Chapter 3: Processes

Chapter 3: Processes

- **Process Concept**
- Process Scheduling
- Operations on Processes
- Interprocess Communication
- Examples of IPC Systems
- Communication in Client-Server Systems

Objectives

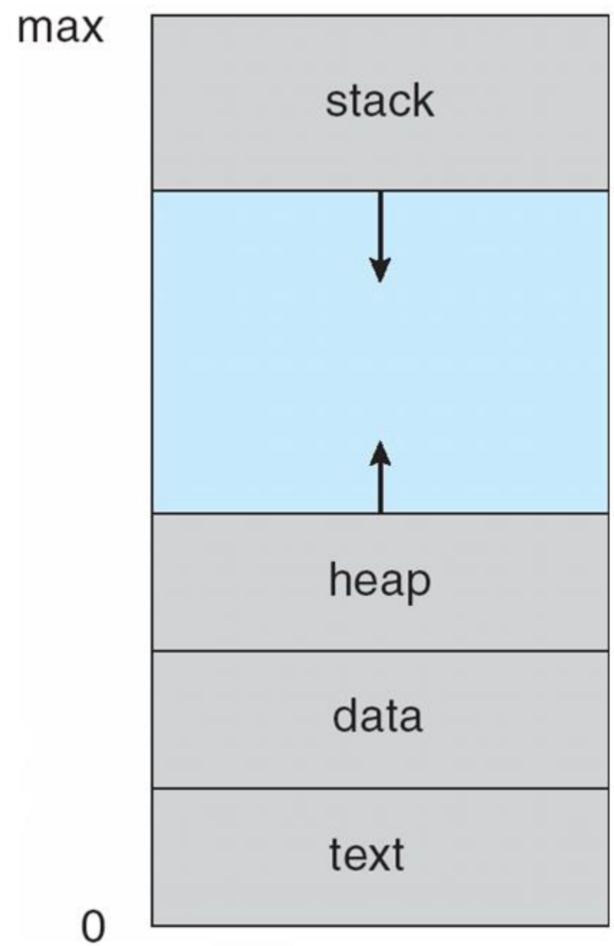
- To introduce the notion of a process -- a program in execution, which forms the basis of all computation
- To describe the various features of processes, including *scheduling*, *creation and termination*, and *communication*
- To explore interprocess communication using shared memory and message passing
- To describe communication in client-server systems

Process Concept (Cont.)

- Textbook uses the terms ***job*** and ***process*** almost interchangeably
- **Process** – a program in execution; process execution must progress in sequential fashion
- Program is ***passive*** entity stored on disk (**executable file**), process is ***active***
  - Program becomes process when executable file loaded into memory
- Execution of program started via GUI mouse clicks, command line entry of its name, etc

Process Address Space

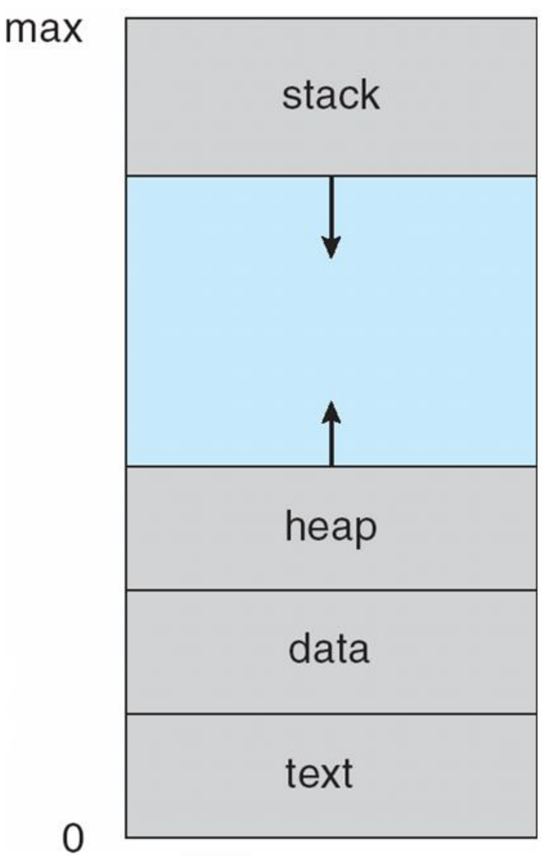
- A list of memory locations from some min (usually 0) to some max that a process can read and write.



Process Concept

- Multiple parts
  - The program code, also called **text section**
  - Current activity including **program counter**, processor registers
  - **Stack** containing temporary data
    - Function parameters, return addresses, local variables
  - **Data section** containing global variables
  - **Heap** containing memory dynamically allocated during run time
- One program can be several processes
  - Consider multiple users executing the same program (Several email programs or web browsers)
  - They are separate processes with equivalent code segment (i.e. ***same text section***)
  - Program→ blueprint, process→an instance

Process in Memory



Process Control Block (PCB)

Information associated with each process

(Process represented in OS by PCB also called task control block)

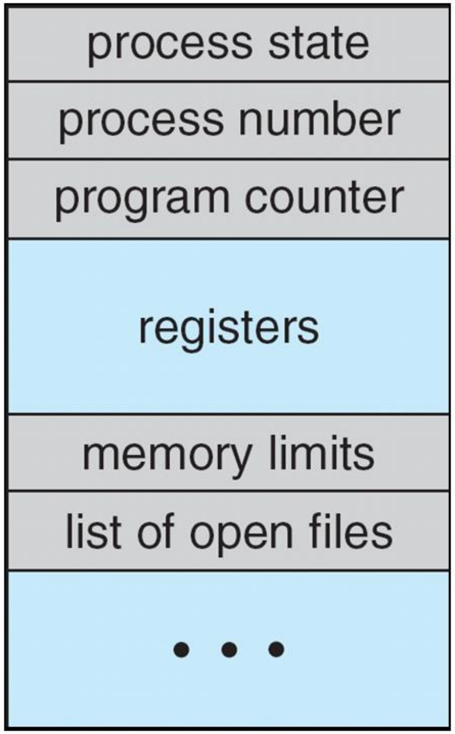
- PCB (execution context) is the data needed (process attributes) by OS to control process
  1. Process location information
  2. Process identification information
  3. Processor state information
  4. Process control information (mode)

Process Control Block (PCB)

Information associated with each process

(Process represented in OS by PCB also called **task control block**)

- **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



Role of the Process Control Block

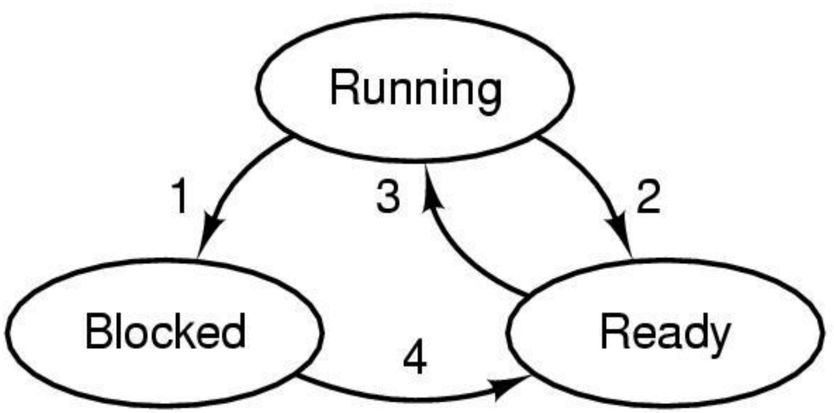
- The most important *data structure* in an OS
  - contains all of the information about a process that is needed by the OS
  - blocks are read and/or modified by virtually every module in the OS
  - defines the state of the OS
- Protection
  - a bug in a single routine could damage process control blocks, which could destroy the system's ability to manage the affected processes
  - a design change in the structure or semantics of the process control block could affect a number of modules in the OS

Process States (A Three-state Process Model)

- Let us start with three states:
  1) Running state –
     - the process that gets executes;
       its instructions are being executed.
  2) Ready state –
     - any process that is ready to be executed; the process is waiting to be assigned to a processor.
  3) Waiting/Blocked state –
     - when a process cannot execute until its I/O completes or some other event occurs.

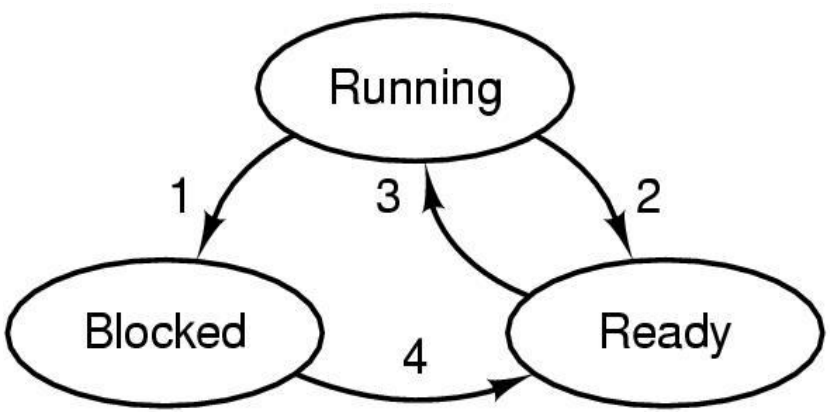
Process Transitions (1)

- Ready –> Running
  - When it is time, the dispatcher selects a new process to run.
- Running –> Ready
  - the running process has expired his time slot.
  - the running process gets interrupted because a higher priority process is in the ready state.



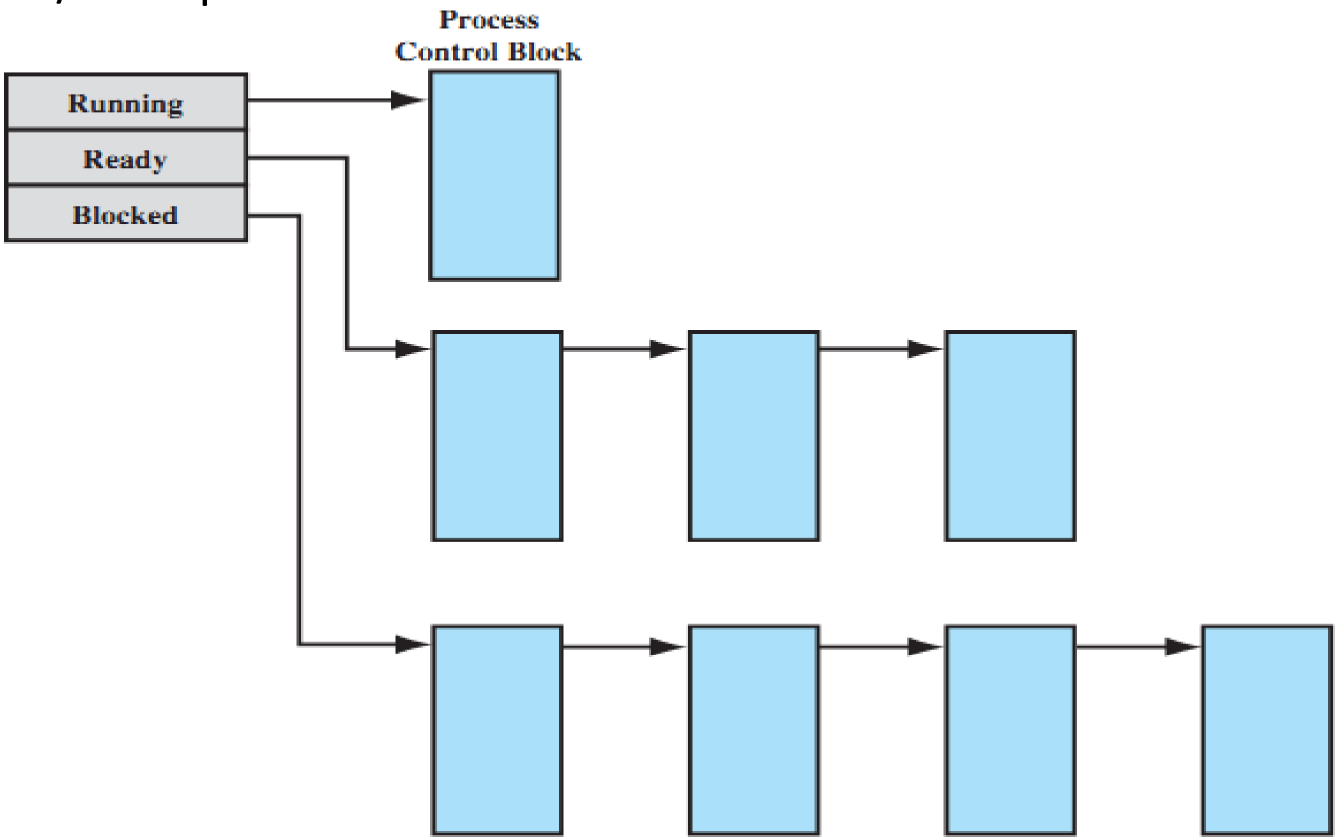
Process Transitions (2)

- Running –> Waiting (blocking)
  - When a process requests something for which it must wait:
    - a service that the OS is not ready to perform.
    - an access to a resource not yet available.
    - initiates I/O and must wait for the result.
    - waiting for a process to provide input.
- Waiting –> Ready
  - When the event for which it was waiting occurs.

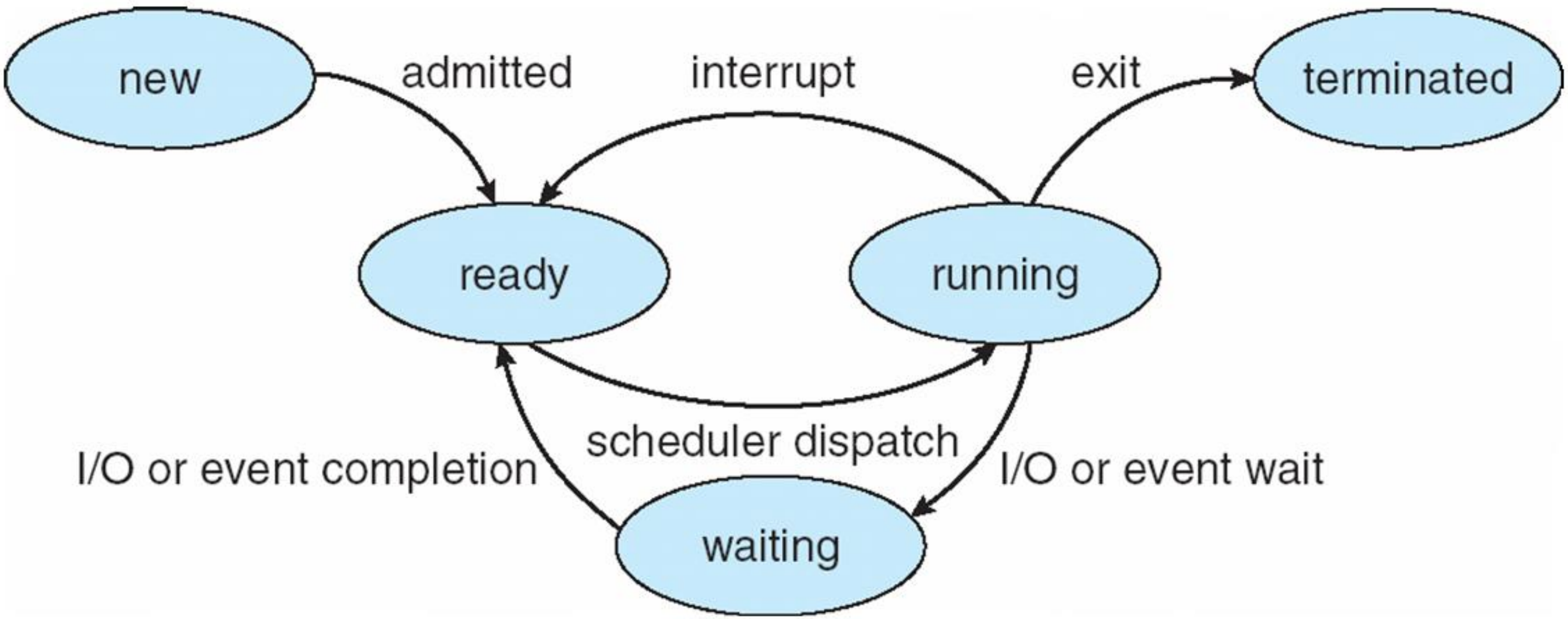


Process List Structures

- Three queues in three state process model:
  - Running Queue,
  - Ready Queue
  - Blocked/wait queue

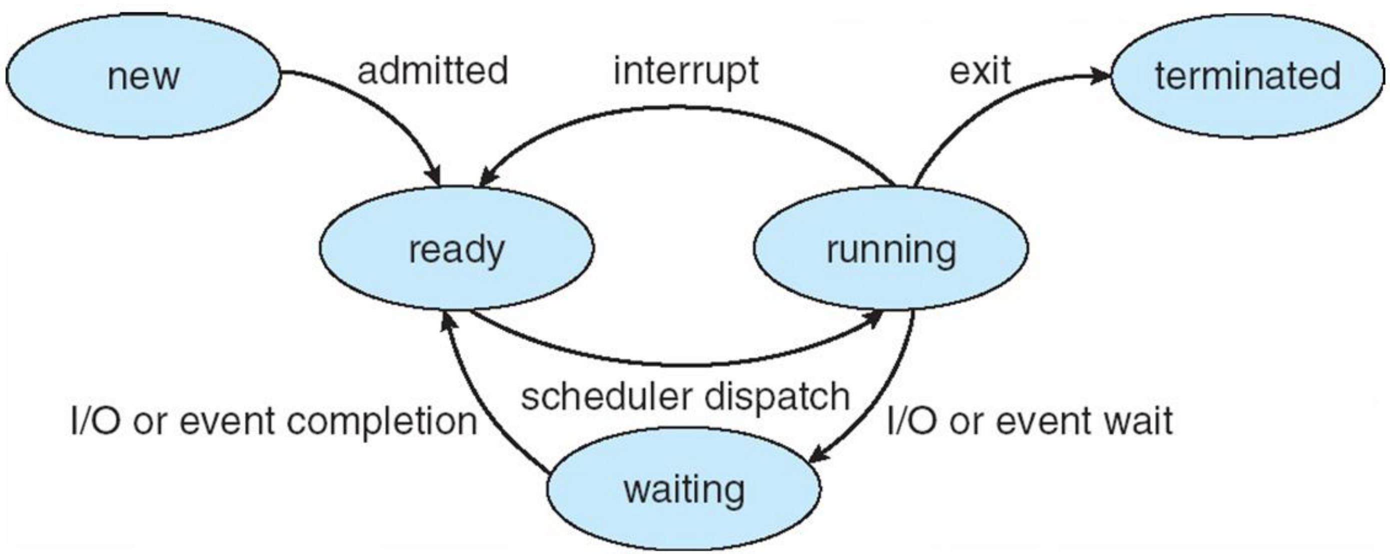


Five State Process Model



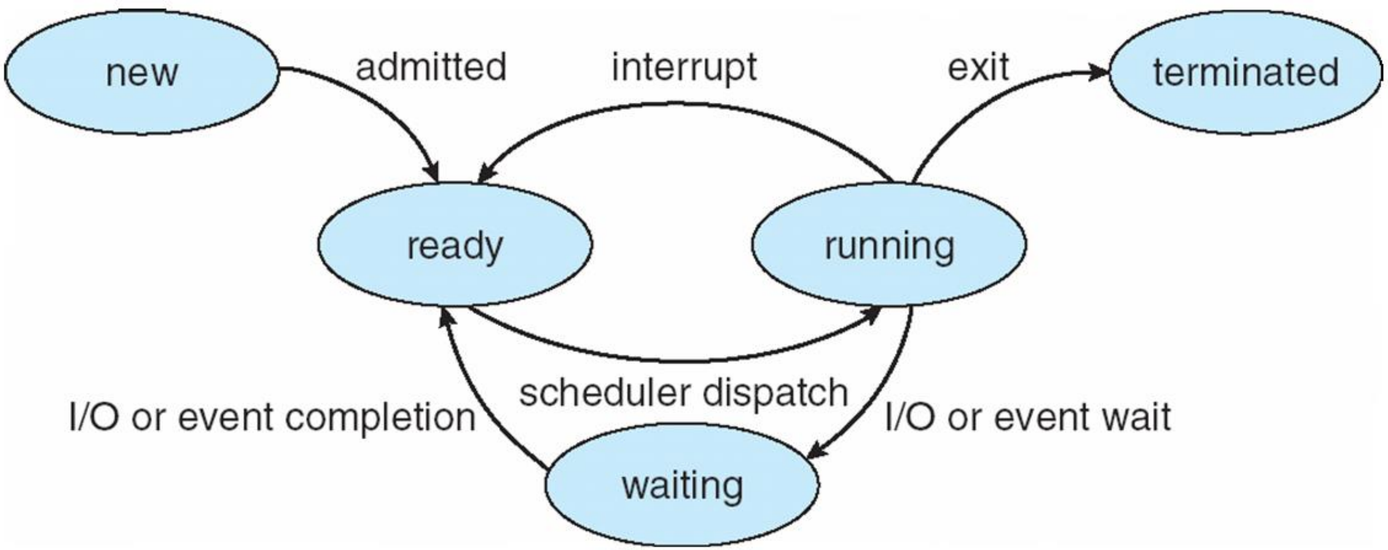
Five State Process Model

- **New state –**
  - OS has performed the necessary actions to create the process:
    - has created a process identifier.
    - has created tables needed to manage the process.
  - but has not yet committed to execute the process (not yet admitted):
    - because resources are limited.

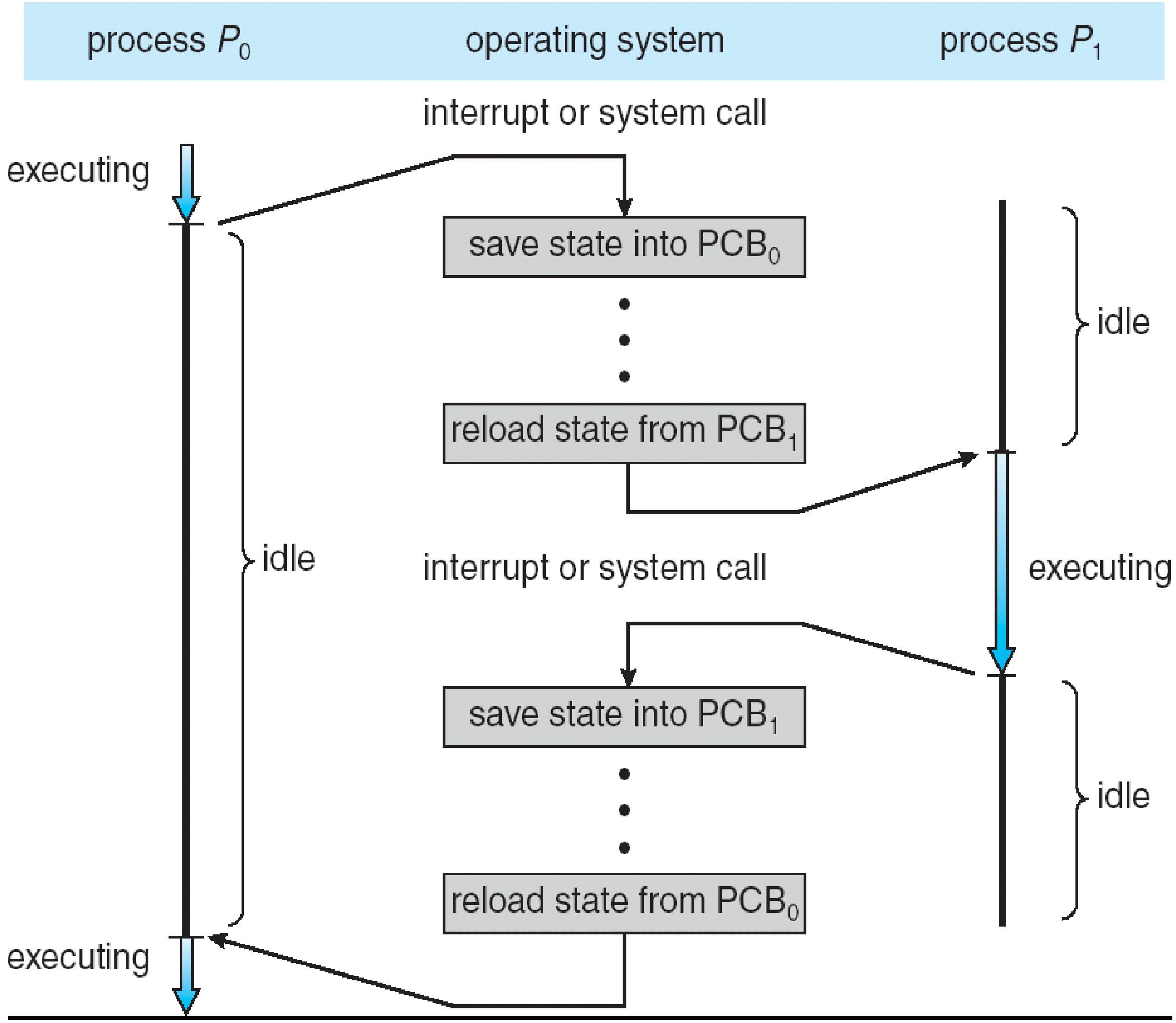


Five State Process Model

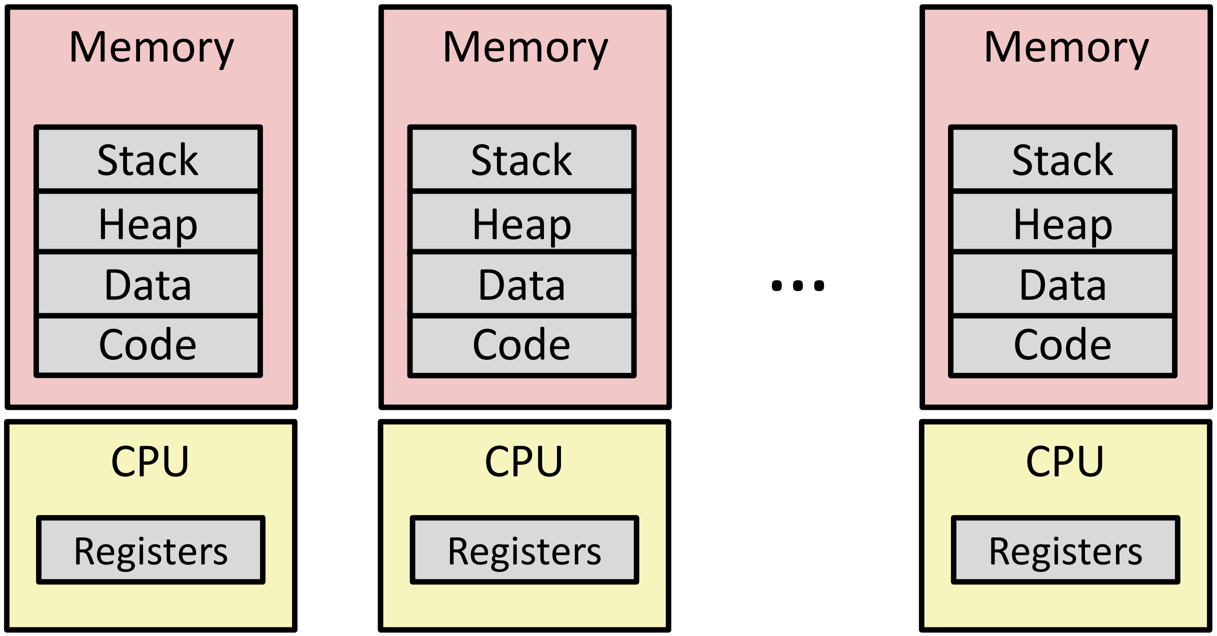
- **Terminated state –**
  - Program termination moves the process to this state.
  - It is no longer eligible for execution.
  - Tables and other info are temporarily preserved for auxiliary program –
- **Example:** accounting program that cumulates resource usage for billing the users.
- The process (and its tables) gets deleted when the data is no more needed.



CPU Switch From Process to Process

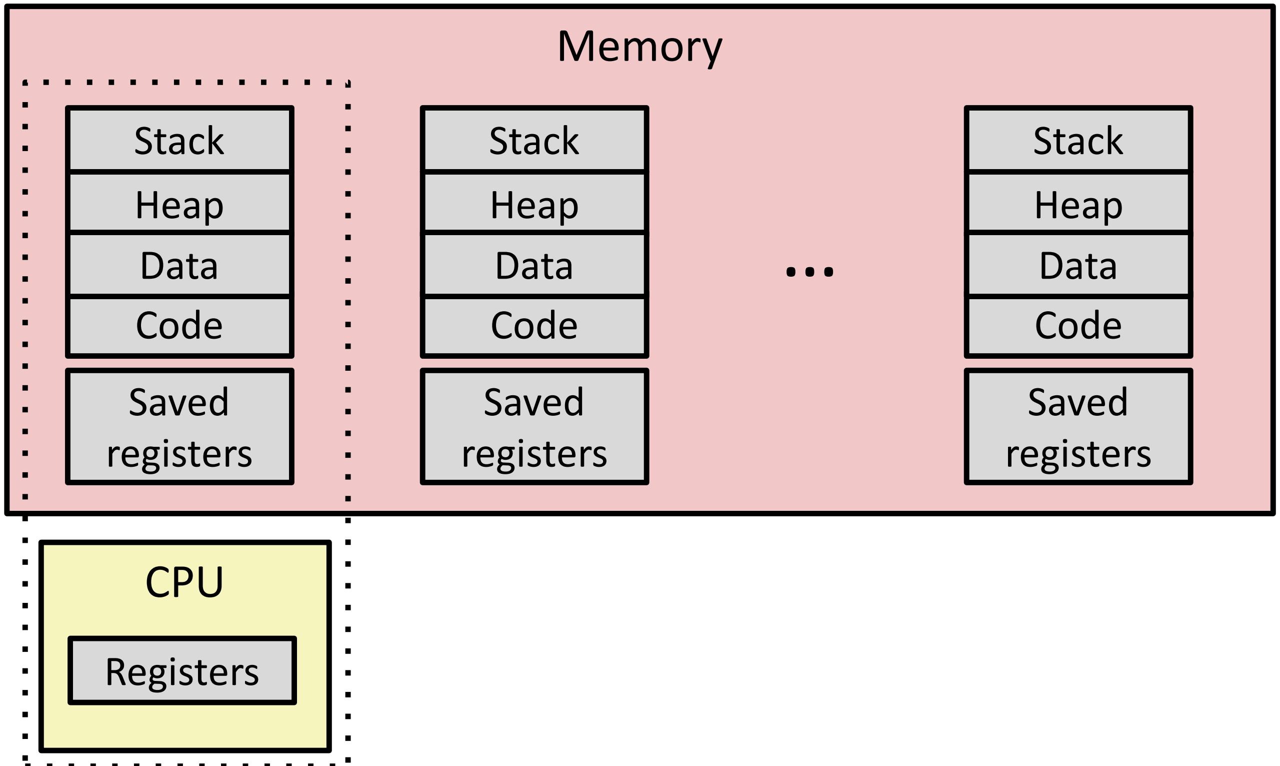


Multiprocessing: Illusion vs Reality



- Computer runs many processes simultaneously
  - Applications for one or more users
    - Web browsers, email clients, editors, …
    - Monitoring network & I/O devices

Multiprocessing: Illusion vs. Reality



- Single processor executes multiple processes *concurrently*
  - Process executions interleaved, CPU runs *one at a time*
  - Address spaces managed by virtual memory system (later in course)
  - *Execution context* (register values, stack, …) for other processes saved in memory

What is a 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
  - = process state, all register values, memory information
  - Save/restore contexts to/from PCBs when switching among processes
    - Known as **context switch**

- **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. **Typical speed is a few milliseconds**
    - **Depends on machine: memory speed, number of registers, load/save instructions**
    - Time dependent on hardware support

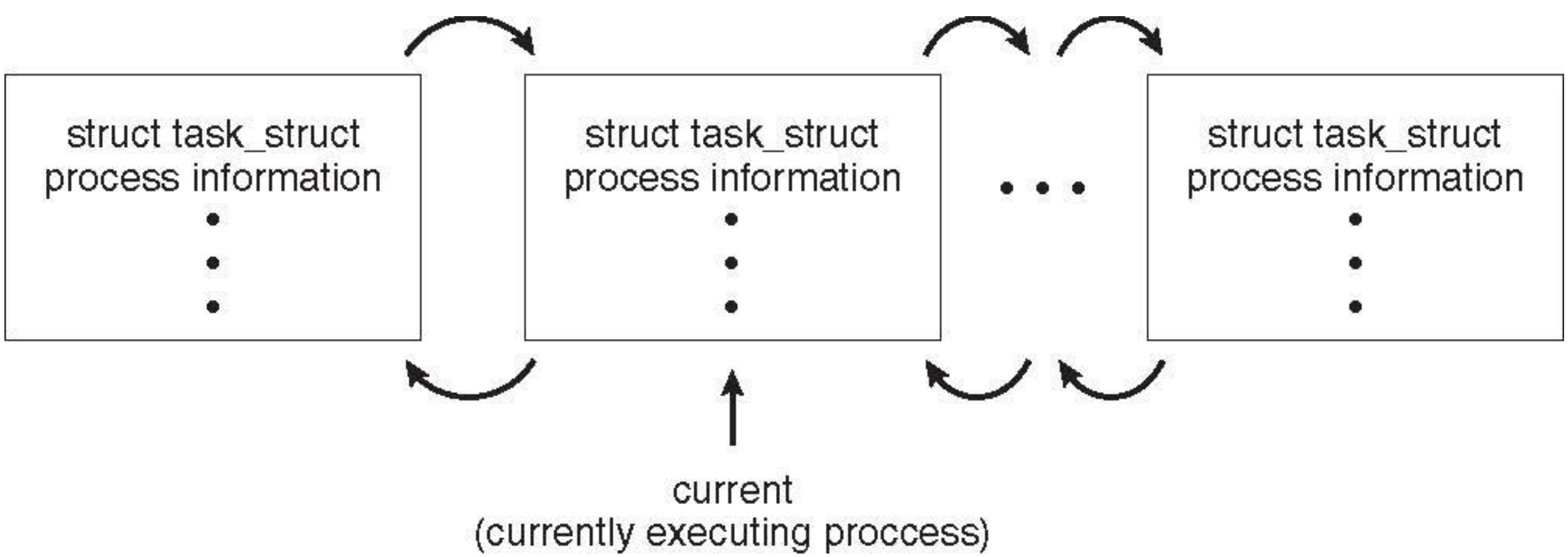
Context switch overhead

- Assume a processor runs for a total of 10 seconds, during which it performs 4 context switches, each of them takes one second.
  - What is the overhead (in %) due to context switches?

Process Representation in Linux

Represented by the C structure `task_struct`

```
pid t_pid; /* process identifier */
long state; /* state of the process */
unsigned int time_slice; /* scheduling information */
struct task_struct *parent; /* this process's parent */
struct list_head children; /* this process's children */
struct files_struct *files; /* list of open files */
struct mm_struct *mm; /* address space of this process */
```



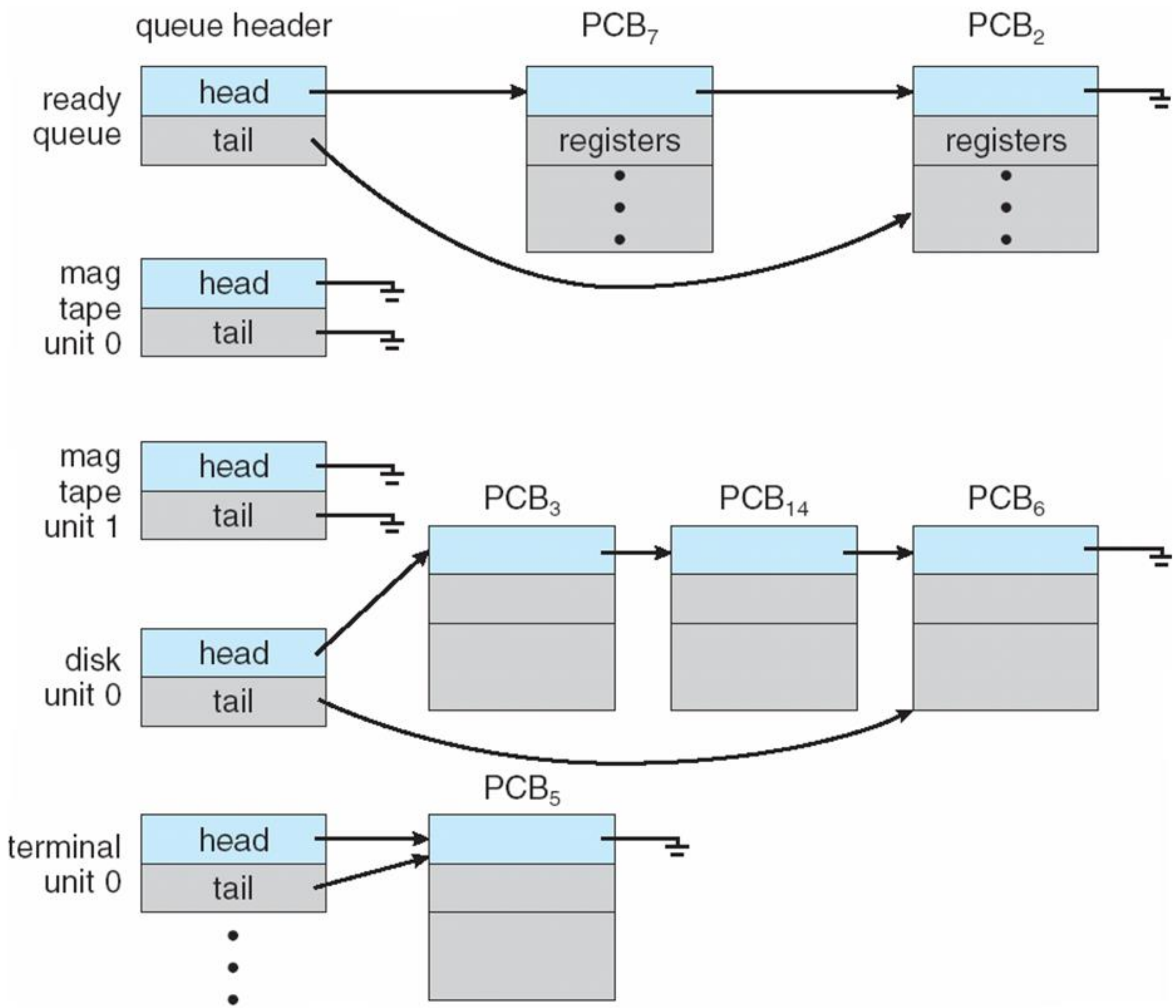
Chapter 3: Processes

- Process Concept
- **Process Scheduling**
- Operations on Processes
- Interprocess Communication
- Examples of IPC Systems
- Communication in Client-Server Systems

Process Scheduling

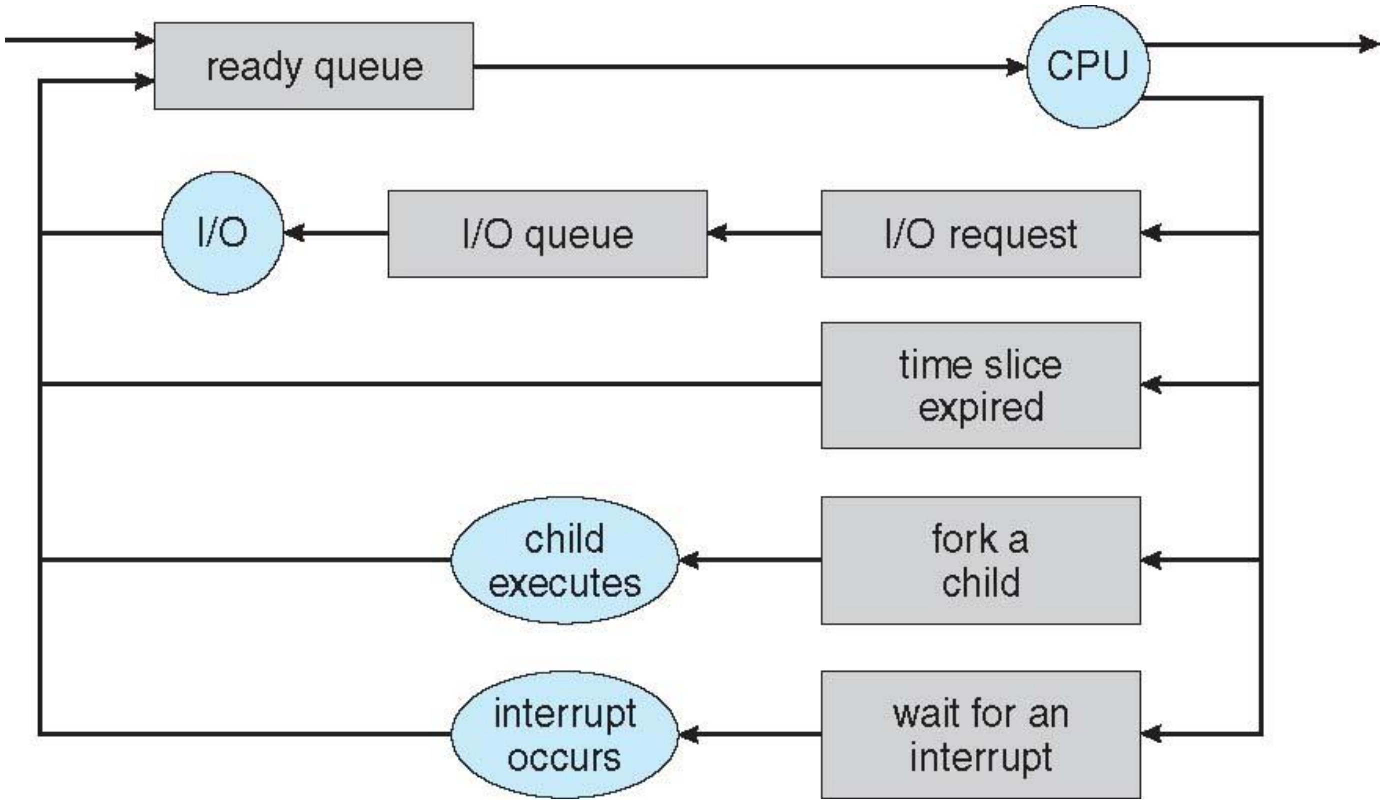
- **OS Objective:** to Maximize CPU use, quickly switch processes onto CPU for time sharing; so that users can interact with programs
- **Process scheduler** selects among available processes for next execution on CPU
  - Process scheduler = CPU scheduler + Job scheduler + other schedulers (e.g., for swapping etc.)
- 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
    - = Linked list of PCBs
  - **Device queues** – set of processes waiting for an I/O device
    - Each shared device has its associated device queue
  - Processes migrate among the various queues

Ready Queue And Various I/O Device Queues



Representation of Process Scheduling

- **Queueing diagram** represents queues, resources, flows

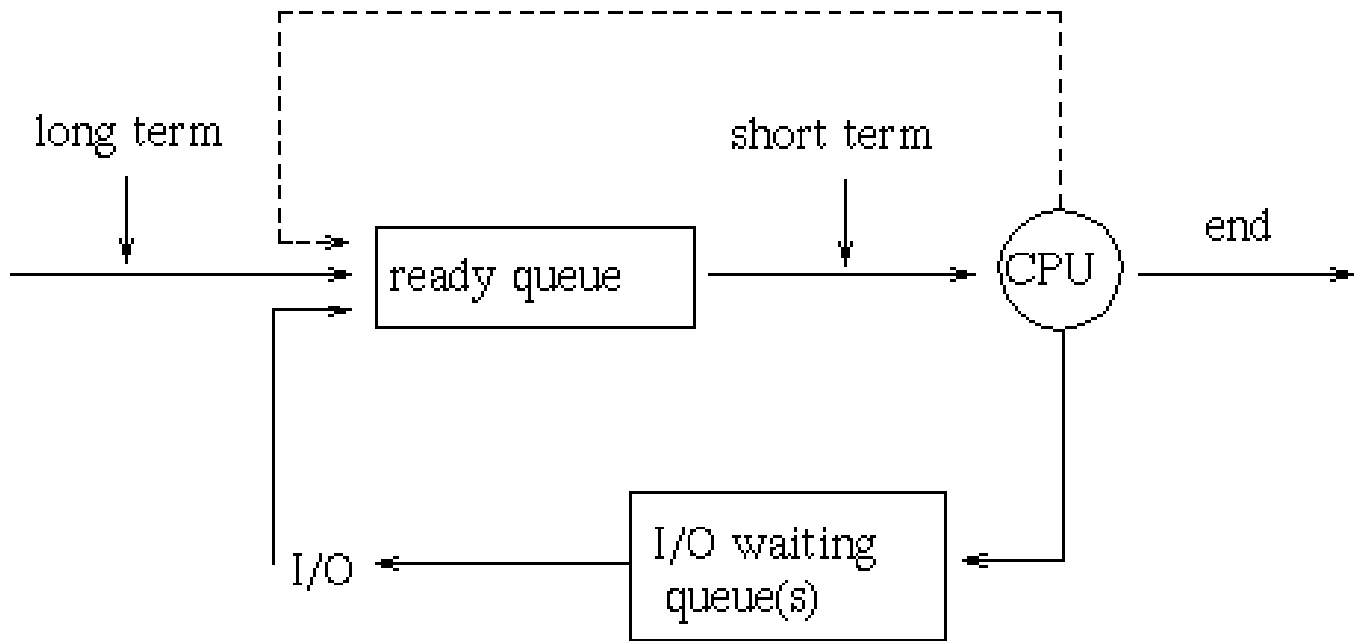


Schedulers

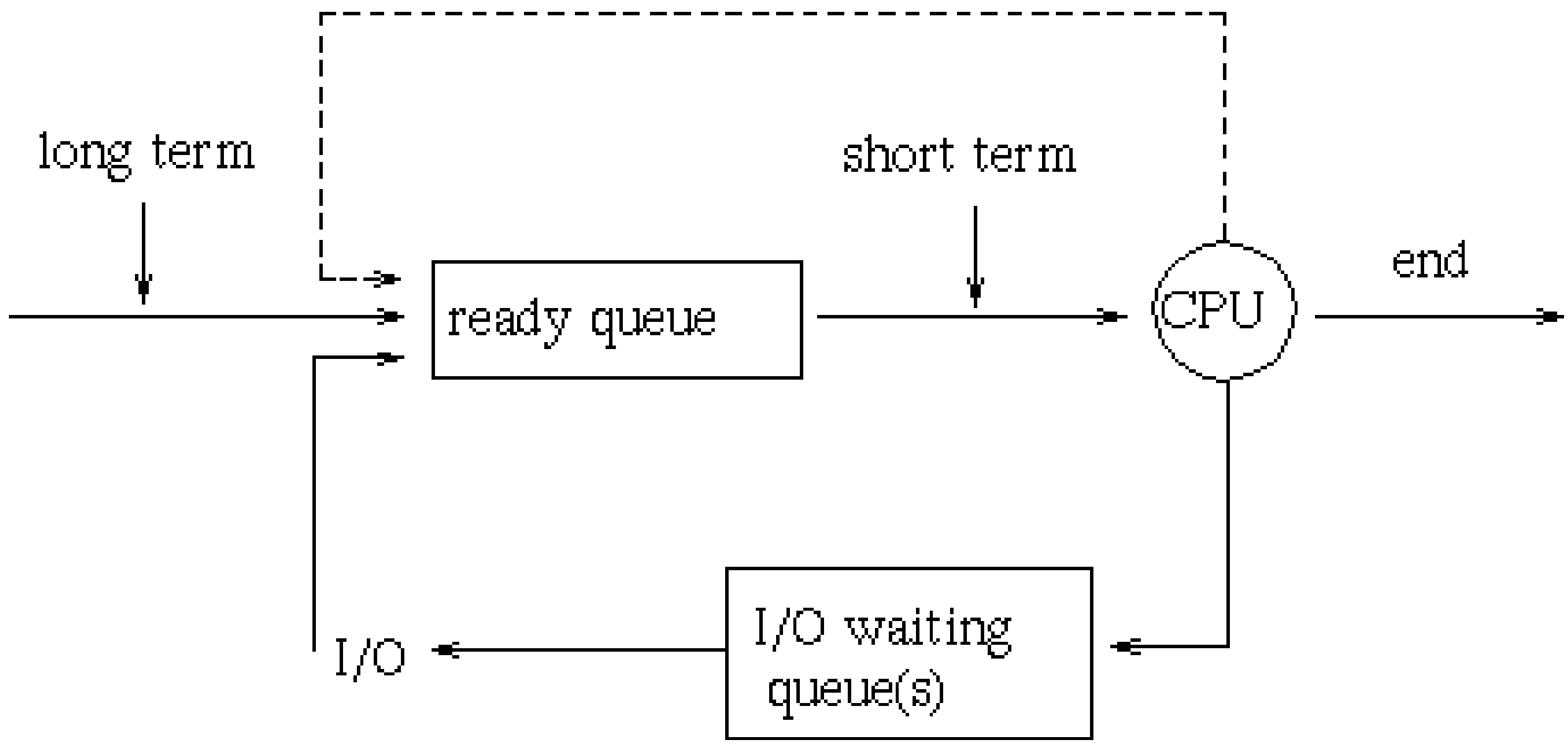
- **Short-term scheduler** (or **CPU scheduler/dispatcher**)
- **Long-term scheduler** (or **job scheduler**)
- **Medium-term scheduler**

Schedulers

- **Short-term scheduler** (or **CPU scheduler/dispatcher**) – 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)
  - Main objective is to increase CPU performance



Schedulers



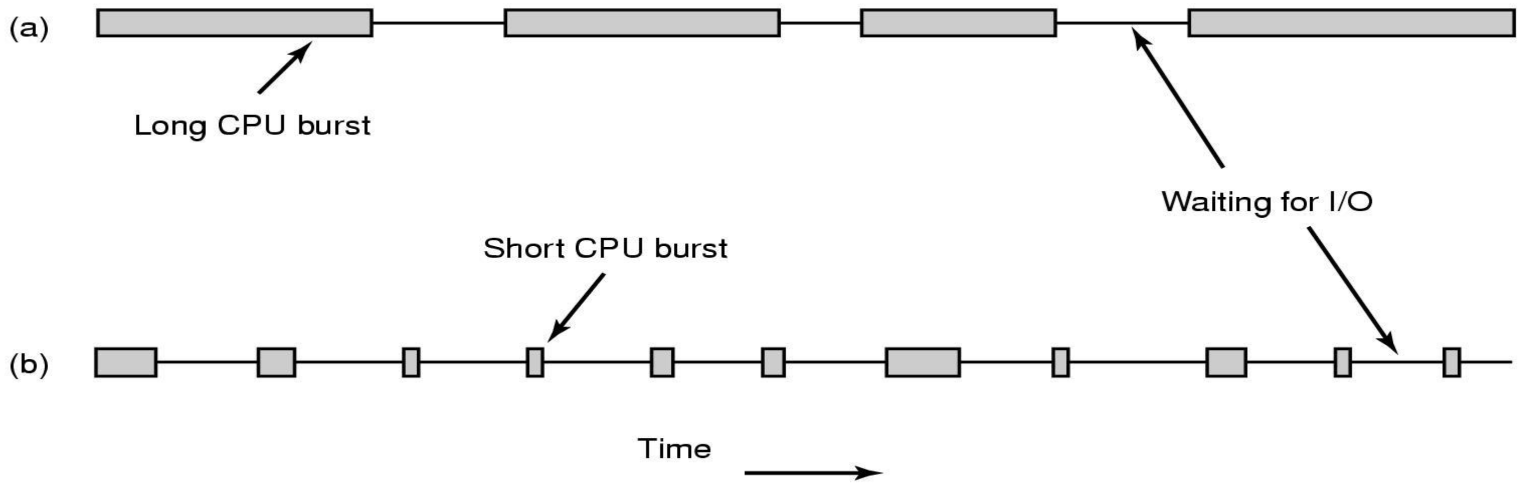
long term
short term
ready queue
CPU
end
I/O
I/O waiting
queue(s)

Schedulers (Cont'd)

- **Long-term scheduler** (or **job scheduler**) – selects which processes should be brought into the ready queue
  - Long-term scheduler is invoked infrequently (seconds, minutes) $\Rightarrow$ (may be slow)
  - The long-term scheduler controls the **degree of multiprogramming**
    - = Number of processes in memory (i.e., in the ready queue)
    - The **degree of multiprogramming** describes the maximum number of processes that a single-processor system can accommodate efficiently.
  - If the degree of multiprogramming is stable, then the average rate of process creation must be equal to the average departure rate of processes leaving the system, i.e., aver no. of process creation = aver no. of process departure.
  - Thus, invoked when a process leaves the system

Schedulers(cont.)

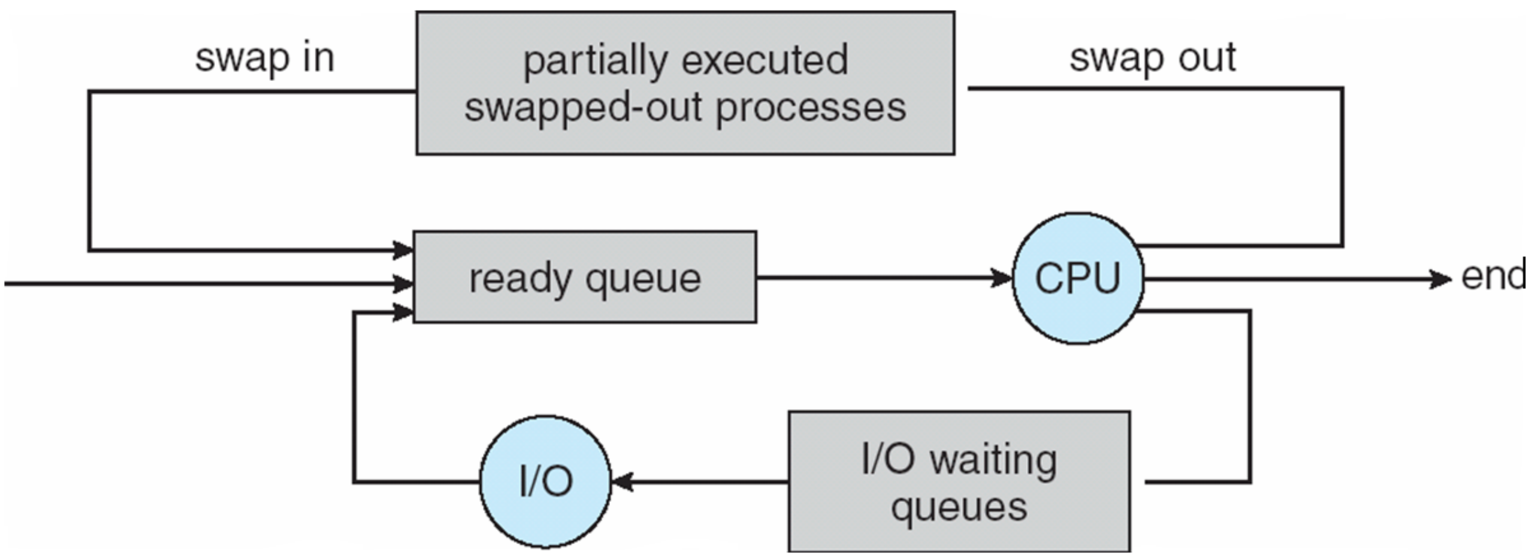
- Processes can be described as either:
  - **I/O-bound process** – spends more time doing I/O than computations, many short CPU bursts
    - The ready queue is almost always empty if all processes are I/O-bound
  - **CPU-bound process** – spends more time doing computations; few very long CPU bursts
    - The I/O queue is almost always empty if all processes are CPU-bound
- Long-term scheduler strives for good ***process mix***
- The primary objective of the job scheduler is to provide a balanced mix of jobs, such as I/O bound and processor bound.



Bursts of CPU usage alternate with periods of waiting for I/O. (a) A CPU-bound process. (b) An I/O-bound process.

Addition of Medium Term Scheduling

- **Medium-term scheduler** can be added if degree of multiprogramming needs to decrease
  - Remove process from memory, store on disk, bring back in from disk to continue execution: **swapping**
  - Swapping helps improve process mix
  - Also necessary when memory needs to be freed up



Chapter 3: Processes

- Process Concept
- Process Scheduling
- **Operations on Processes**
- Interprocess Communication
- Examples of IPC Systems
- Communication in Client-Server Systems