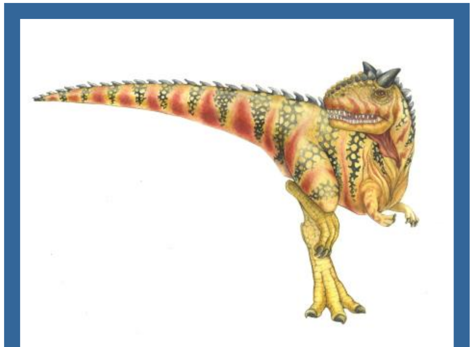
Xi'an Jiaotong-Liverpool University
西交利物浦大學

CPT104 - Operating Systems Concepts

# Operating Systems. Processes

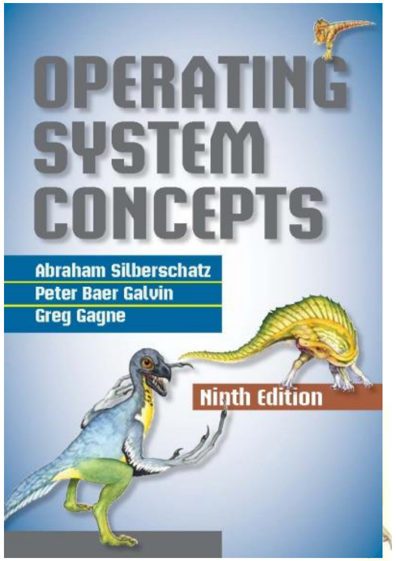


# Operating Systems Concepts

| Sequence | Method | Assessment Type (EXAM or CW) | Learning outcomes assessed (Use codes under learning outcomes.) | Duration | Week | % of final mark | Resit (Y/N/S)[1] |
|---|---|---|---|---|---|---|---|
| #001 | Final Exam | EXAM | ALL | 2 hour(s) | | 80 | S |
| #002 | Assessment task | CW | All | hour(s) | | 10 | |
| #003 | Assessment task | CW | All | hour(s) | | 10 | |
| #900 | Resit Exam | EXAM | ALL | 2 hour(s) | | 100 | |

Assessment I – week 7

Assessment II – week 12




**Module handbook and other important resources**

This folder provides access to the module handbook and other important resources.

# Contents

- Short introduction in Operating Systems concept
- Process Concept
- Process Scheduling
- Operations on Processes
- Inter-process Communication

# What is an Operating System?

- An interface between users and hardware - an environment "architecture”
- Allows convenient usage; hides the tedious stuff
- Allows efficient usage; parallel activity, avoids wasted cycles
- Provides information protection
- Gives each user a slice of the resources
- Acts as a control program.

# What is an Operating System?

Humans

Program Interface

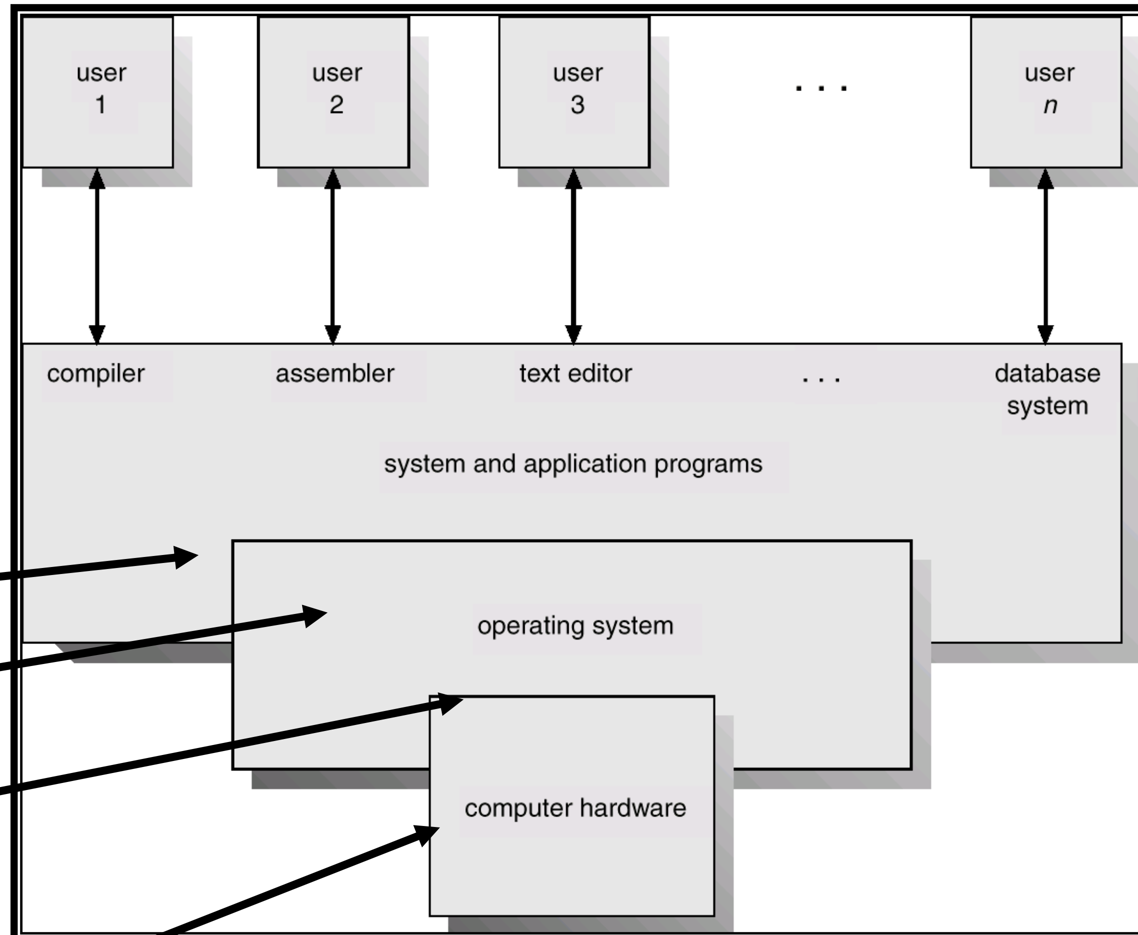
User Programs

O.S. Interface

Operating System

Hardware Interface/
Privileged Instructions

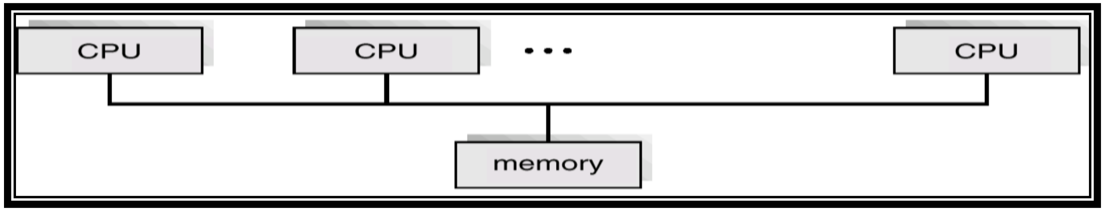
Disk/Tape/Memory



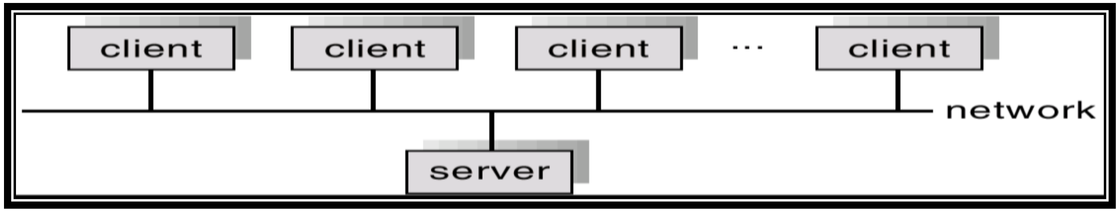
# Characteristics of Operating Systems

- **Time Sharing** - multiprogramming environment that's also interactive.

- **Multiprocessing** - Tightly coupled systems that communicate via shared memory. Used for speed improvement by putting together a number of off-the-shelf processors.




- **Distributed Systems** - Loosely coupled systems that communicate via message passing. Advantages include resource sharing, speed up, reliability, communication.




- **Real Time Systems** - Rapid response time is main characteristic. Used in control of applications where rapid response to a stimulus is essential.

# What is doing an Operating System?

O.S. carry out the most commonly required operations:

1. Process Management,
2. Memory management,
3. File System Management,
4. I/O System Management,
5. Protection and Security.

# PROCESS MANAGEMENT

*Starting and stopping programs and sharing the CPU between them.*

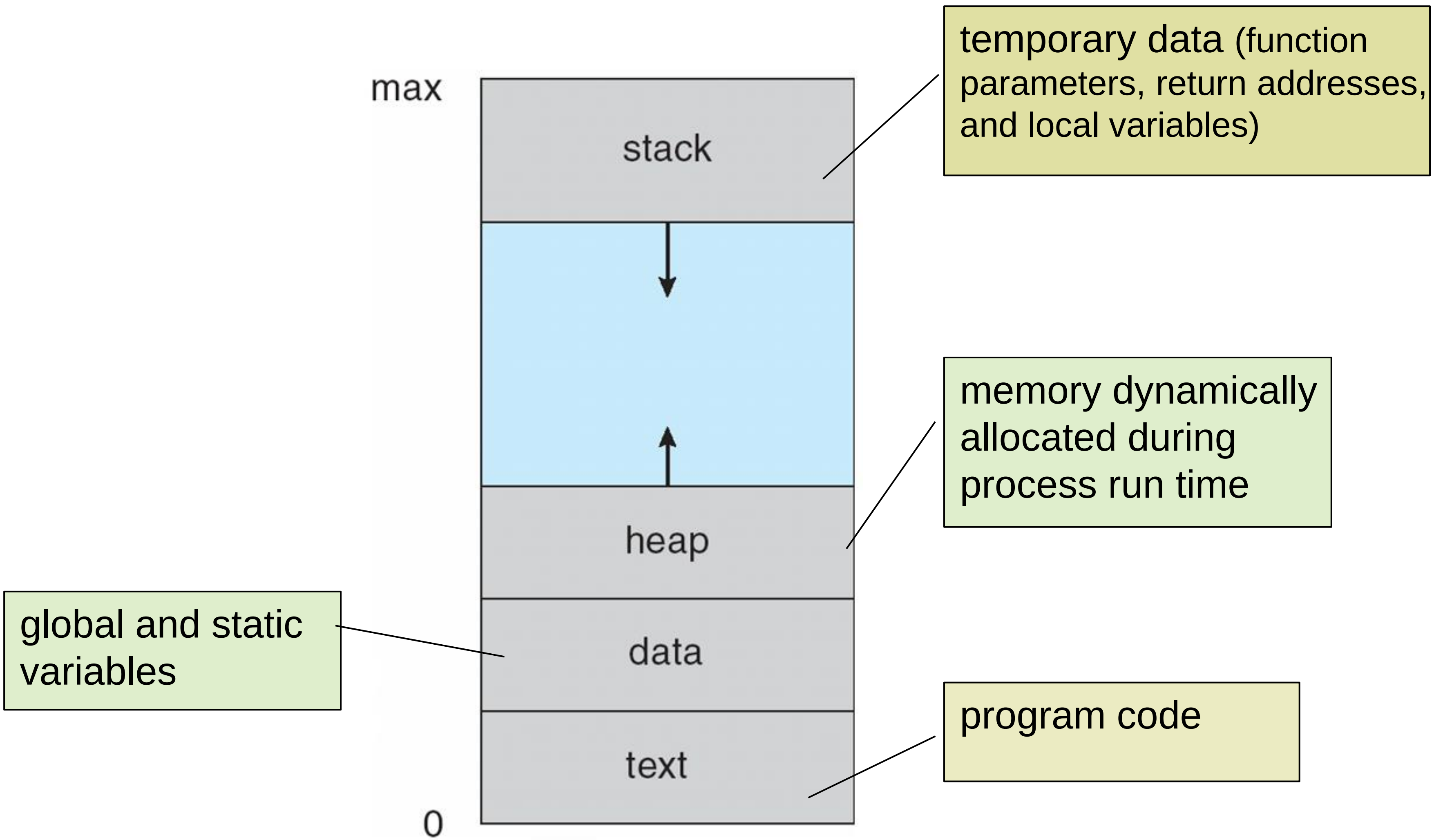
# Process

- Process Concept
- Process Scheduling
- Operations on Processes
- Inter-process Communication

# Process Concept

- **Process** = **a program in execution**; process execution must progress in sequential fashion
- An operating system executes a variety of programs:
  - Time-shared system – **user programs** or **tasks**
- A **process** is considered an '**active**' entity
- A **program** is considered to be a **'passive'** entity (stored on disk (executable file)).
- Program becomes process when executable file loaded into memory

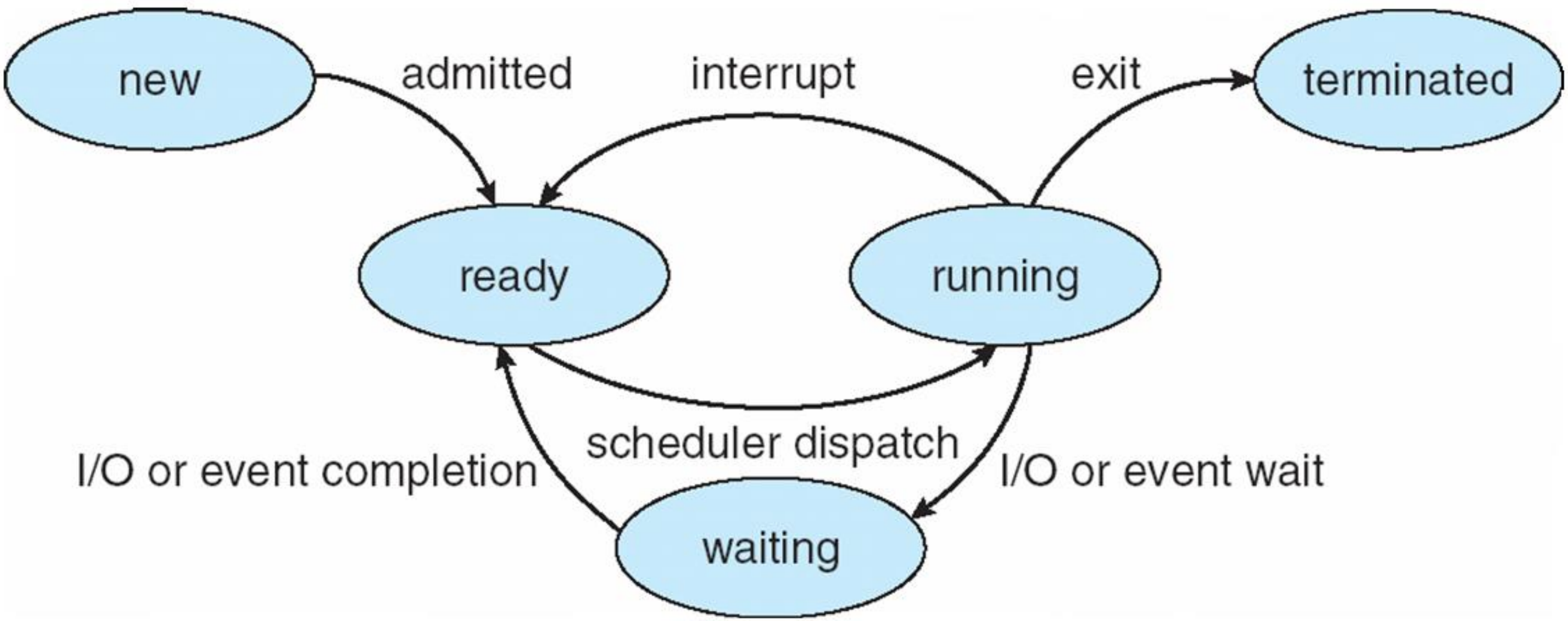
# Process in Memory



# Process State

- As a process executes, it changes **state.**
- The state of a process is defined in part by the **current activity of that process.**
  - **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

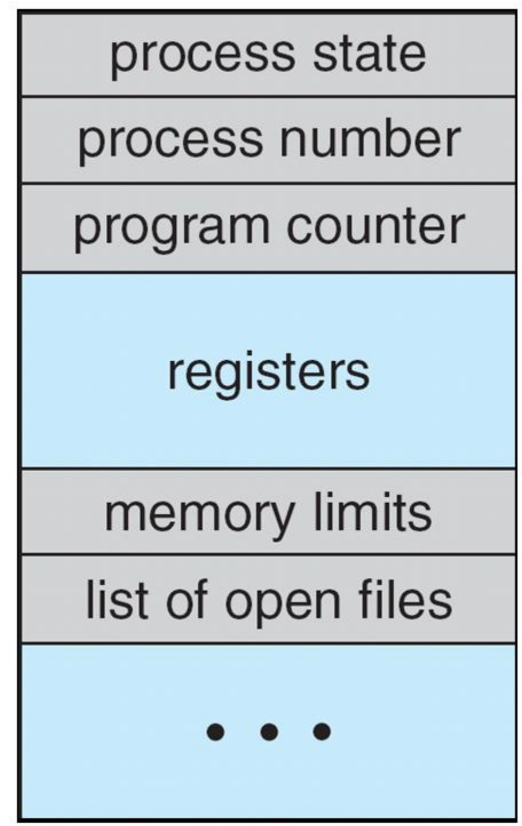
# Diagram of Process State



# Process Control Block (PCB)

- **Process Control block** is a data structure used for storing the information about a process.
- Each & every process is identified by its own PCB.
- It is also called as context of the process.
- PCB of each process resides in the main memory.
- PCB of all the processes are present in a linked list.

PID = Process identifier
Unique number

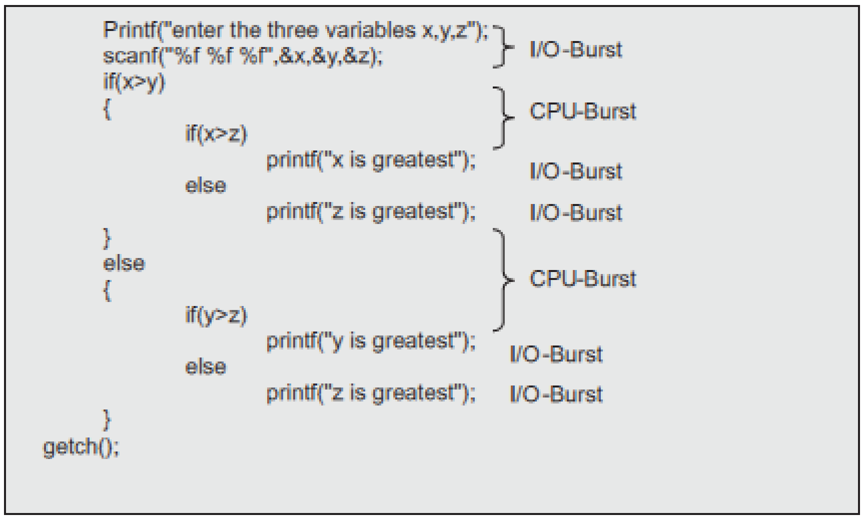


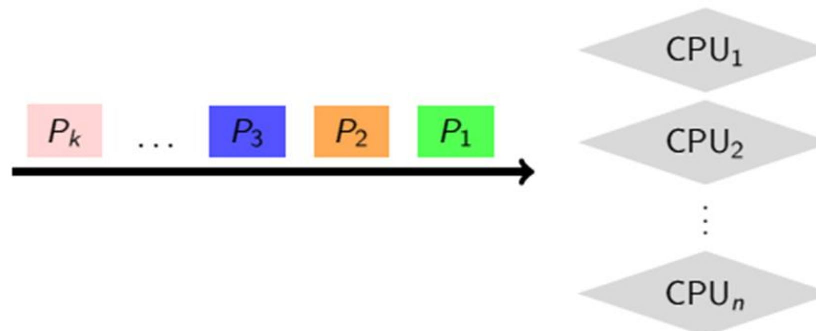

- PCB is important in **multiprogramming** environment as it captures the information pertaining to the number of processes running simultaneously.

- Process Concept
- **Process Scheduling**
- Operations on Processes
- Inter-process Communication

# Process Scheduling

Process execution consists of alternating sequence of *CPU execution* and *I/O wait*.




The scheduling problem:

- The system has **k** processes ready to run
- The system has **n ≥ 1 CPUs** that can run them

# Process Scheduling

- **Process scheduler** selects from among the processes in memory that are ready to execute, and allocates the CPU to one of them
- **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

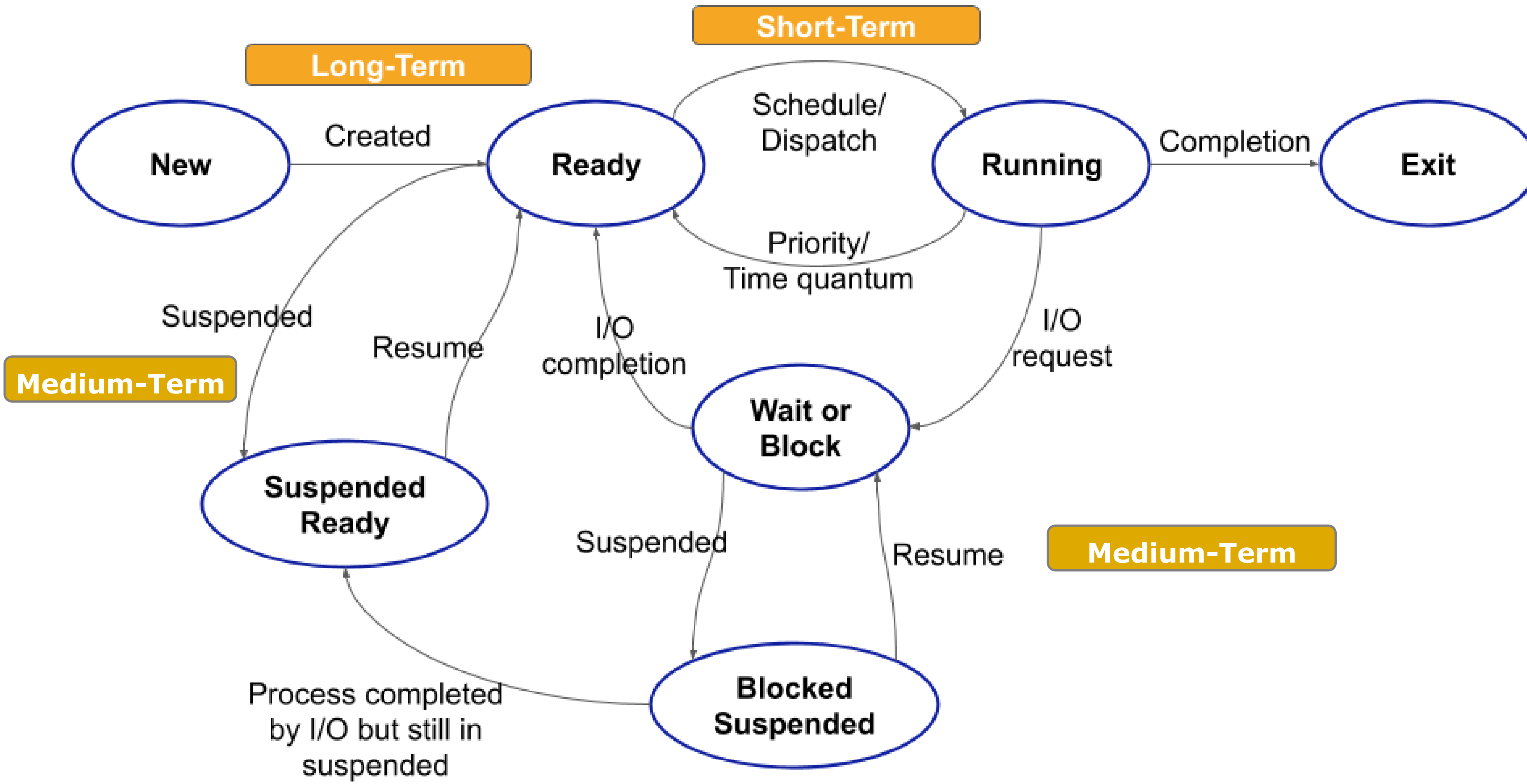
# Schedulers

- **Long-Term Scheduler** is also called ***Job Scheduler*** and is responsible for controlling the Degree of Multiprogramming i.e. the total number of processes that are present in the ready state.

- **Short-Term Scheduler** is also known as ***CPU scheduler*** and is responsible for selecting one process from the ready state for scheduling it on the running state.

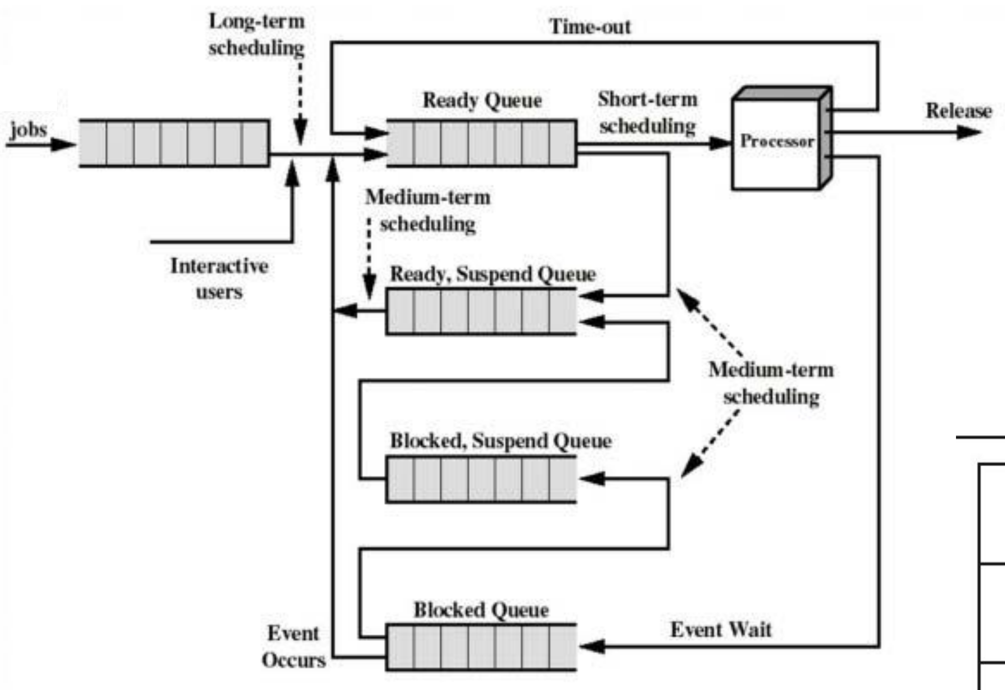
- **Medium-term scheduler** is responsible for ***swapping of a process from the Main Memory to Secondary Memory and vice-versa*** (mid-term effect on the performance of the system).

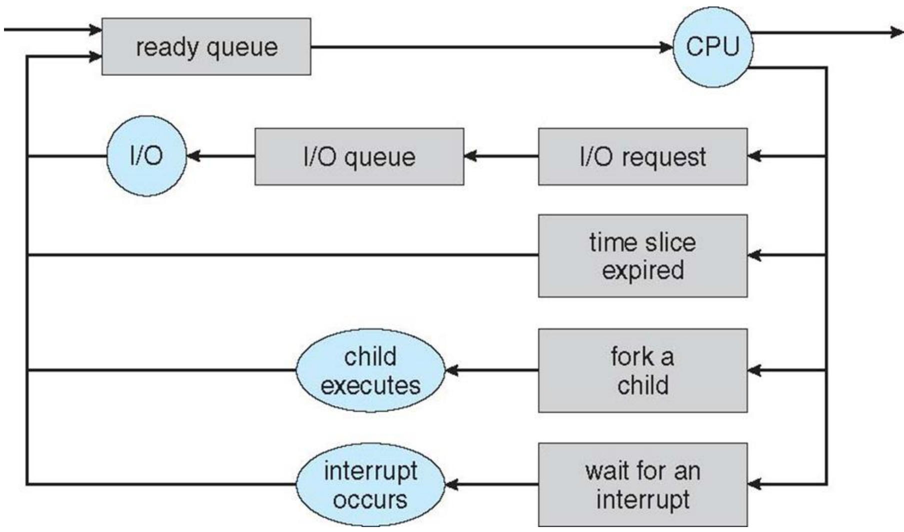
# Scheduling Levels



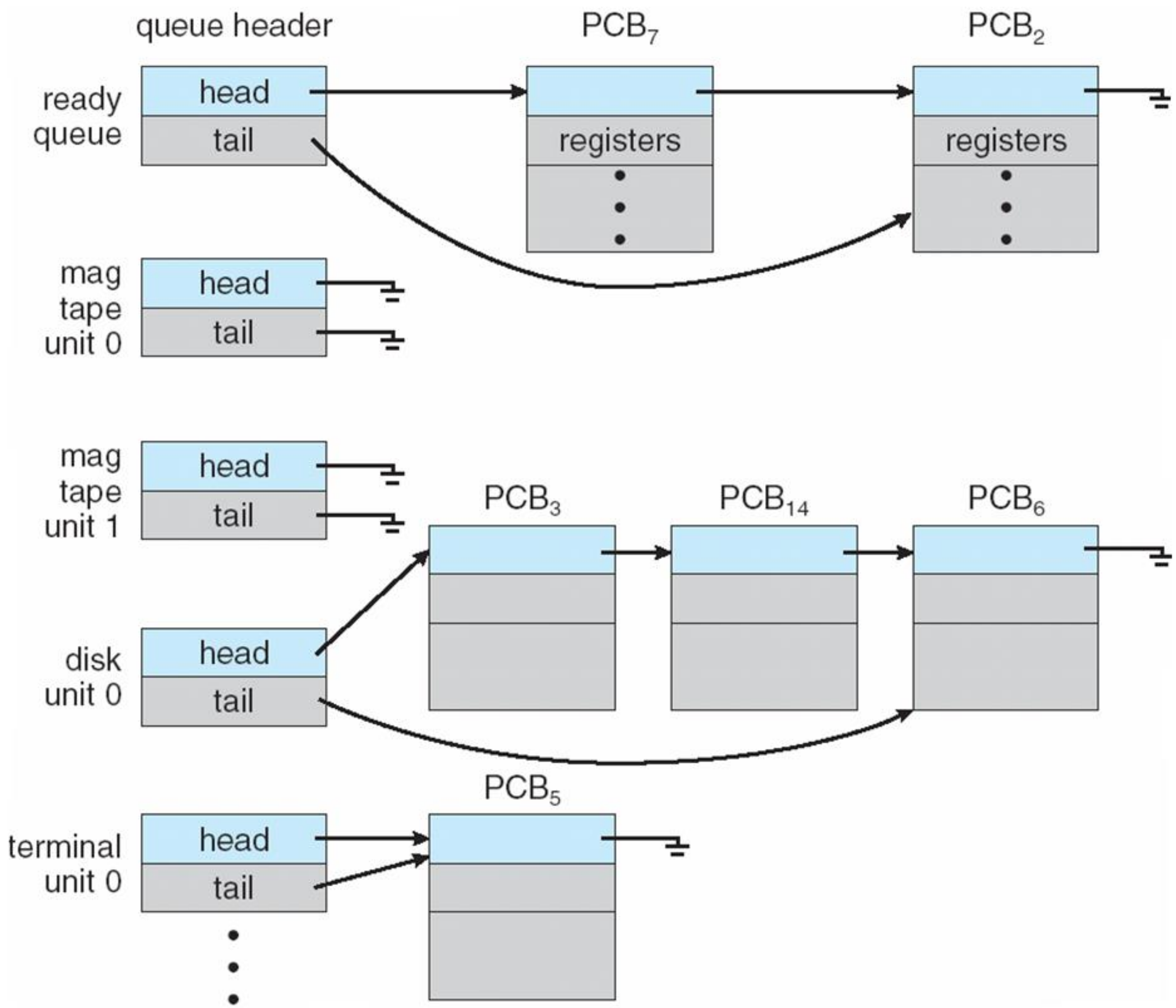
# Queueing Diagram for Scheduling

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



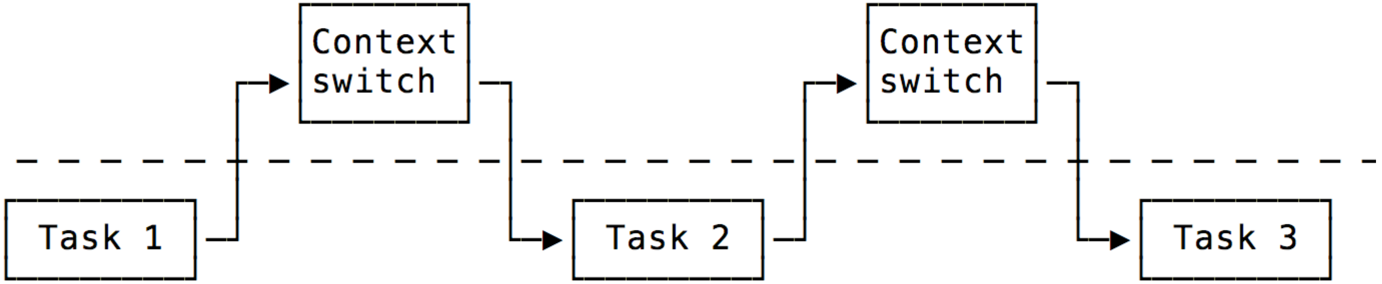



# Ready Queue And Various I/O Device Queues

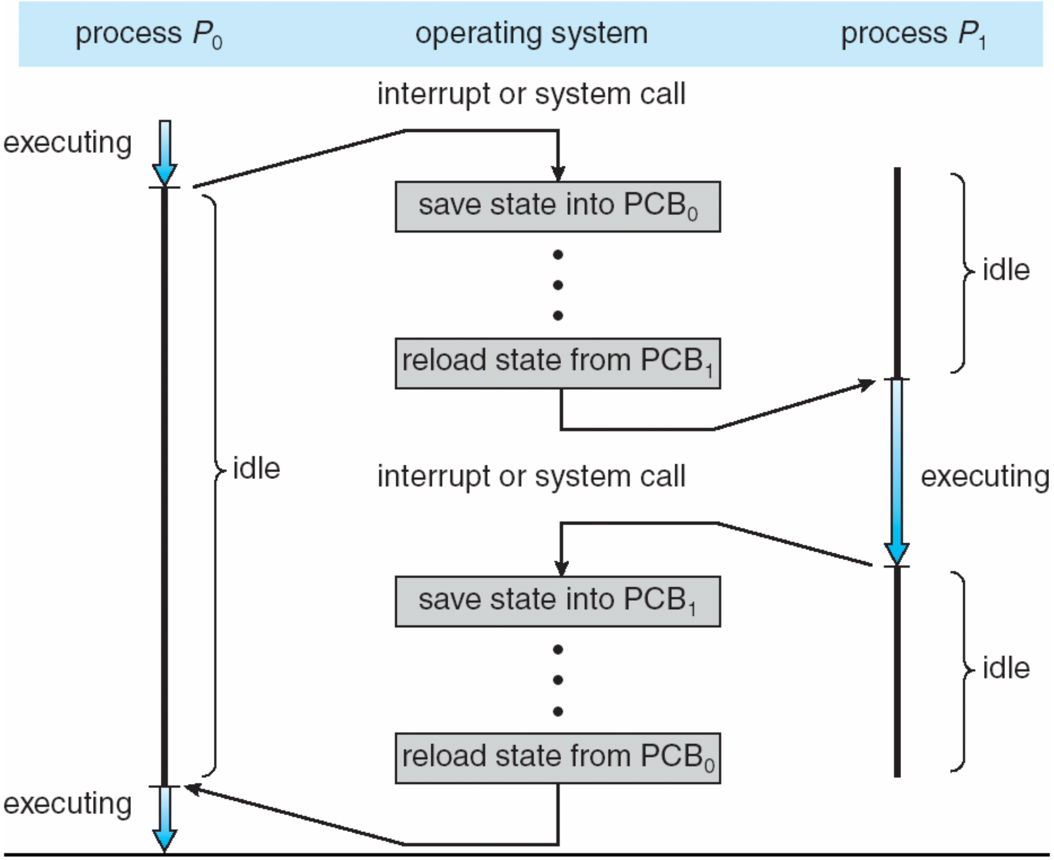


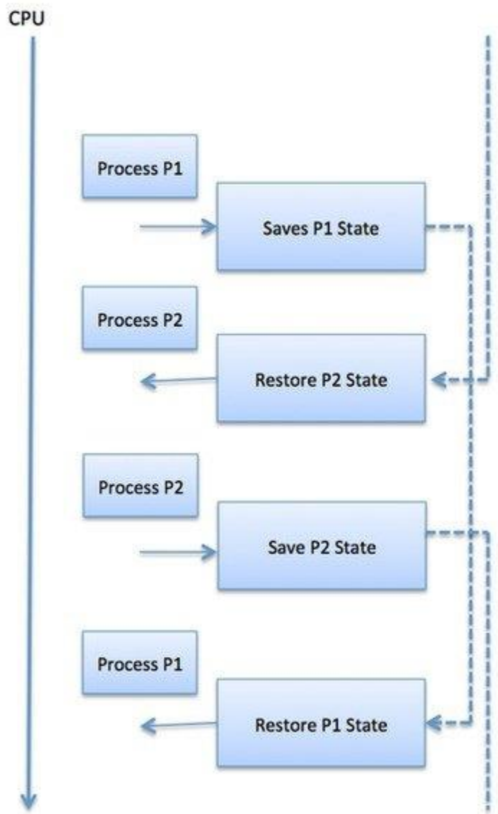
# Context Switch

- 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



# CPU switch from process to process



- Process Concept
- Process Scheduling
- **Operations on Processes**
- Inter-process Communication

# Operations on Processes

System must provide mechanisms for:

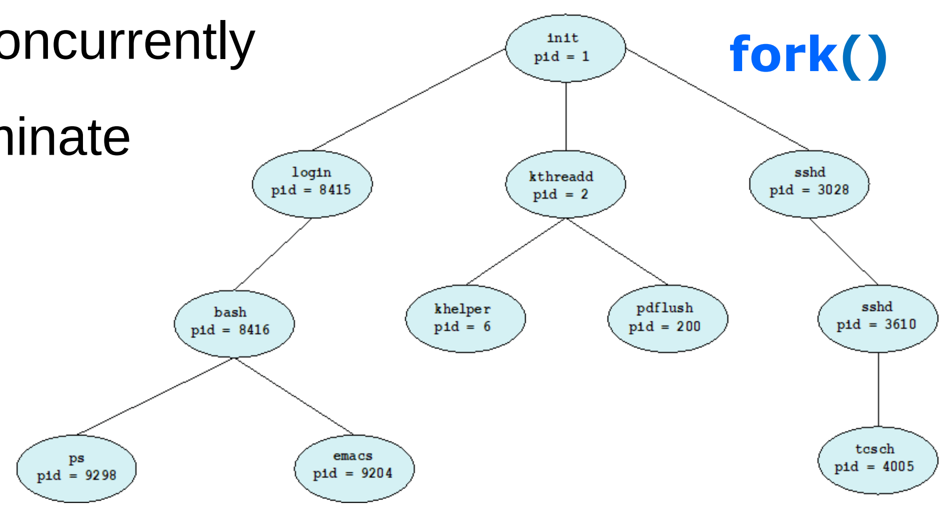
- process creation,
- process termination,

# Process Creation

**Parent** process create **children** processes, which, in turn create other processes, forming a **tree** of processes

- **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

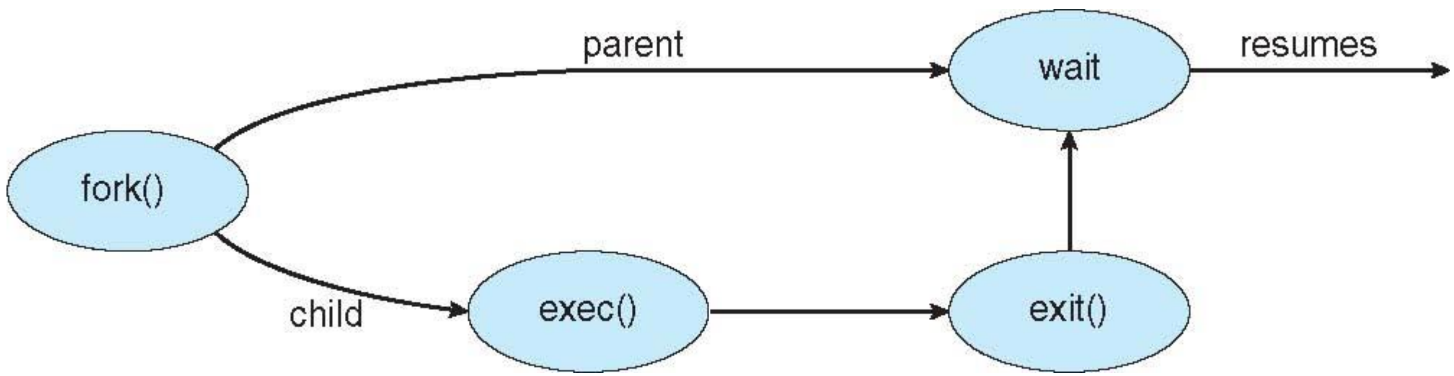
**fork()**



# Process Termination

- Process executes last statement and then asks the operating system to delete it using the **exit()** system call.
  - Returns  status data from child to parent
  - Process' resources are deallocated by operating system

- Parent may wait terminate the execution of children processes.
  - Child has exceeded allocated resources



- Process Concept
- Process Scheduling
- Operations on Processes
- **Inter-process Communication**

# Inter-process Communication

**INDEPENDENT PROCESSES** - *neither affect other processes or be affected by other processes*.

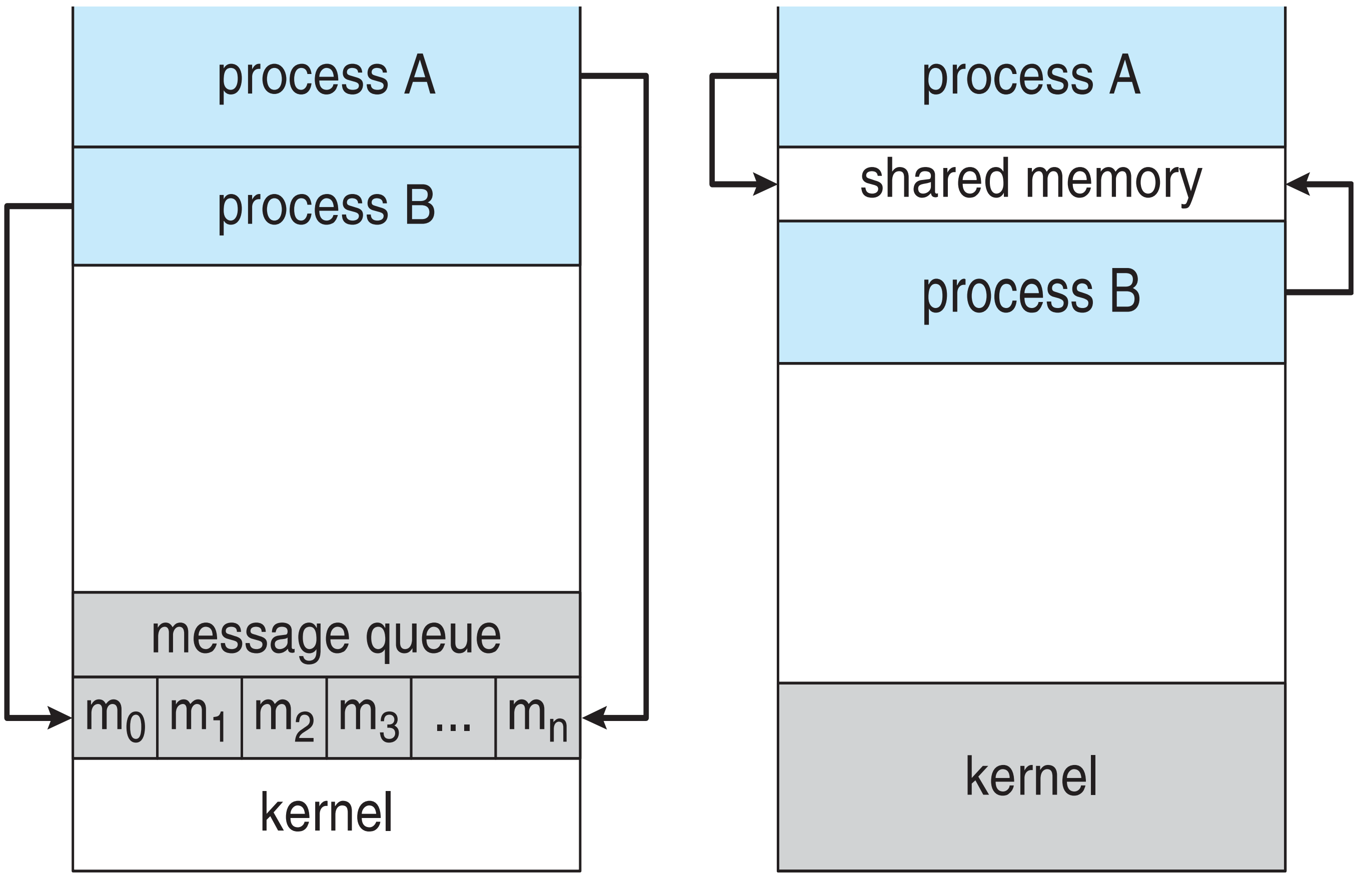
**COOPERATING PROCESSES** - *can affect or be affected by other processes.*

There are several reasons why cooperating processes are allowed:

- **Information Sharing** - processes which need access to the same file for example.
- **Computation speedup** - a problem can be solved faster if the problem can be broken down into sub-tasks to be solved simultaneously
- **Modularity** - break a system down into cooperating modules. (e.g. databases with a client-server architecture.)
- **Convenience** - even a single user may be multi-tasking, such as editing, compiling, printing, and running the same code in different windows.

# Communications Models

**(a) Message passing.** **(b) Shared memory.**



# (a) Message-Passing Systems

- communication takes place by way of messages exchanged among the <u>*cooperating processes*</u>.
- A message-passing facility provides at least two operations:
  - **send**(*message*)
  - **receive**(*message*)
- The *message* size is either fixed or variable

# (a) Message-Passing Systems

- If processes *P* and *Q* want to communicate: *a communication link must exist between them*.
- Are several methods for logically implementing a link and the `send()`/`receive()` operations:
  - **Direct** or **indirect communication**
  - **Synchronous** or **asynchronous communication**

# (b) Shared-Memory Systems

- *a region of memory is shared by cooperating processes*.
- processes can *exchange information by reading and writing all the data to the shared region*.

**Two types of buffers** can be used:

- **unbounded-buffer** places **no** practical **limit on the size of the buffer**
- **bounded-buffer** assumes that there is a **fixed buffer size**

# Direct or Indirect communication

- exchanged messages by communicating processes reside in a **temporary queue**.

**BUFFERING**

**Zero capacity**. The queue has a maximum length of zero; thus, the link cannot have any messages waiting in it.

**Bounded capacity**. The queue has finite length *n*; thus, at most *n* messages can reside in it.

**Unbounded capacity**. The queue's length is potentially infinite.

# 1. Direct Communication

- Processes must name each other explicitly:
  - **send** (*P, message*) – send a message to process P
  - **receive**(*Q, message*) – receive a message from process Q
- *Direct Communication* is implemented when *the processes use specific process identifier for the communication*, but it is hard to identify the sender ahead of time.

# 2. Indirect Communication

- **create** a new **mailbox (port)**
- send and receive messages through mailbox
  - **send**(*A, message*) – send a message to mailbox A
  - **receive**(*A, message*) – receive a message from mailbox A
- **destroy** a **mailbox**

# Synchronous and Asynchronous Message Passing

- Message passing may be either *blocking* or *non-blocking*
- **Blocking** is considered **synchronous**
  - **Blocking send** -- the sender is blocked until the message is received
  - **Blocking receive** -- the receiver is blocked until a message is available
- **Non-blocking** is considered **asynchronous**
  - **Non-blocking send** -- the sender sends the message and continue
  - **Non-blocking receive** -- the receiver receives:
    - *A valid message*, or
    - *Null message*

# End of Lecture

- **Summary**

  - Process Concept
  - Process Scheduling
  - Operations on Processes
  - Inter-process Communication

- **Reading**
  - Textbook 9th edition, **chapter 1 + chapter 2 + chapter 3 of module textbook**