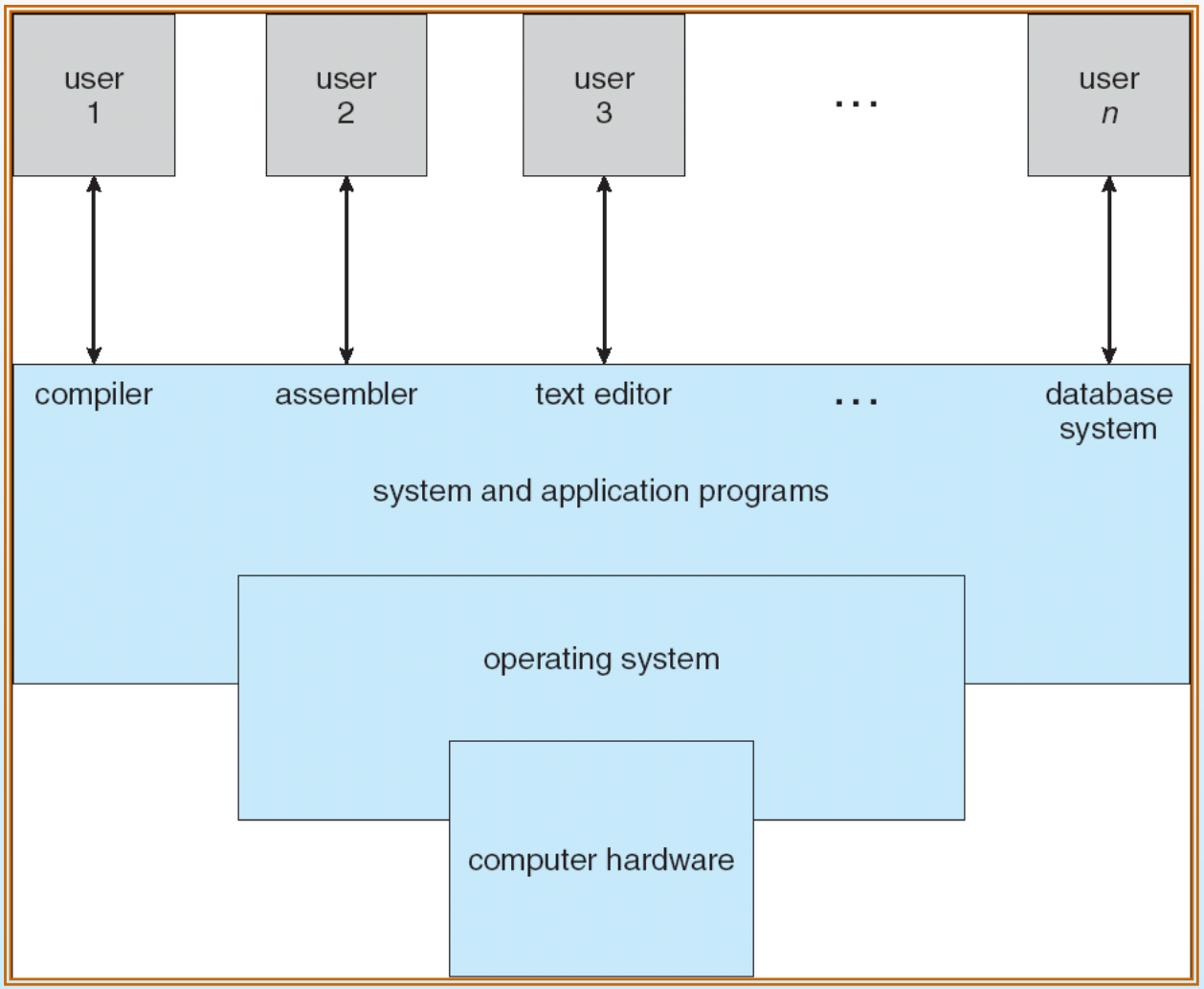
What is an Operating System?

- A program that acts as an intermediary between a user of a computer and the computer hardware.
- Operating system goals:
  - Execute user programs and make solving user problems easier.
  - Make the computer system convenient to use.
- Use the computer hardware in an efficient manner.

Computer System Structure

- Computer system can be divided into four components
  - Hardware – provides basic computing resources
    - CPU, memory, I/O devices
  - Operating system
    - Controls and coordinates use of hardware among various applications and users
  - Application programs – define the ways in which the system resources are used to solve the computing problems of the users
    - Word processors, compilers, web browsers, database systems, video games
  - Users
    - People, machines, other computers

Four Components of a Computer System



Operating System Definition

- OS is a **resource allocator**
  - Manages all resources
  - Decides between conflicting requests for efficient and fair resource use
- OS is a **control program**
  - Controls execution of programs to prevent errors and improper use of the computer

The picture can't be displayed.

Chapter 3: Processes

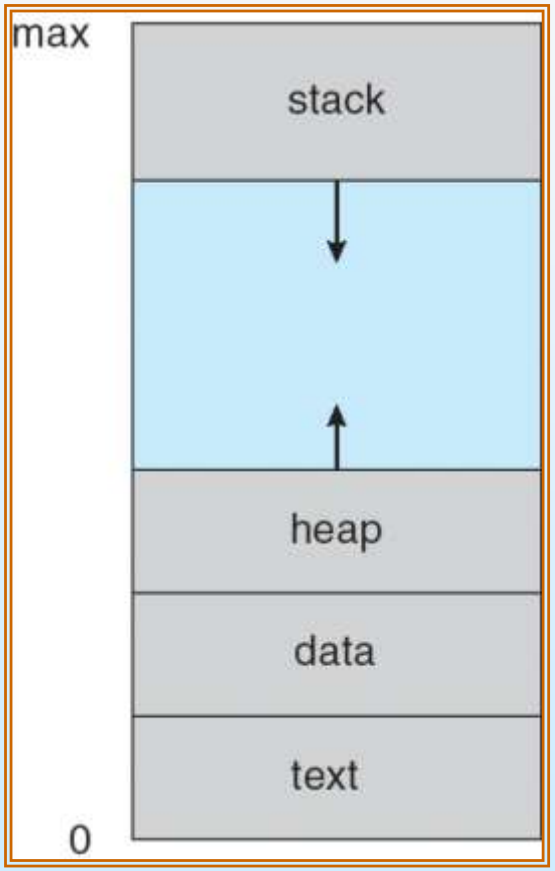
Chapter 3: Processes

- Process Concept
- Process Scheduling
- Operations on Processes
- Cooperating Processes
- Interprocess Communication
- Communication in Client-Server Systems

Process Concept

- Program in execution is called a process
- Program—bunch of instructions. When we execute the program it becomes a process
- A process includes:
  - program counter
  - Stack: **Stacks** are fundamental to function calls. Each time a function is called it gets a new **stack** frame
  - Heap: a **heap** is an area of pre-reserved computer main storage (memory) that a program **process** can use to store data in some variable amount that won't be known until the program is running
  - data section

Process in Memory



Process in Memory ..

- Stack: contains the temporary data such as method/function parameters, return address and local variables
- Heap: Dynamically allocated run time memory
- Text: Contains current activity –program counter and register contents
- Data: Global and static variables

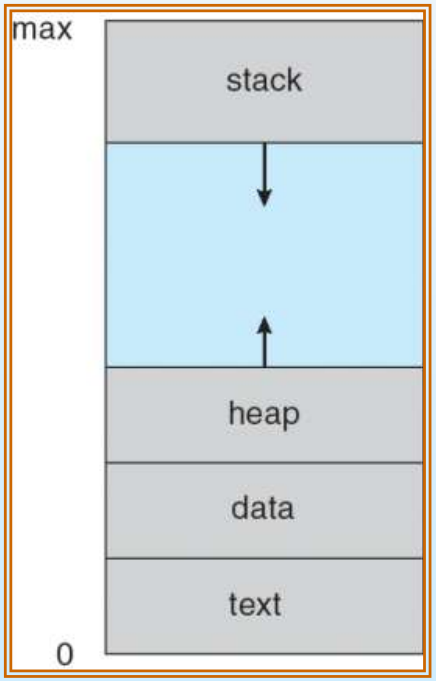
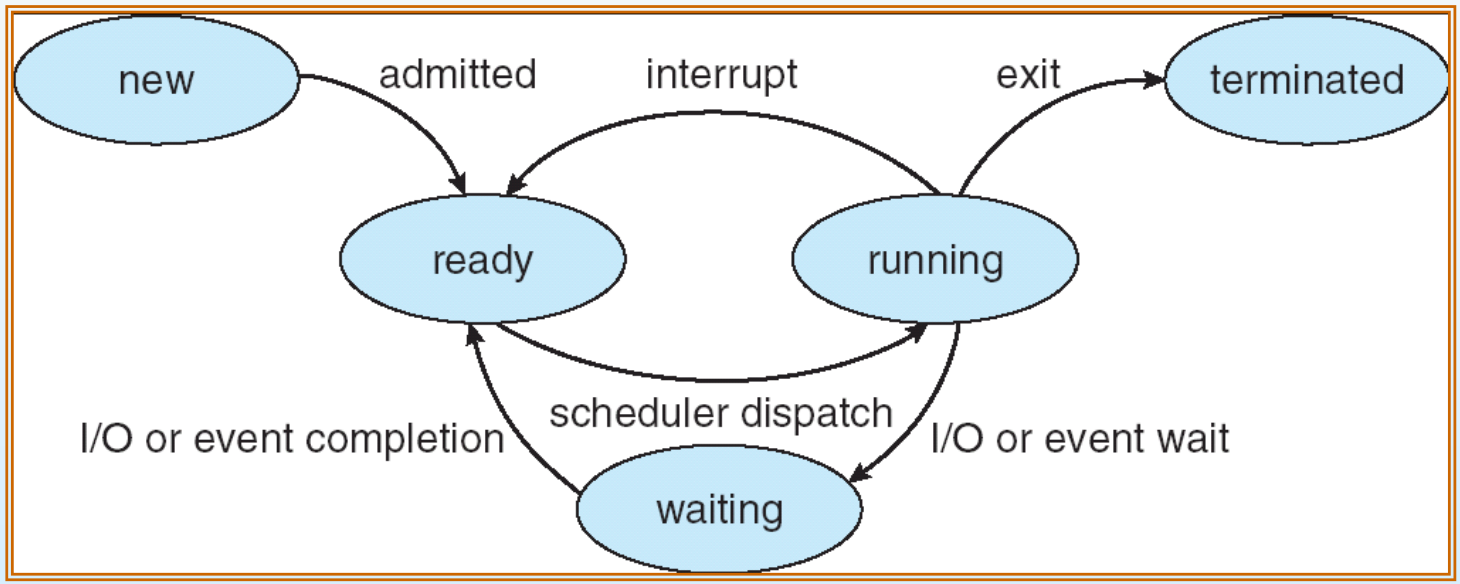
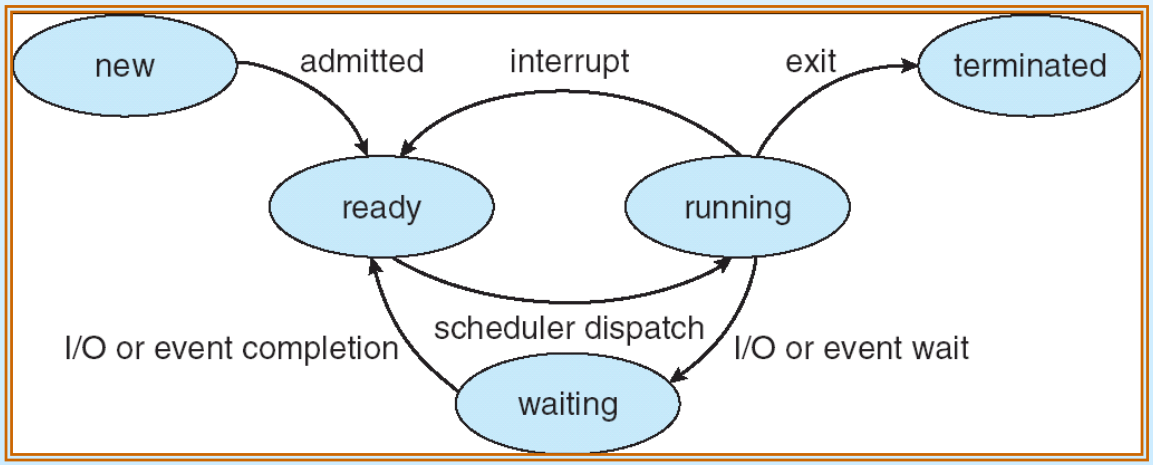


Diagram of Process State



Process State

- As a process executes, it changes *state*
  - **new**: The process is being created
  - **running**: Instructions are being executed
  - **waiting**: The process is waiting for some event to finish(Eg:I/O to complete)
  - **ready**: The process is waiting to be assigned to a process
  - **terminated**: The process has finished execution



Process Control Block (PCB)

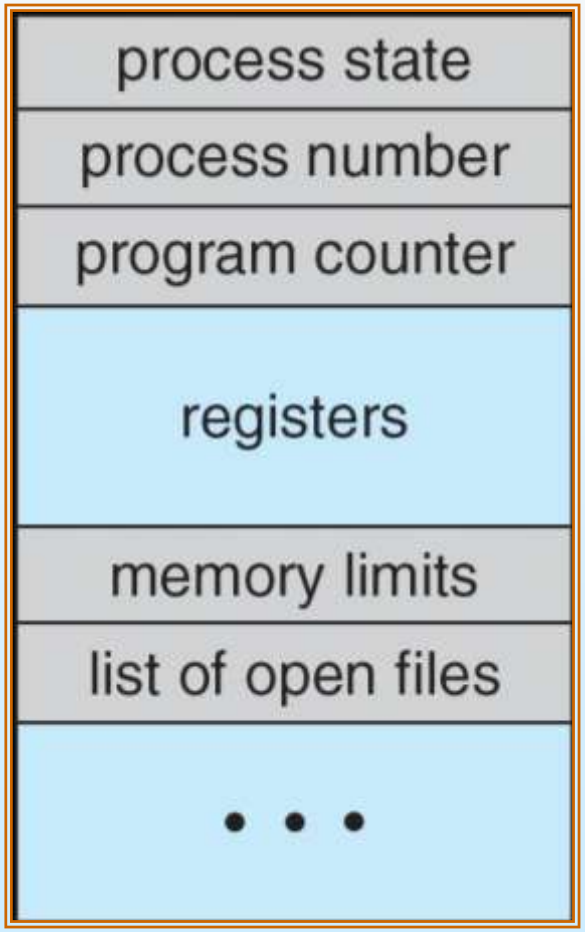
Information associated with each process. Stored in kernel of OS when process goes to waiting state and reloaded into hardware registers when process ready to run.

- Process ID: Unique ID given to each process in OS
- Pointer: A pointer to the parent process
- Process Privileges: Privileges to allow/disallow access to system resources.
- Process state: Current state as to if running, waiting, ready, etc
- Program counter: Pointer to address of next instruction to be executed

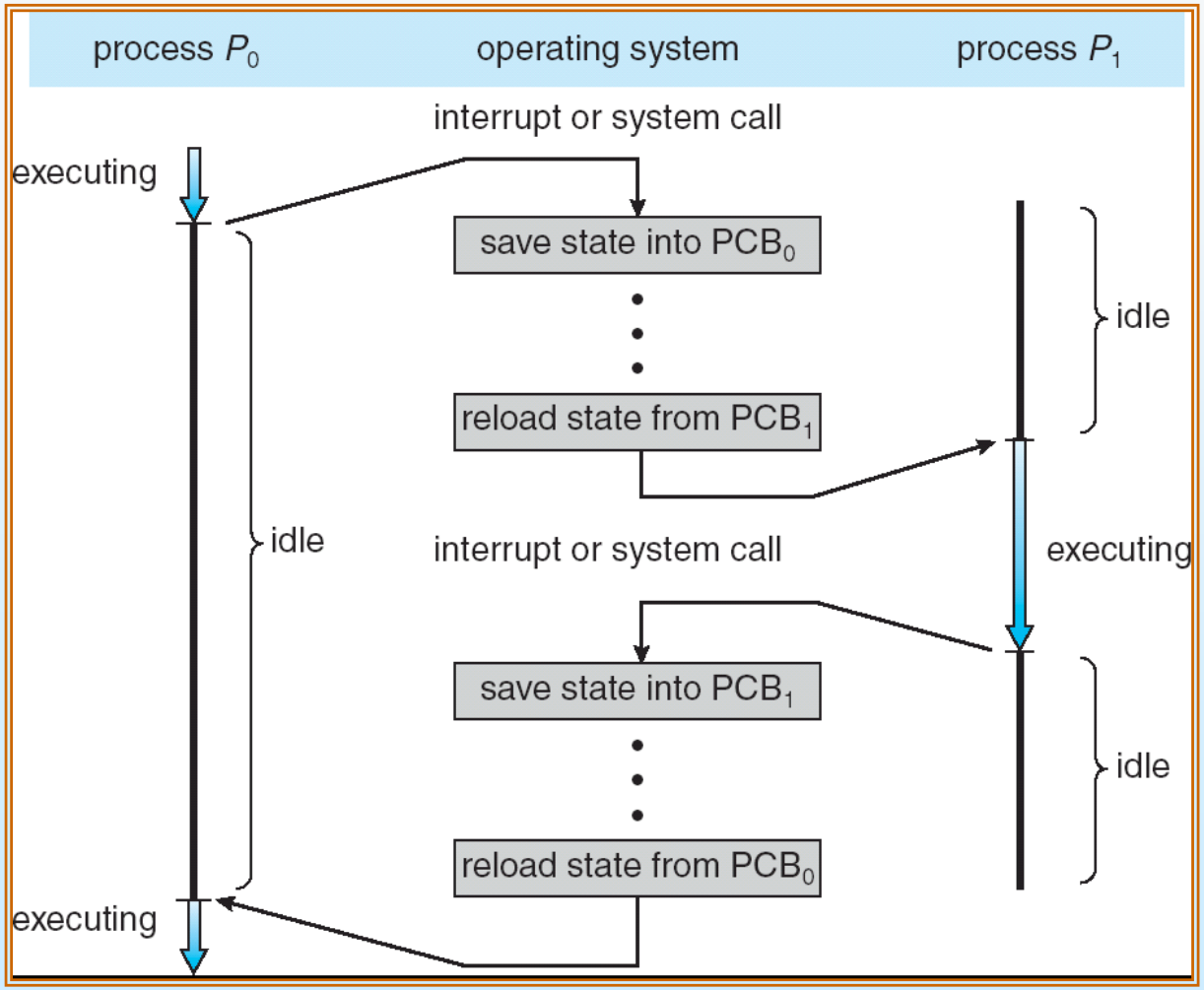
PCB continued ….

- CPU scheduling information: Process priority or any other scheduling information to schedule the process.
- CPU registers: CPU register information where process stores information while running
- Memory-management information: OS dependent information on page table, memory limits, segment table ,etc
- Accounting information:This includes the amount of CPU used for process execution, time limits, execution ID etc
- I/O status information: This includes a list of I/O devices allocated to the process.

Process Control Block (PCB)



CPU Switch From Process to Process



Process Scheduling

The **process scheduler** is the component of the operating system that decides whether the currently running process should continue to run and, if not, which process should run next.

Scheduler required when

- The current process goes from the *running* to the *waiting* state (due to I/O or interrupt)
- The current process terminates.
- A timer interrupt causes the scheduler to move process from the *running* to the *ready* state.
- An I/O operation is complete and hence the process should move from the *waiting* to the *ready* state.
- The scheduler may preempt the currently running process and run another process

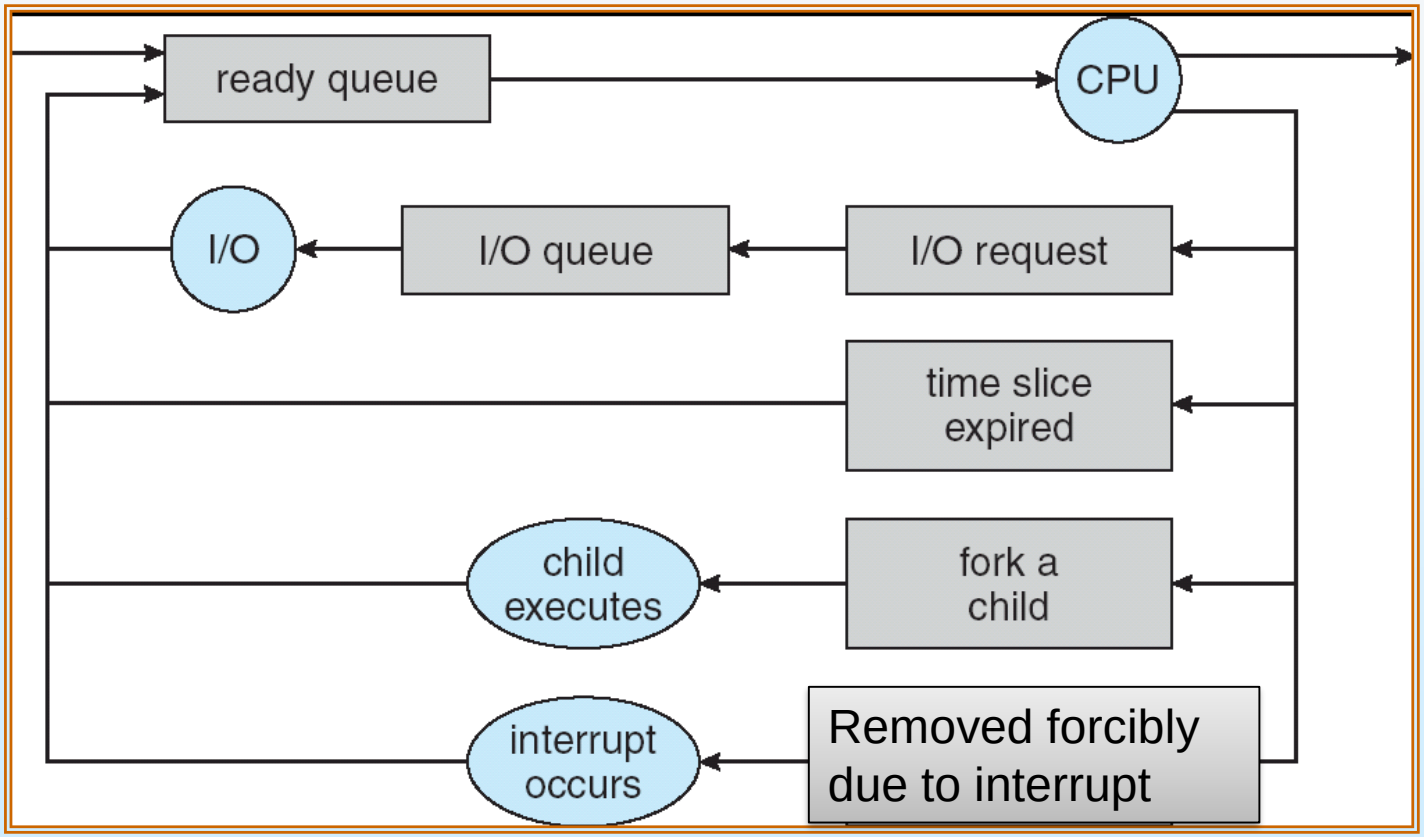
Process Scheduling Queues

All PCB's are kept in process scheduling queues. The OS maintains one queue each for each of the process state's. When the process state is changed, the process joins the new state queue. Types of queues:

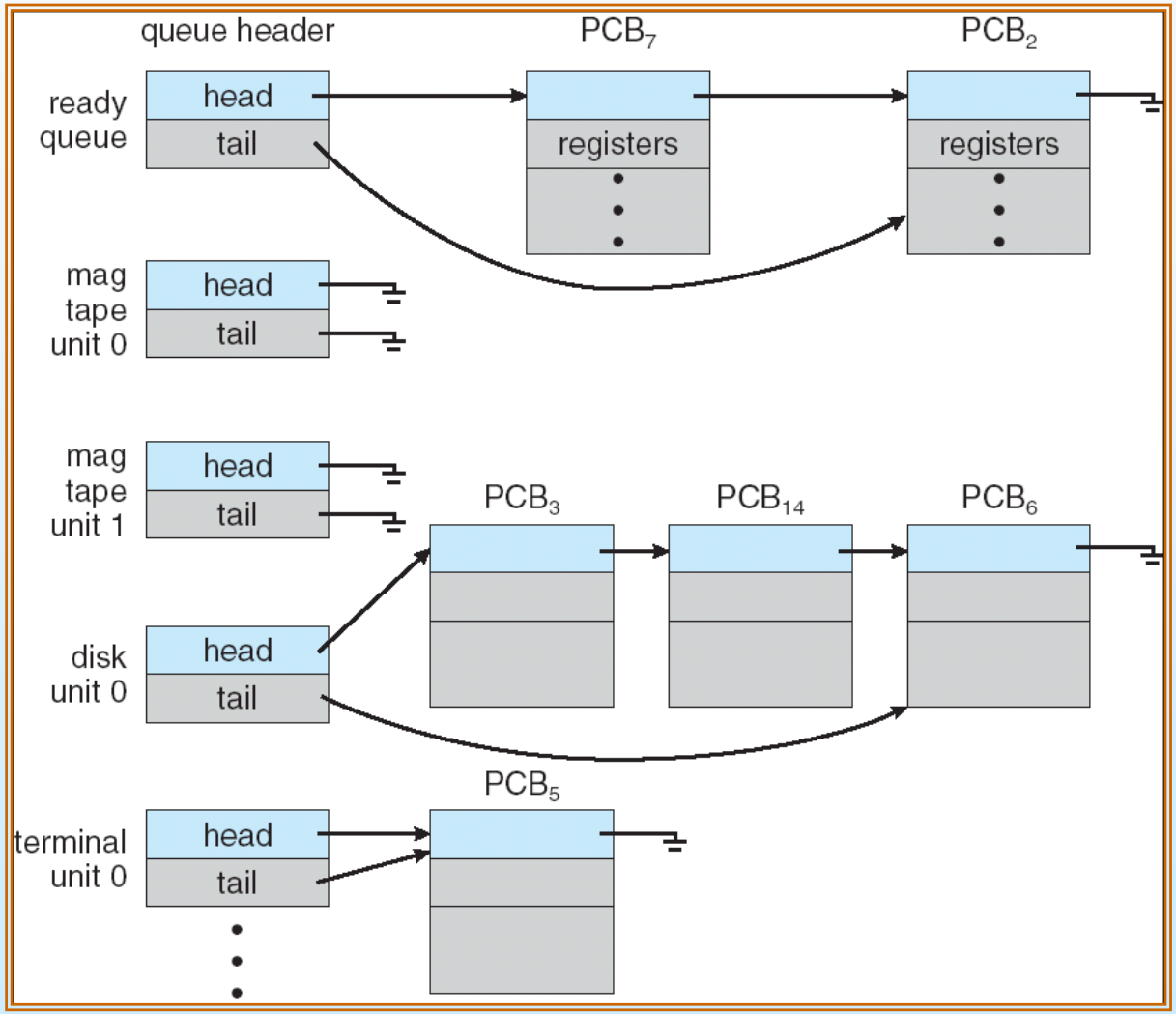
- **Job queue** – set of all processes in the system
- **Ready queue** – set of all processes residing in main memory, ready and waiting to execute
- **Device(waiting) queues** – set of processes waiting for an I/O device

Processes migrate among the various queues

Representation of Process Scheduling



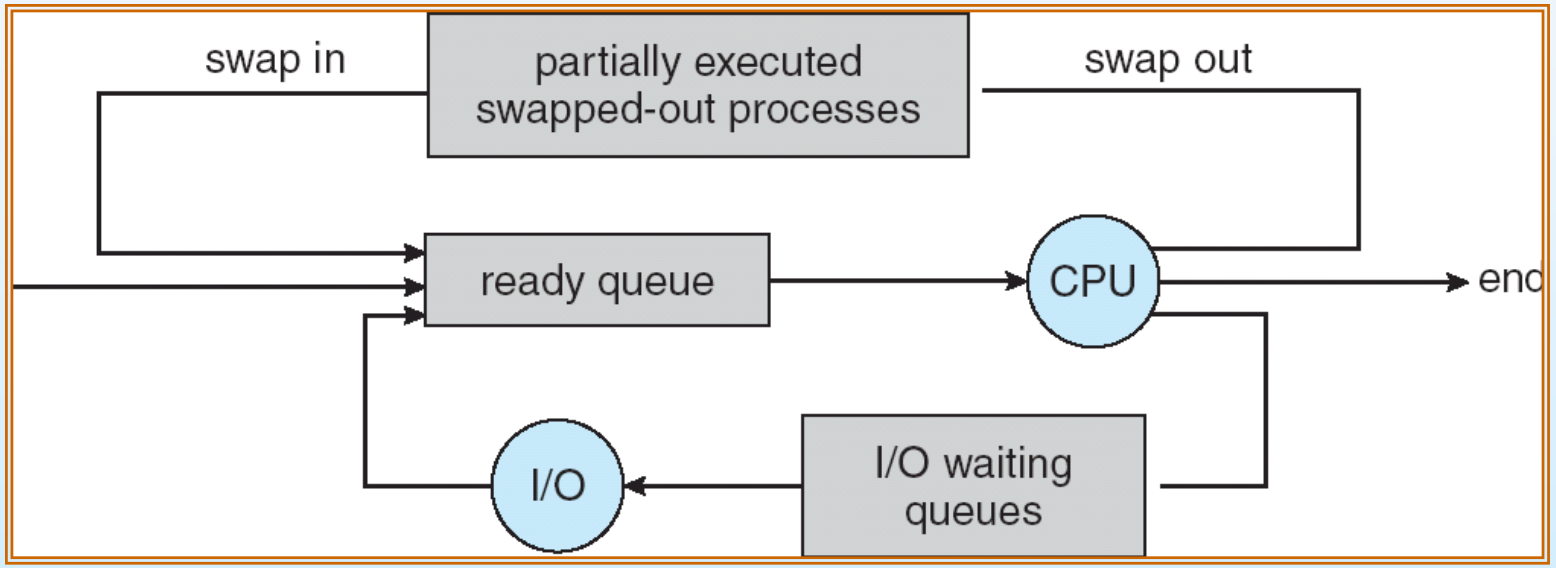
Ready Queue And Various I/O Device Queues



Schedulers

- **Long-term scheduler** (or job scheduler) – selects processes from pool on mass storage device that should be brought into the ready queue. Loads into memory. (invoked in mins)
  - provides balanced mix of I/O and processor bound jobs.
- **Short-term scheduler** (or CPU scheduler or dispatcher) – selects which process should be executed next and allocates CPU.
  - Faster than long term scheduler. (invoked in milliseconds)
- Medium Term scheduler: Used when you might want to swap a process waiting for I/O to secondary storage to make place for other processes. Done sometimes to improve the process mix

Addition of Medium Term Scheduling



Schedulers (Cont.)

- 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

Context Switch

A context switch is the mechanism to store and restore the state or context of a CPU in PCB so that a process execution can be resumed from the same point later

Context-switch time is overhead; the system does no useful work while switching. To avoid the amount of context switching time, some hardware systems employ two or more sets of processor registers

Time dependent on hardware support