# Lecture 2. Processes in Operating System

Lecturer: Aidana Karibayeva

# Process

The most central concept in any operating system is the process: an abstraction of a running program.
All modern computers can do several things at the same time. While running a user program, a computer can also be reading from a disk and outputting text to a screen or printer. In a multiprogramming system, the CPU also switches from program to program, running each for tens or hundreds of milliseconds.
CPU gives the users the illusion of parallelism (pseudoparallelism).

# What is a process?

- The basic unit of software that the operating system deals with in scheduling the work done by the processor.
- Not the same as an application, since an application can cause multiple processes to be executed.
- Can be defined as software that performs some action and can be controlled - by a user, by other applications or by the operating system.

# Process Model

All the runnable software on the computer (also the operating system) is organized into a number of sequential processes.
A process is just an executing program, including the current values of the program counter, registers, and variables.
Conceptually, each process has its own virtual CPU. In reality, of course, the real CPU switches back and forth from process to process. This rapid switching back and forth is called multiprogramming.

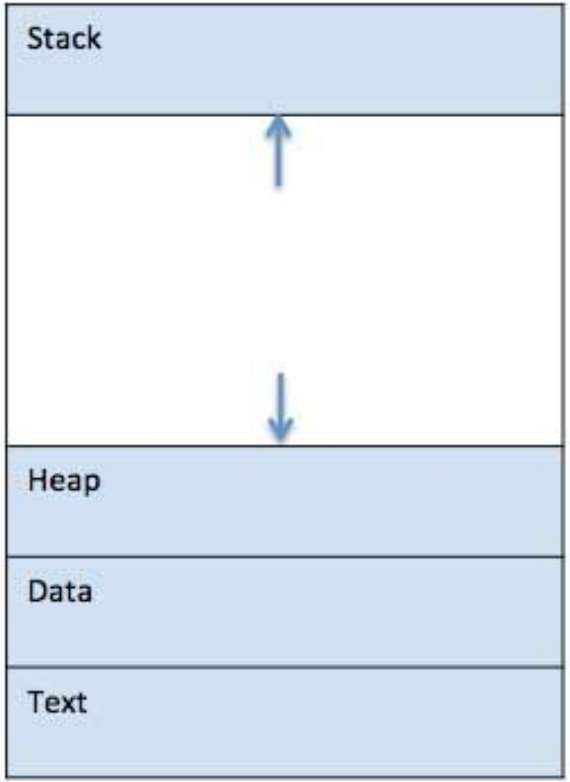
# Process Control Block(PCB)

- A process consists of the machine code image of the program and a Process Control Block (PCB).
- The PCB typically contains:
  - An ID number that identifies the process
  - Pointers to the locations in the program and its data where processing last occurred
  - Register contents
  - States of various flags and switches
  - Pointers to the upper and lower bounds of the memory required for the process
  - A list of files opened by the process
  - The priority of the process
  - The status of all I/O devices needed by the process

A process is basically a program in execution. The execution of a process must progress in a sequential fashion.

To put it in simple terms, we write our computer programs in a text file and when we execute this program, it becomes a process which performs all the tasks mentioned in the program.

When a program is loaded into the memory and it becomes a process, it can be divided into four sections ─ stack, heap, text and data. The following image shows a simplified layout of a process inside main memory −

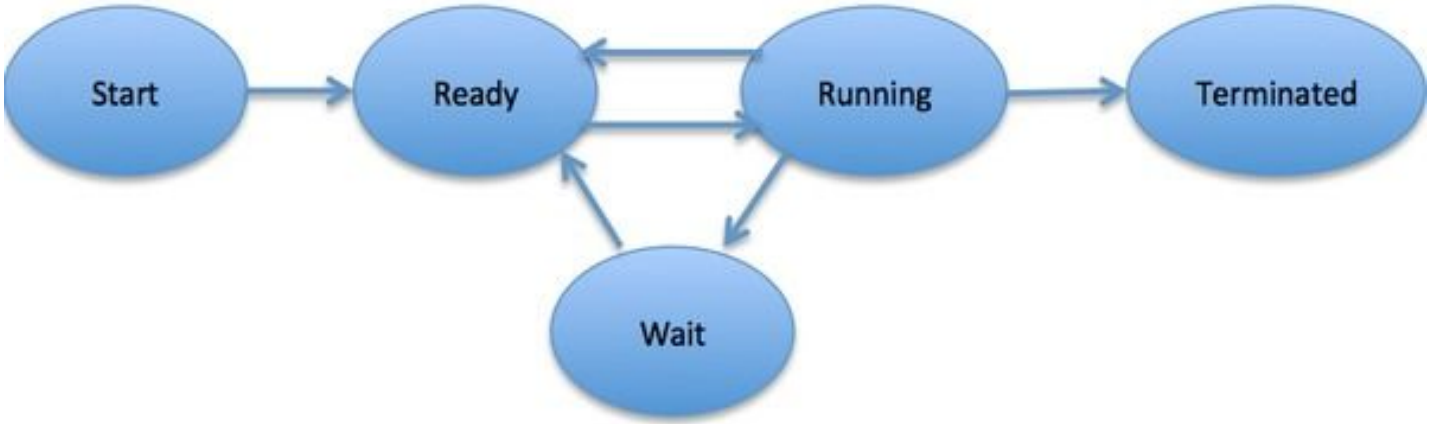


| | |
|---|---|
| Stack | The process Stack contains the temporary data such as method/function parameters, return address and local variables. |
| Heap | This is dynamically allocated memory to a process during its run time. |
| Text | This includes the current activity represented by the value of Program Counter and the contents of the processor's registers. |
| Data | This section contains the global and static variables |

## Process Life Cycle

When a process executes, it passes through different states. These stages may differ in different operating systems, and the names of these states are also not standardized.

In general, a process can have one of the following five states at a time.



- **Start**

This is the initial state when a process is first started/created.

- **Ready**

The process is waiting to be assigned to a processor. Ready processes are waiting to have the processor allocated to them by the operating system so that they can run. Process may come into this state after Start state or while running it by but interrupted by the scheduler to assign CPU to some other process.

- **Running**

Once the process has been assigned to a processor by the OS scheduler, the process state is set to running and the processor executes its instructions.

- **Waiting**

Process moves into the waiting state if it needs to wait for a resource, such as waiting for user input, or waiting for a file to become available.

- **Terminated or Exit**

Once the process finishes its execution, or it is terminated by the operating system, it is moved to the terminated state where it waits to be removed from main memory.

## Process Control Block (PCB)

A Process Control Block is a data structure maintained by the Operating System for every process. The PCB is identified by an integer process ID (PID). A PCB keeps all the information needed to keep track of a process as listed below:

- process state
- Process privileges
- Process ID
- Pointer
- Program Counter
- CPU registers
- CPU Scheduling Information
- Memory management information
- Accounting information
- I/O status information

- **Process State**

The current state of the process i.e., whether it is ready, running, waiting, or whatever.

- **Process privileges**

This is required to allow/disallow access to system resources.

- **Process ID**

Unique identification for each of the process in the operating system.

- **Pointer**

A pointer to parent process.

- **Program Counter**

Program Counter is a pointer to the address of the next instruction to be executed for this process

- **CPU registers**

Various CPU registers where process need to be stored for execution for running state.

- **CPU Scheduling Information**

Process priority and other scheduling information which is required to schedule the process.

- **Memory management information**

This includes the information of page table, memory limits, Segment table depending on memory used by the operating system.

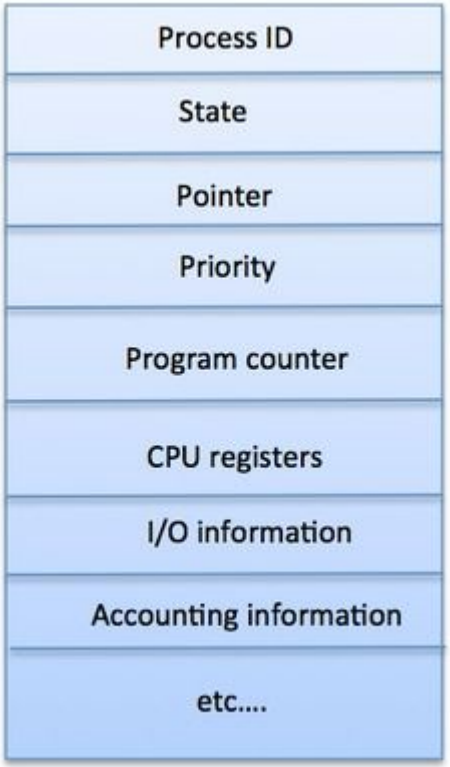
- **Accounting information**

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.

**NOTE:** The architecture of a PCB is completely dependent on Operating System and may contain different information in different operating systems.



Thank you for your attention!