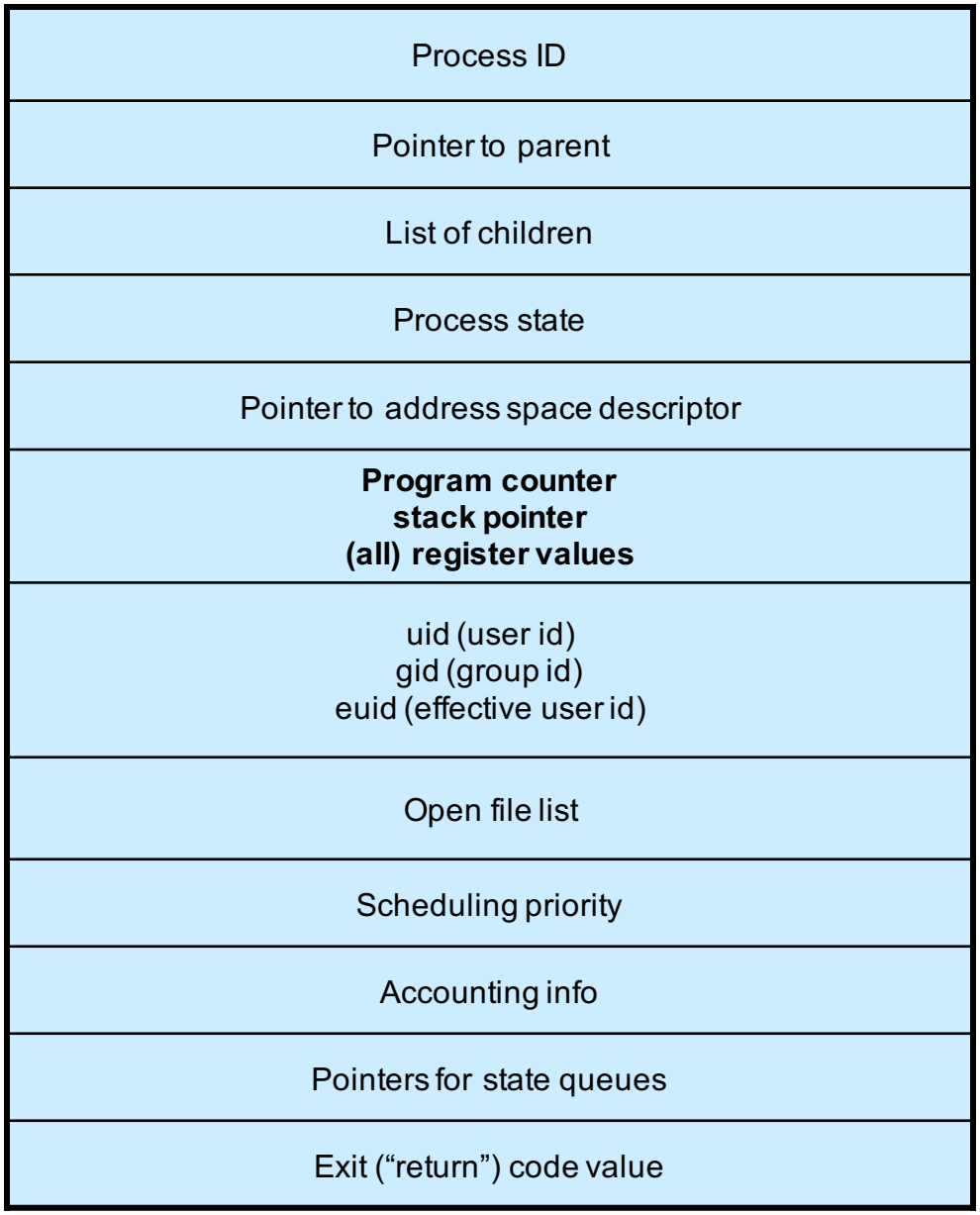
- Processes can be described as either:
  - **I/O-bound process** – spends more time doing I/O than computations, many short CPU bursts
  - **CPU-bound process** – spends more time doing computations; few very long CPU bursts

PCBs and CPU state

- When a process is running, its CPU state is inside the CPU
  - PC, SP, registers
  - CPU contains current values
- When the OS gets control because of a …
  - Trap: Program executes a syscall
  - Exception: Program does something unexpected (e.g., page fault)
  - Interrupt: A hardware device requests service

  the OS saves the CPU state of the running process in that process's PCB

- When the OS returns the process to the running state, it loads the hardware registers with values from that process's PCB – general purpose registers, stack pointer, instruction pointer
- The act of switching the CPU from one process to another is called a context switch
  - systems may do 100s or 1000s of switches/sec.
  - takes a few microseconds on today's hardware
- Choosing which process to run next is called scheduling

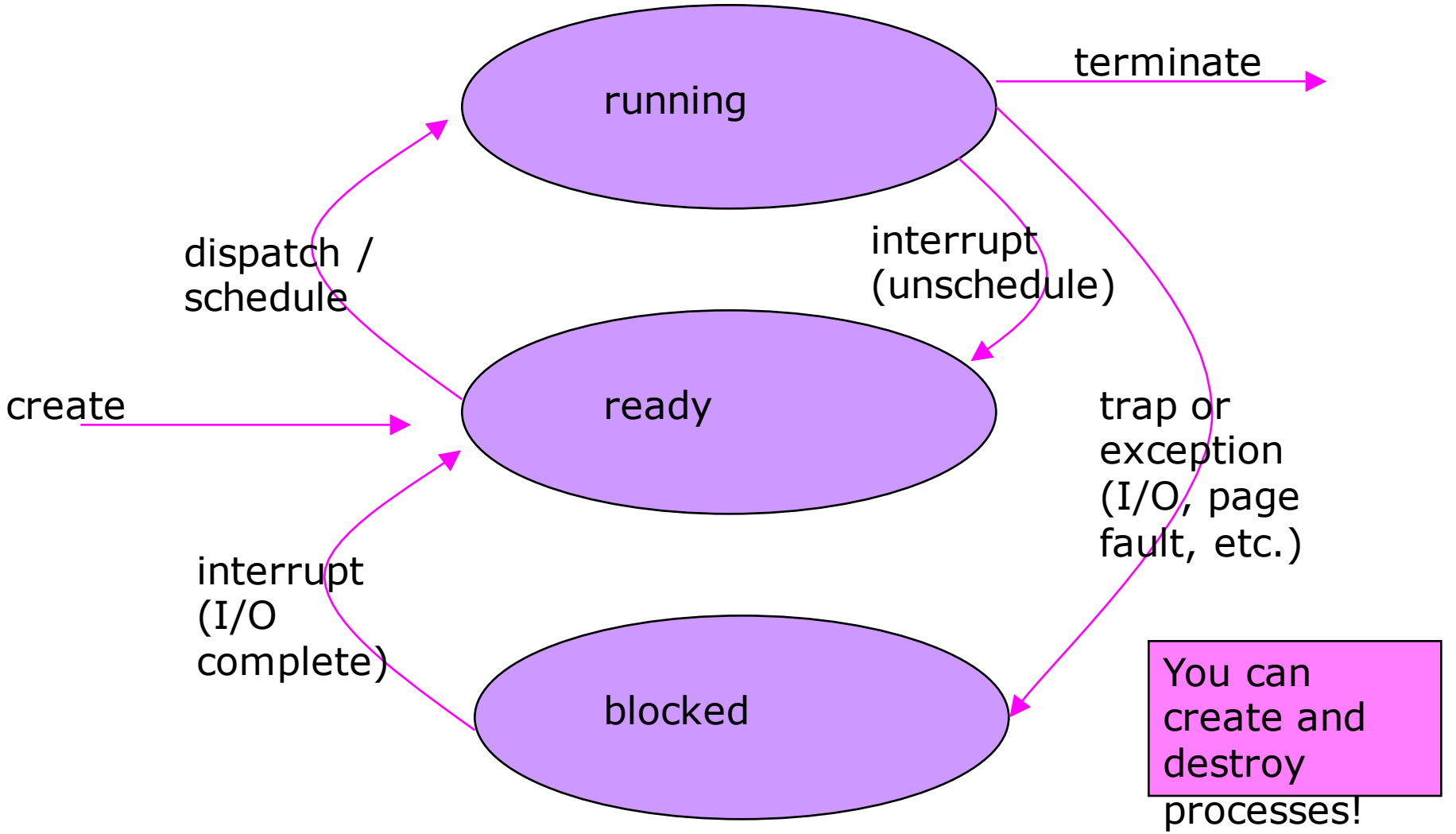


This is (a simplification of) what each of those PCBs looks like inside!

Process execution states

- Each process has an execution state, which indicates what it's currently doing
  - ready: waiting to be assigned to a CPU
    - could run, but another process has the CPU
  - running: executing on a CPU
    - it's the process that currently controls the CPU
  - waiting (aka "blocked"): waiting for an event, e.g., I/O completion, or a message from (or the completion of) another process
    - cannot make progress until the event happens
- As a process executes, it moves from state to state
  - UNIX: run `ps`, STAT column shows current state
  - which state is a process in most of the time?

Process states and state transitions



Context Switch

- When CPU switches to another process, the system must **save the state** of the old process and load the **saved state** for the new process via a **context switch**
- **Context** of a process represented in the PCB
- Context-switch time is overhead; the system does no useful work while switching
  - The more complex the OS and the PCB ➔ the longer the context switch
- Time dependent on hardware support
  - Some hardware provides multiple sets of registers per CPU ➔ multiple contexts loaded at once

Process Creation

- **Parent** process create **children** processes, which, in turn create other processes, forming a **tree** of processes
- Generally, process identified and managed via a **process identifier** (**pid**)
- Resource sharing options
  - Parent and children share all resources
  - Children share subset of parent's resources
  - Parent and child share no resources
- Execution options
  - Parent and children execute concurrently
  - Parent waits until children terminate

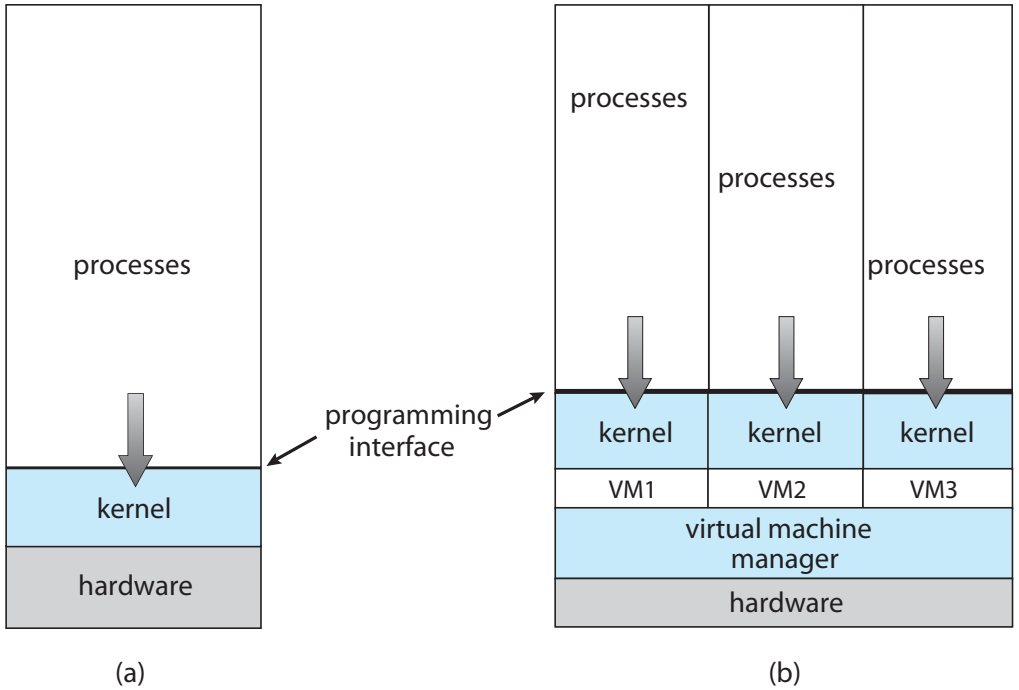
Process Termination

- Some operating systems do not allow child to exist if its parent has terminated.
  - If a process terminates, then all its children must also be terminated.
  - **cascading termination**. All children, grandchildren, etc. are terminated.
  - The termination is initiated by the operating system.
- The parent process may wait for termination of a child process by using the `wait()` system call. The call returns status information and the pid of the terminated process

```
pid = wait(&status);
```

- If no parent waiting (did not invoke `wait()`) process is a **zombie**
- If parent terminated without invoking `wait`, process is an **orphan**

Computing Environments - Virtualization

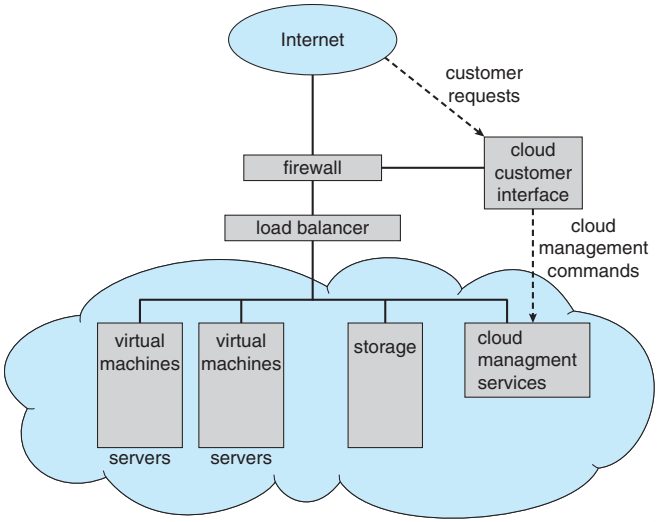


Computing Environments – Cloud Computing

- Delivers computing, storage, even apps as a service across a network
- Logical extension of virtualization because it uses virtualization as the base for it functionality.
  - Amazon **EC2** has thousands of servers, millions of virtual machines, petabytes of storage available across the Internet, pay based on usage
- Many types
  - **Public cloud** – available via Internet to anyone willing to pay
  - **Private cloud** – run by a company for the company's own use
  - **Hybrid cloud** – includes both public and private cloud components

Computing Environments – Cloud Computing

- Cloud computing environments composed of traditional OSes, plus VMMs, plus cloud management tools
  - Internet connectivity requires security like firewalls
  - Load balancers spread traffic across multiple applications



HOW DOES THIS LOOK?!?

Memory (Physical/Virtual)? Disk?

Devices?

Multicore? Multiprocessor?

Modes?

Microkernels?

Clouds?

Shells? Built-in commands? fork/exec?

Process states?

Context switches?

Pointers and doubly linked lists!!!