Operating System Design

Processes

Neda Nasiriani

Fall 2018

Review: Booting the OS

Booting Steps Review

1) Where is BIOS stored when your machine powers up?
    1) The BIOS is stored on Flash disk which is mapped to BIOS address in memory
2) What is in the RAM (main memory) when your machine powers up?
    1) Garbage!
3) What are example boot devices and who chooses the boot device for the BIOS?
    1) CD-ROM, Hard drive, …
    2) The user chooses it
4) What is a Master Boot Record (MBR) and what does it contain?
    1) First 512 Bytes of the boot device and contains the bootstrapper program and partition table

Processes

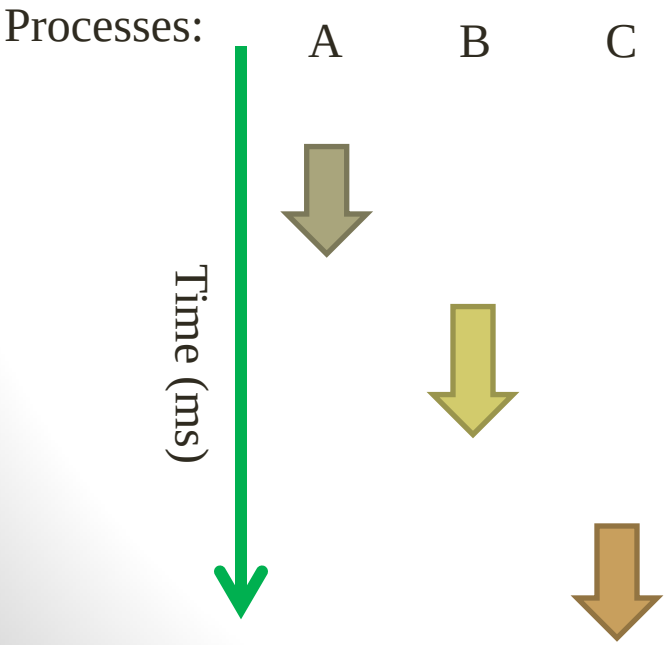
Processes

- What is a process?
  - Informally: a program in execution
- Examples of processes in a computer system
  - The Kernel and all its related processes
  - Web browser
  - Word editor
  - JVM
  - Python IDE
  - …
- How can you see the list of processes on your machine?
  - top, htop
  - ps -el

You want to design the OS to allow for multi processes running at the same time…

Specs of the multi-processor Computer System

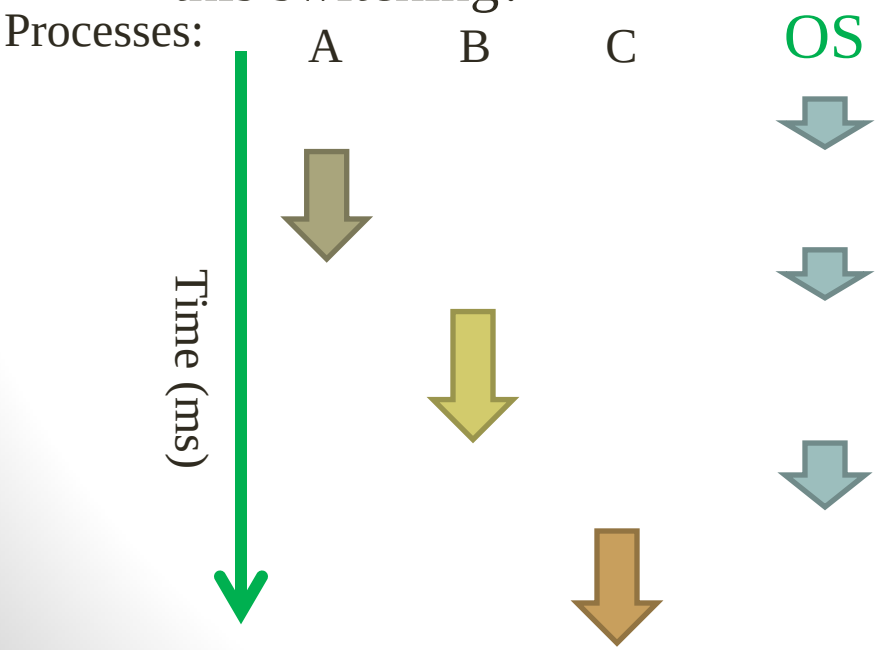
- We want processes to run concurrently, so (i) they can interact with each other, and (ii) maximize CPU utilization
  - Fact: at each time only one process can run on each processor
  - Remedy: So, we should switch processes fast enough so they feel like they are all running simultaneously (illusion)



How can this be implemented in a real computer system?

Specs of the multi-processor Computer System

- We want processes to run concurrently, so (i) they can interact with each other, and (ii) maximize CPU utilization
  - Fact: at each time only one process can run on each processor
  - Remedy: So, we should switch processes fast enough so they feel like they are all running simultaneously (illusion)
- Can the OS kernel as the main process in the system perform this switching?



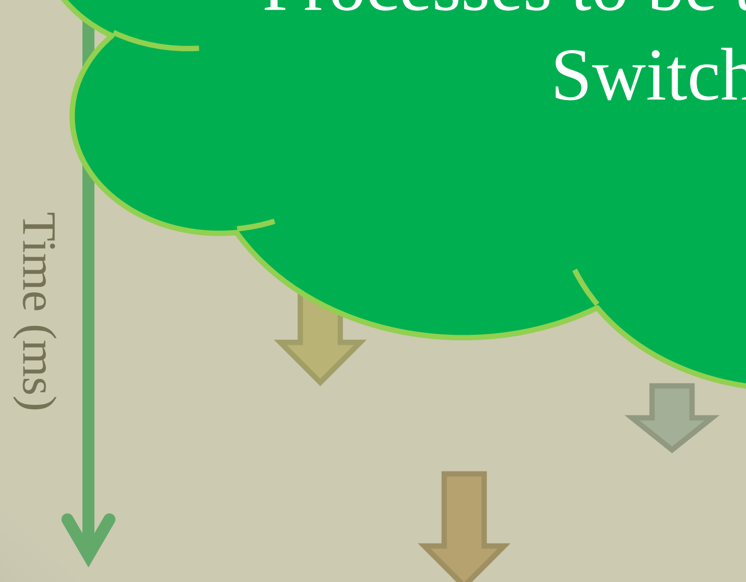
OS tasks?

- deciding who should run next,
- Handle interrupts if any happened
- …

Specs of the multi-processor Computer System

- We want processes to run concurrently, so (i) they can interact with each other, and (ii) maximize CP… utiliz…
  - Fact: at ea… …sor
  - Rem… …el like
- Ca… t…

Processes:



What does the OS need to know about the Processes to be able to do this Switching?

…s?
- …ho should run next,
- Handle interrupts if any happened
- …

Processes Components

- What are the main components of a process?
  - Text section
    - The code
  - Stack
    - Local variables
    - Function parameters
    - …
  - Heap
    - Dynamically allocated memory
  - Data Section
    - Global variables
  - What else?

Processes Components

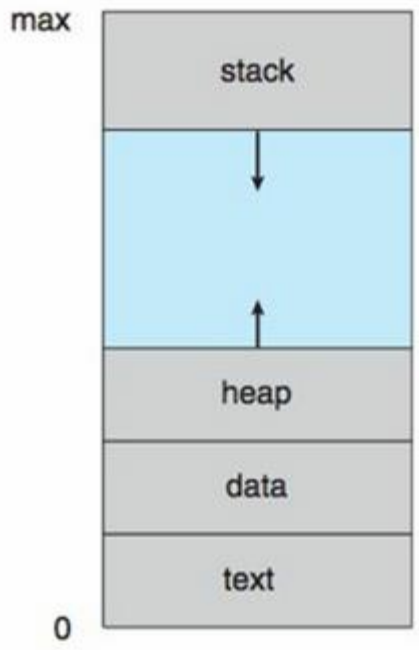
- Assume processes A is running in a system
  - The CPU decides to switch from process A to another process
  - What information will the CPU need to resume process A later?
    - Program Counter
    - Value of registers
- SO, a process is associated with the following components
  - Text section
  - Data section
  - Heap
  - Stack
  - Program Counter
  - Value of Registers

Process A

```
lw $t0, offset($s0)
lw $t1, offset($s1)
add $d, $t0, $t1
.
.
.
```

Processes Components

- A process is associated with the following components
  - Text section
  - Data section
  - Heap
  - Stack
  - Program Counter
  - Value of Registers
- The process in memory looks like this



Process States

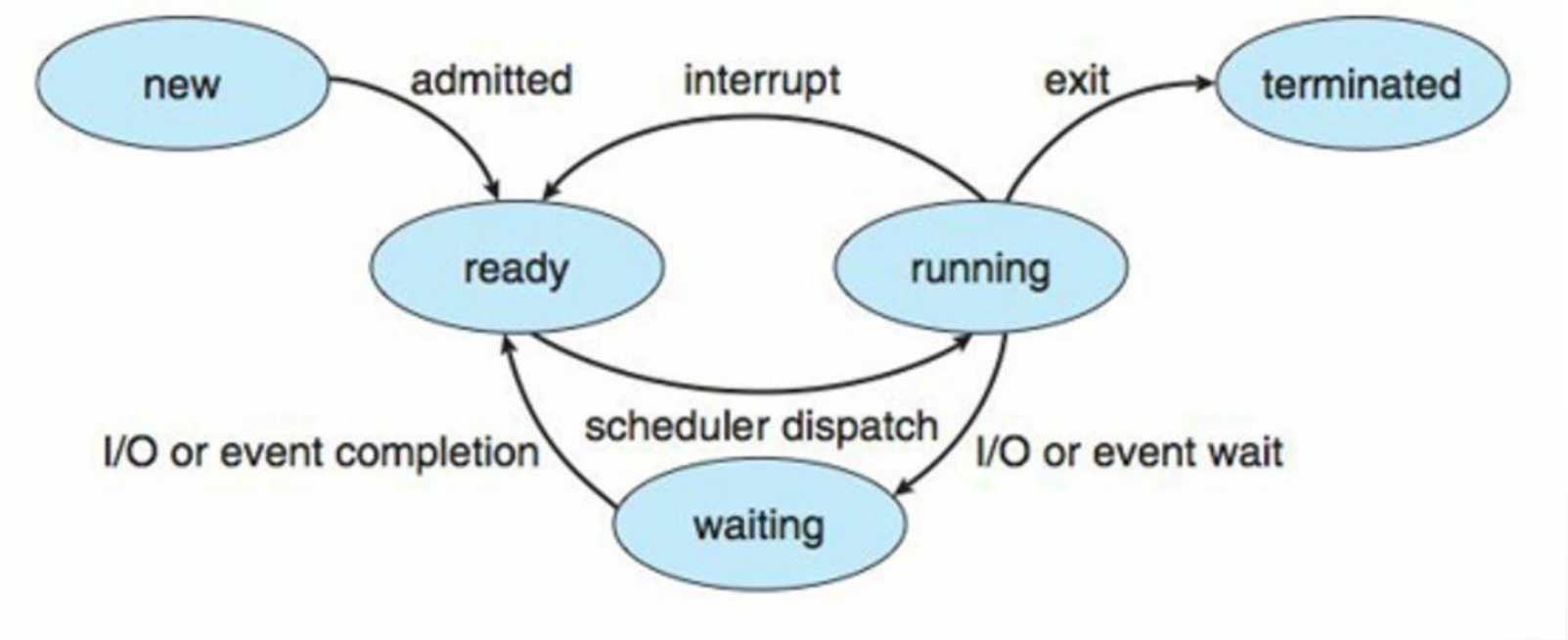
- When a process is running
  - Running
- When a program execution is paused so other processes can run
  - Ready
- When a process is waiting on an input or output (I/O operation)
  - Waiting
- When a process is just created
  - New
- When a process is all done
  - Terminated
- How does the state diagram of process look like?

FYR: Processes States

As a process executes in the system its state changes

- new: The process is being created.
- running: Instructions are being executed.
- waiting: The process is waiting for some event to occur.
- ready: The process is waiting to be assigned to a processor.
- terminated: The process has finished execution.

Process Lifecycle

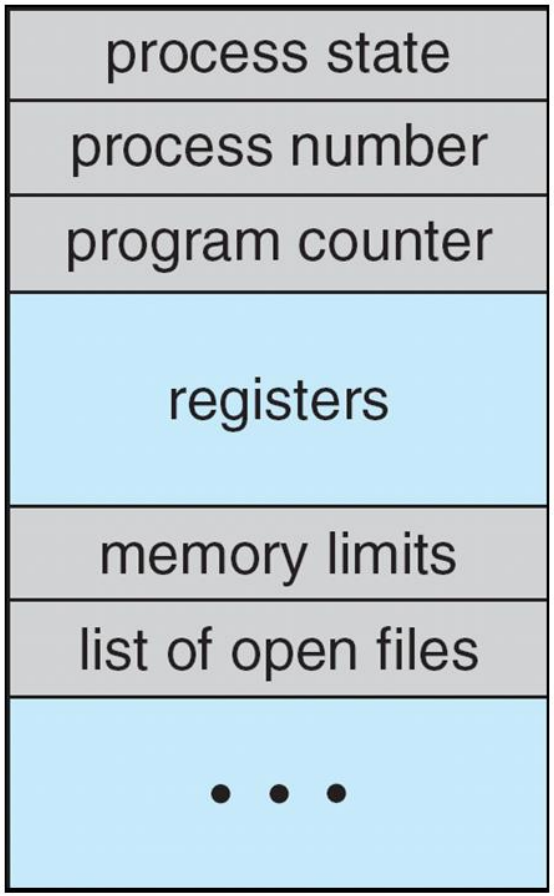


What other information is needed?

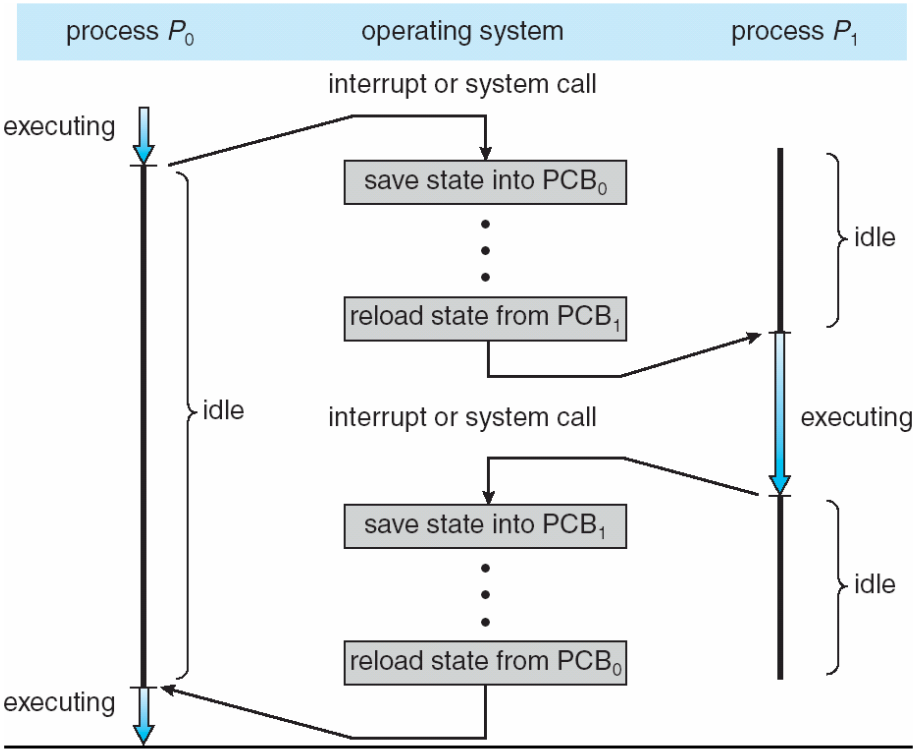
- If you want to design a scheduler to divide your time resource between a bunch of different processes, what info would you need in order to schedule effectively and fairly
  - Process state – running, waiting, etc
  - Program counter – location of instruction to next execute
  - CPU registers – contents of all process-centric registers
  - CPU scheduling information- priorities, scheduling queue pointers
  - Memory-management information – memory allocated to the process
  - Accounting information – CPU used, clock time elapsed since start, time limits
  - I/O status information – I/O devices allocated to process, list of open files

Where to keep that information?

- There is a data type called Process Control Block (PCB) that contains all this information about each process



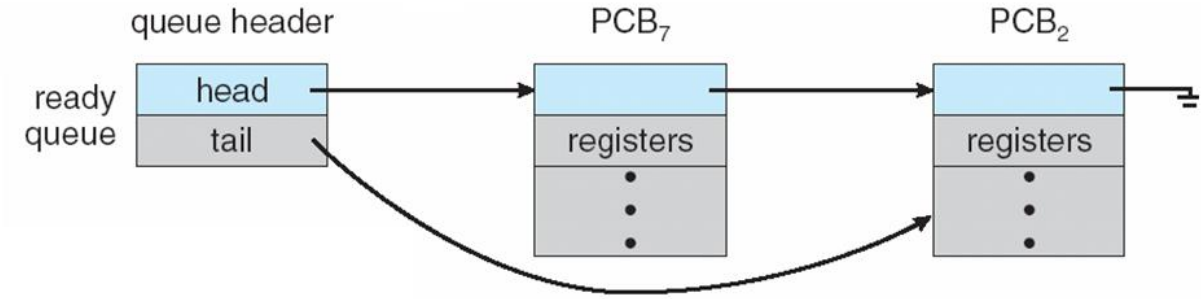
CPU Switch between Processes



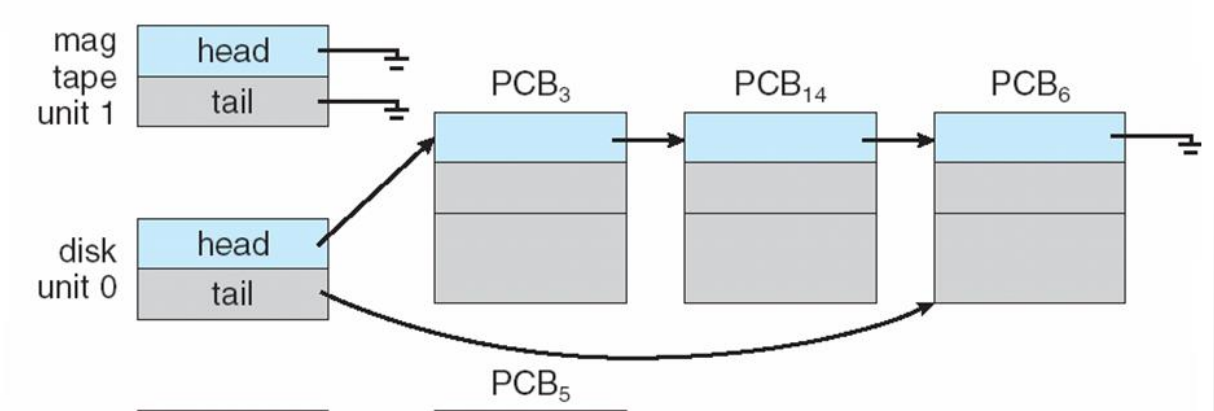
- Context Switch: When CPU switches to another process, the system must **save the state** of the old process and load the **saved state** for the new process via a **context switch**
- This time is pure overhead!

Scheduler

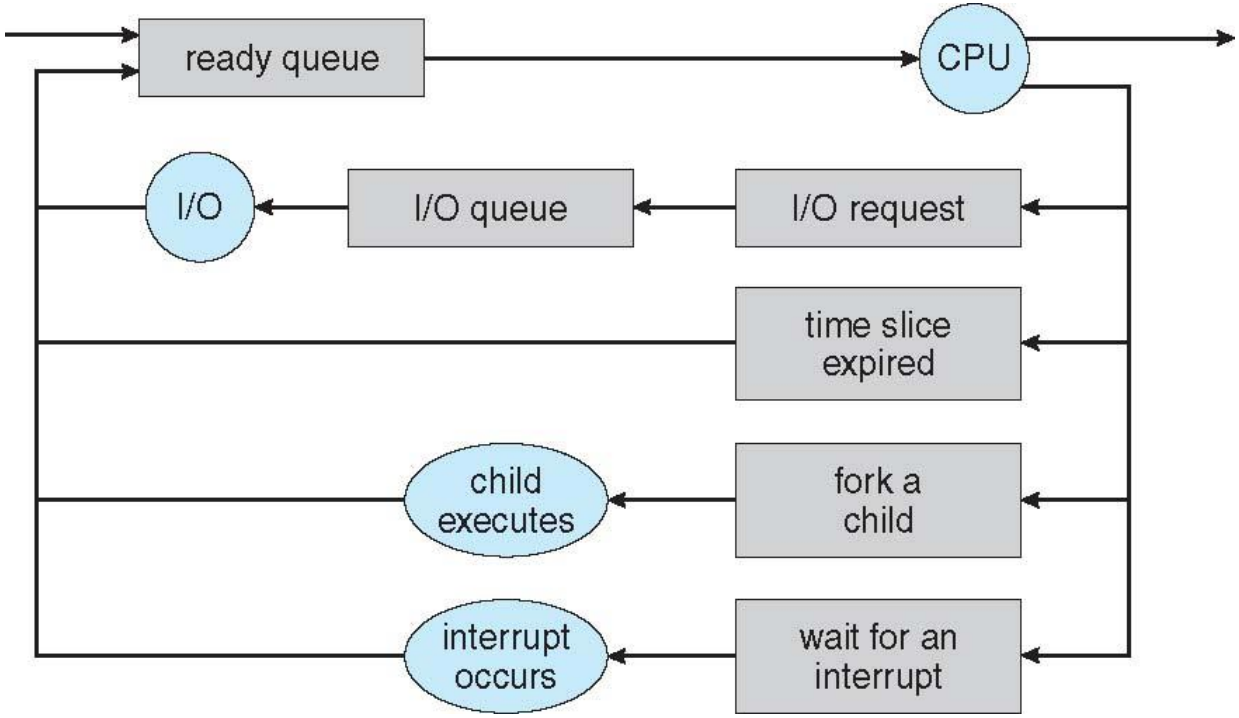
- A list of all processes PCBs is available to OS scheduler



- Ready queue: a list of all processes which are ready and waiting to execute
- Device queue: a list of all processes waiting for an I/O operation on a device, e.g., Disk queue, terminal queue
-



Scheduler

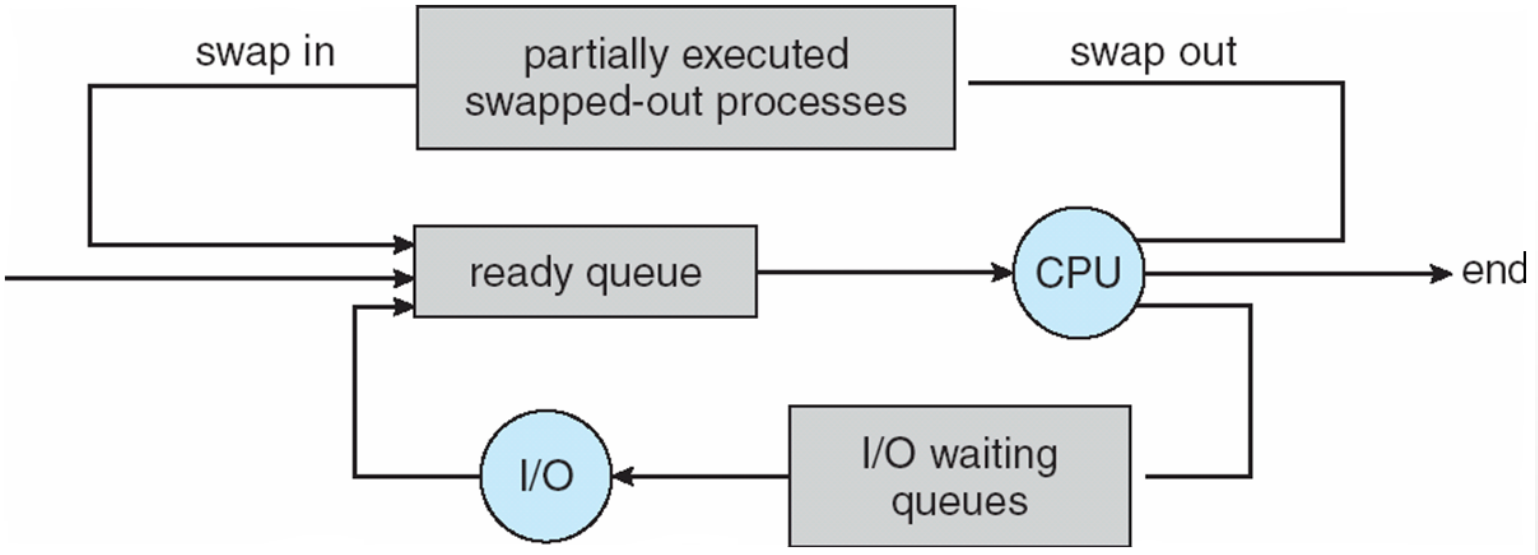

ready queue
CPU
I/O
I/O queue
I/O request
time slice expired
child executes
fork a child
interrupt occurs
wait for an interrupt

Scheduler

- **Short-term scheduler** (or **CPU scheduler**) – selects which process should be executed next and allocates CPU
  - Sometimes the only scheduler in a system
  - Short-term scheduler is invoked frequently (milliseconds) ⇒ (must be fast)
- **Long-term scheduler** (or **job scheduler**) – selects which processes should be brought into the ready queue
  - Long-term scheduler is invoked infrequently (seconds, minutes) ⇒ (may be slow)
  - The long-term scheduler controls the **degree of multiprogramming**
- 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
- Long-term scheduler strives for good ***process mix***

Medium-Term Scheduler

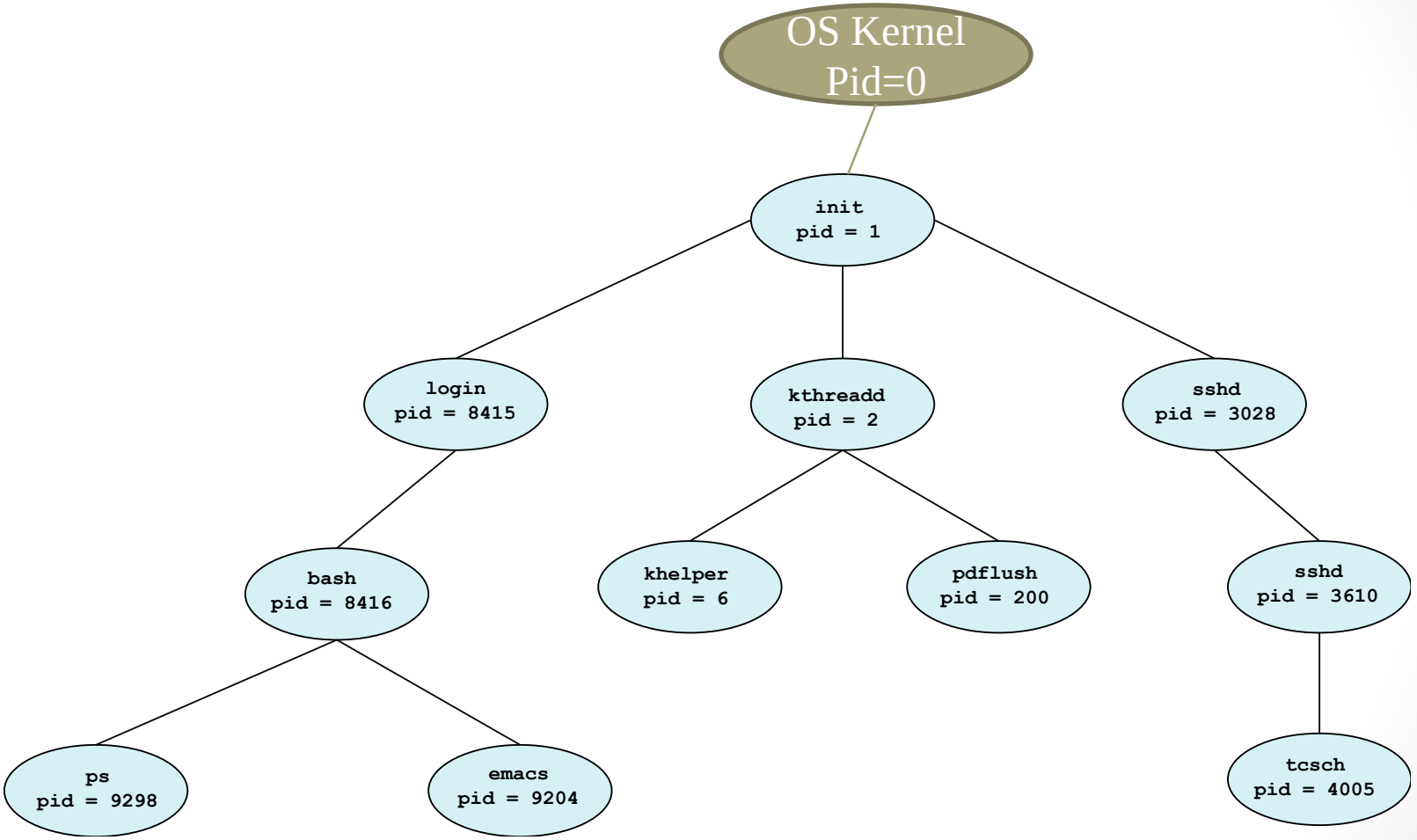
- **Medium-term scheduler** can be added if degree of multiple programming needs to decrease
  - Remove process from memory, store on disk, bring back in from disk to continue execution: **swapping**



Process Creation

- Processes may be created or deleted dynamically in the system
- Examples of creating a process within a process?
  - If you are designing a web server, you need to constantly listen to possible incoming requests
  - Also there could be 1000 of request every second, how can you address them all in a timely fashion?
  - You would like to run another program within your program
- Parent process creates children processes, which, in turn can create other processes, forming a tree of processes.

A Process Tree on Linux



Process Creation

- Resource sharing:
  - Parent and children share all resources,
  - Children share subset of parent's resources,
  - Parent and child share no resources.
- Execution:
  - Parent and children execute concurrently,
  - Parent may wait until children terminate.
- Address space:
  - Child has duplicate of parent's address space, or
  - Child can have a program loaded onto it.
- UNIX examples:
  - fork system call creates new process and returns with a pid (0 in child, > 0 in the parent),
  - exec system call can be used after a fork to replace the process' memory space with a new program.