# OPERATING SYSTEMS

Module2_Part1

Textbook : Operating Systems Concepts by Silberschatz

# Process Management

Process:

Most central concept of any operating system is the process.

process is a program in execution; process execution must progress in sequential fashion. No parallel execution of instructions of a single process.

The process includes the current activity as represented by the value of the program counter and the content of the processor registers.

The program is only part of a process.

Also it includes the process stack which contain temporary data (such as method parameters return address and local variables) & a data section which contain global variables.

# Process Management

**Difference between process & program:**

A program by itself is not a process. A program in execution is known as a process. A program becomes a process when an executable file is loaded into memory.

A program is

a passive entity, such as the contents of a file stored on disk .

process is an active entity with

- 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

A system therefore consists of a collection of processes:

operating system processes -executing system code

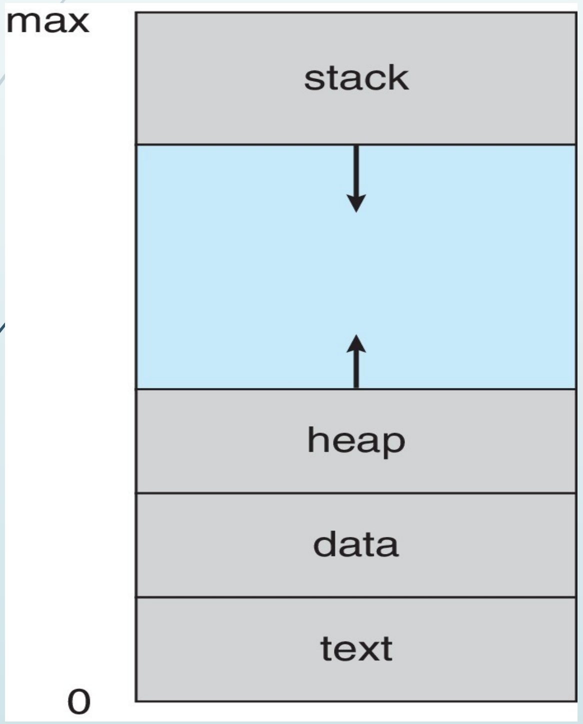
user processes-executing user code.

Potentially all these processes can execute concurrently with the CPU

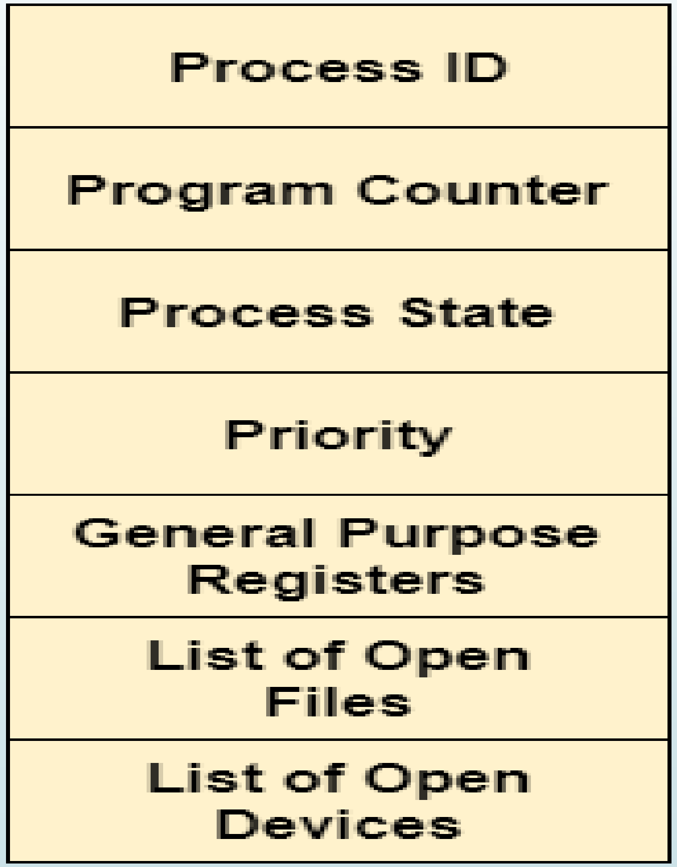
By switching the CPU between processes

the operating system can make the computer more productive.

## Process in memory



## Attributes of a process



Process Attributes

## Process States

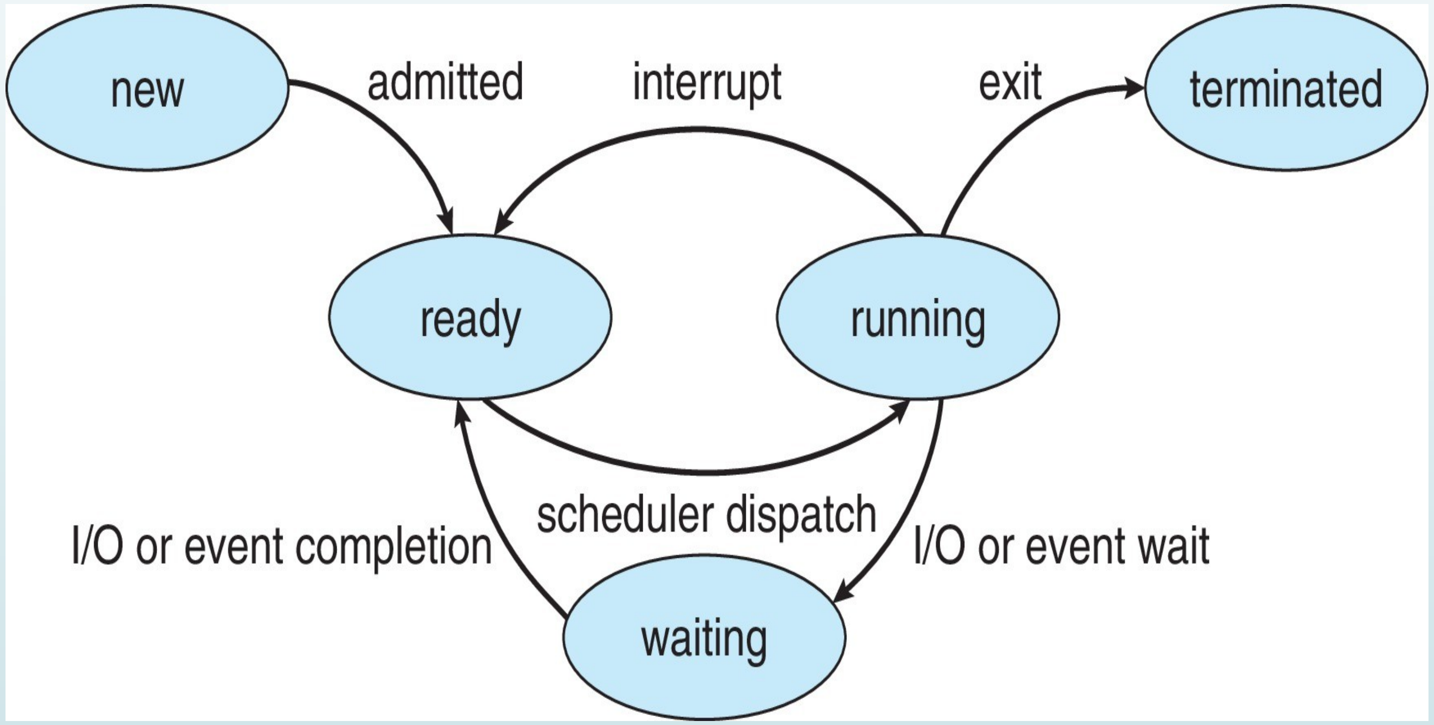
**Process State**

As a process executes, it changes state. The state of a process is defined in part by the

current activity of that process. Each process may be in one of the following states:

- **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 states



# Process control block

- Each process is represented in the operating system by a **process control block (PCB)**—

  also called a **task control block.**
- It contains many pieces of information associated with a specific process, including these:

  Process state – running, waiting, etc.

  Program counter – indicates the address of the next instruction to be executed
- CPU registers – include accumulators, index registers, stack pointers, and general-purpose registers, plus any condition-code information.

Along with the program counter, this state information must be saved when an interrupt occurs, to

allow the process to be continued correctly afterwards

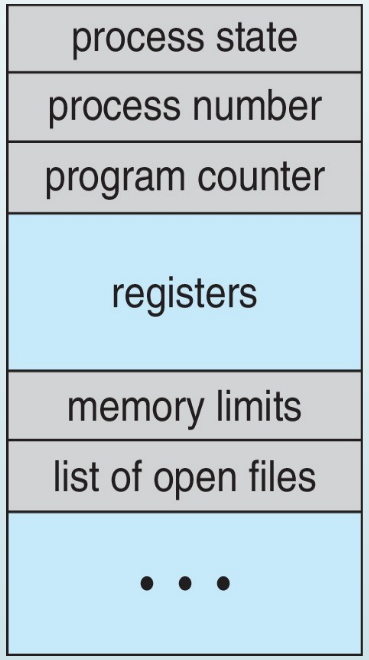
# Process control block

CPU scheduling information
- priorities, scheduling queue pointers

Memory-management information
– memory allocated to the process This information may include such information as the value of the base and limit registers, the page tables, or the segment tables, depending on the memory system used by the operating system

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



## Threads

A **thread** is a single sequential flow of control within a program. It is possible that multiple **threads** can be run at the same time and performing different tasks in a single program.

Threads are very useful in modern programming whenever a process has multiple tasks to perform independently of the others.

A process was an executing program with a single thread of control. Most modern operating systems now provide features enabling a process to contain multiple threads of control.

On a system that supports threads, the PCB is expanded to include information for each thread

In a multithreaded process on a single processor, the processor can switch execution between threads, resulting in concurrent execution.

# Threads

A thread is a basic unit of CPU utilization; it comprises a thread ID, a program

counter, a register set, and a stack.

It shares with other threads belonging to the same process, its code section, data section, and other operating-system resources, such as open files and signals

for example.

A word processor may have a thread for displaying graphics, another thread

for responding to keystrokes from the user, and a third thread for performing

spelling and grammar checking in the background.