CO - 1

Session : 3

COURSE NAME : SYSTEM DESIGN AND INTRODUCTION TO CLOUD

COURSE CODE : 23AD2103A

TOPICS : PROCESS VIRTUALIZATION:

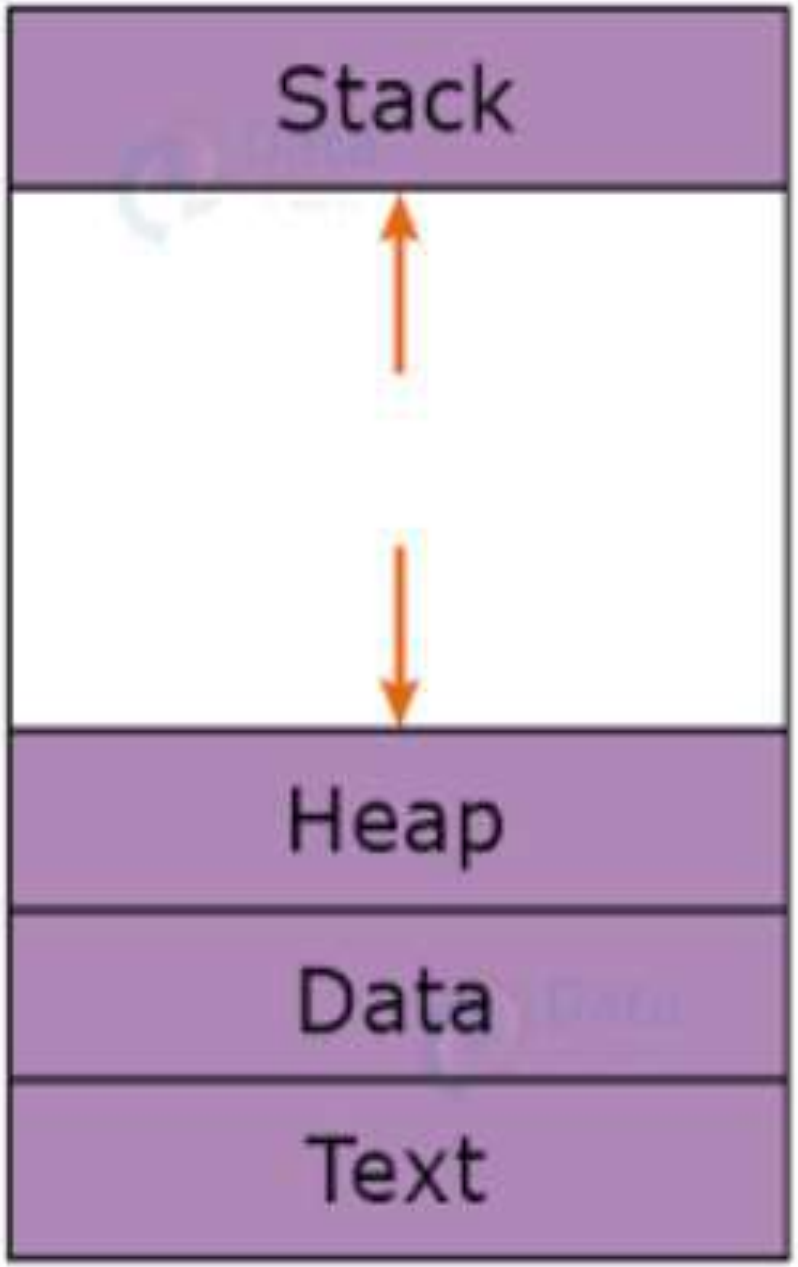
PROCESSES, PROCESS API CODE, DIRECT EXECUTION

SESSION DESCRIPTION

- Process
- Process virtualizatiom
- Process API
- Process synchronization
- Process control block

PROCESS

- A process is mainly a program in execution where the execution of a process must progress in a sequential order or based on some priority or algorithms.
- In other words, it is an entity that represents the fundamental working that has been assigned to a system.
- When a program gets loaded into the memory, it is said to as process. This processing can be categorized into 4 sections.
- These are:
    - Heap
    - Stack
    - Data
    - Text



•The **Stack** is used for local variables. Space on the stack is reserved for local variables when they are declared.

•The **Heap** is used for the dynamic memory allocation and is managed via calls to new, delete, malloc, free, etc.

•The **Data section** is made up of the global and static variables, allocated and initialized prior to executing the main.

•The **Text section** is made up of the compiled program code, read in from non-volatile storage when the program is launched.

PROCESS VIRTUALIZATION

- Process virtualization is a technology that allows individual processes (or applications) to run in isolated environments, separated from the underlying operating system and other processes.
- This technology abstracts the application from the underlying hardware and operating system, providing a more flexible and secure way to manage and run applications.

KEY ASPECTS OF PROCESS VIRTUALIZATION

- **Isolation:**
- Each virtualized process runs in its own isolated environment, preventing interference from other processes. This isolation enhances security and stability, as problems in one process do not affect others.
- **Portability:**
- Virtualized processes can be moved and executed on different systems without modification. This portability is possible because the virtual environment abstracts the application from the specific details of the underlying hardware and OS.

KEY ASPECTS OF PROCESS VIRTUALIZATION

- **Resource Management:** Process virtualization allows better control and allocation of system resources such as CPU, memory, and I/O. This ensures that each process gets the resources it needs while avoiding conflicts and overuse.
- **Security:** By isolating processes, virtualization enhances security. Each process operates in a sandbox, reducing the risk of one process compromising the entire system.
- **Scalability:** Virtualized processes can be easily replicated and scaled to meet demand. This scalability is beneficial in cloud environments where applications need to handle varying loads.
- **Simplified Deployment and Management:** Deploying and managing applications becomes simpler with process virtualization. Administrators can package applications with all their dependencies and configurations, ensuring consistent behavior across different environments.

PROCESS API

- Process APIs interact with and shape data within a single system or across systems — breaking down data silos.
- Process APIs provide a means of combining data and orchestrating multiple System APIs for a specific business purpose.

PROCESS API

- These APIs are available on any modern OS
- **Create**

  OS is invoked to create a new process to run a program
- **Destroy**

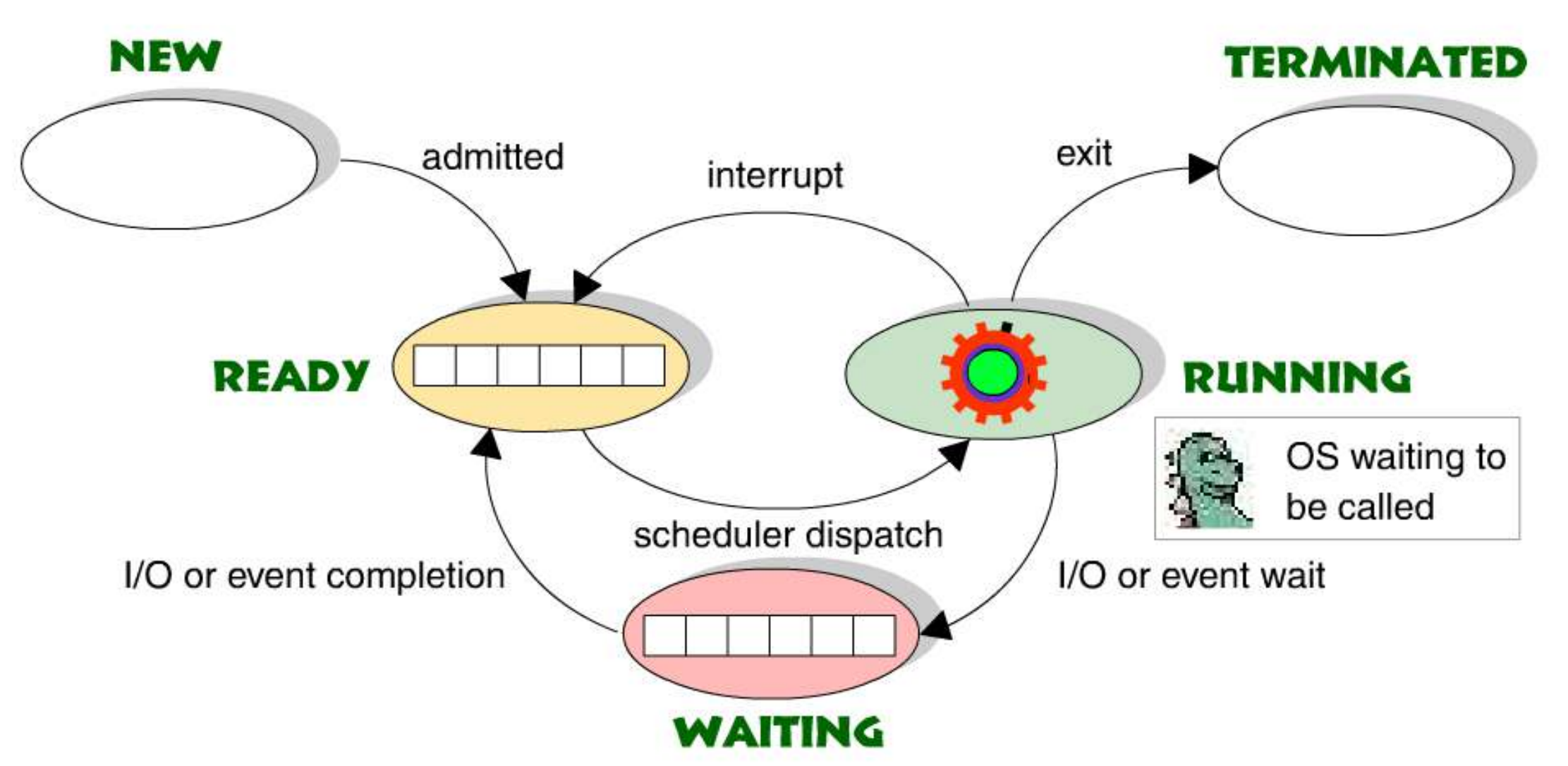
  Halt a runway process
- **Wait**

  Wait for a process to stop running
- **Miscellaneous control**

  Some method to suspend a process and then resume it
- **Status**

  Get some status info about a process

PROCESS STATES IN OPERATING SYSTEM



PROCESS STATES IN OPERATING SYSTEM

- **Start / New :** This is the initial state when a process is first started or created.
- **Ready :** The process is waiting to be assigned to a processor.
  1. Ready processes are waiting to have the processor allocated to them by the operating system so that they can run.
  2. 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 / 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. These names are arbitrary, and they vary across operating systems.

PROCESS SYNCHRONIZATION

- Process synchronization as designed to provide an efficient use of shared data between more than one thread among 1 process or from different processes. It resolves the problem of Race condition.
- **Types of Processes:** There are two types of processes
- **Independent process**
- Independent process is a process that cannot be affected by other processes and don't affect the other processes as well.
- It does not share any data, variable, or any resource like Memory, CPU and printer etc.
- **Co-operative process**
- Co-operative process is a process that can be affected by other processes and can also affect the other processes as well.
- Co-operative processes share any data, variable, or any resource like Memory, CPU and printer etc.

PROCESS SYNCHRONIZATION

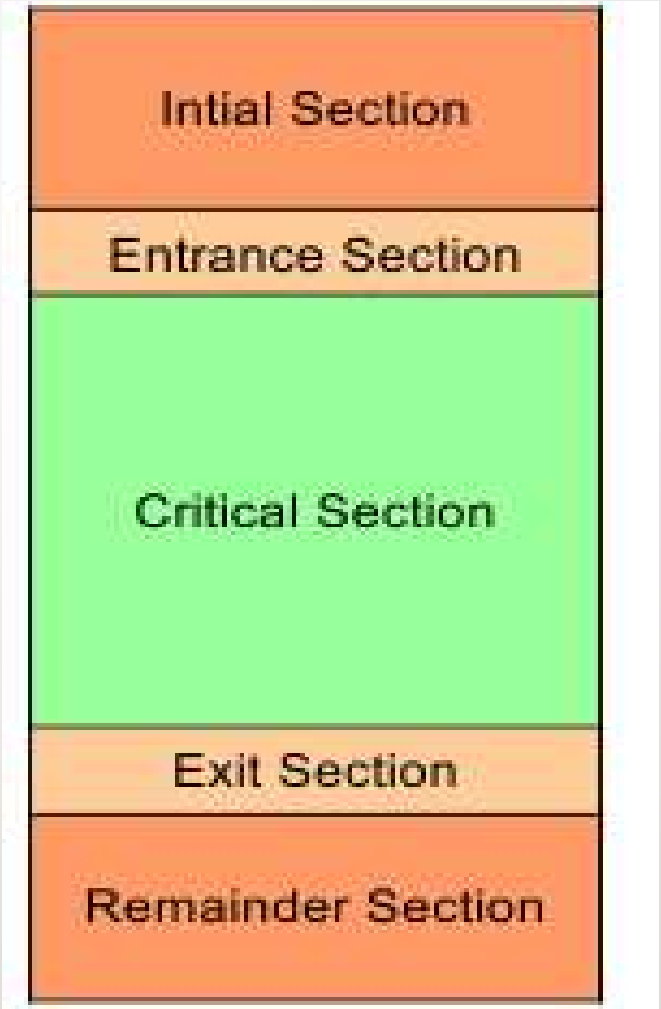
- **Race condition:** A situation where several processes access and manipulate the same data (critical section). The outcome depends on the order in which the access take place. So, the results are inconsistent.
- Race Condition occurs when two or more threads try to read, write or access memory concurrently.
- A race condition is a condition when there are many processes and every process shares the data with each other and accessing the data concurrently, and the output of execution depends on a particular sequence in which they share the data and access.
- To prevent the race condition, we need to ensure that only one process can access the shared data at a time. Prevented race conditions by process synchronization
- This is the main reason why we need to synchronize the processes.

PROCESS SYNCHRONIZATION

- **Critical Section:** Critical section is a specific region of program which contains the shared resources of cooperative processes.
- It allows only one process at a time to enter into the critical section. So that, race conditions can be eliminated, and process synchronization can be achieved.
- **Purpose of process synchronization:** To handle multiple cooperative processes which are running parallelly to eliminate race condition(s).

WHY PROCESS SYNCHRONIZATION IS IMPORTANT?

- Process synchronization can be achieved by using critical section. To get process synchronization every process follows the following diagram.
- **Initial section:** it contains Non-shared data .
- **Entry section:** Entry code contains conditions and lock the allocated resources
- **Critical section:** All shared data is placed in critical section.
- **Exit Section:** Contains exit code which release the locked resources.
- **Remainder Section:** it contains Non-shared data.

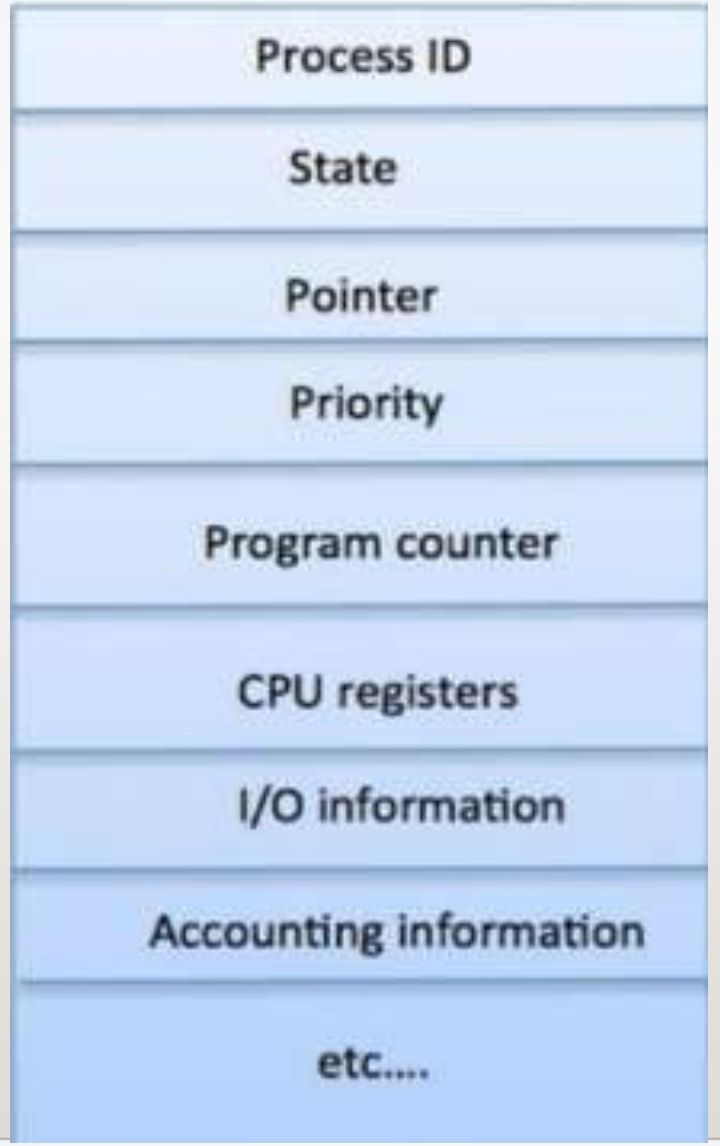


Process Synchronization

PROCESS SYNCHRONIZATION

```
Process()
  {
     while()
      {
         initial section
         Entry section
         Critical section
         Exit section
         Remainder section
  }
}
```

PROCESS CONTROL BLOCK (PCB)



PROCESS CONTROL BLOCK (PCB)

- **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.

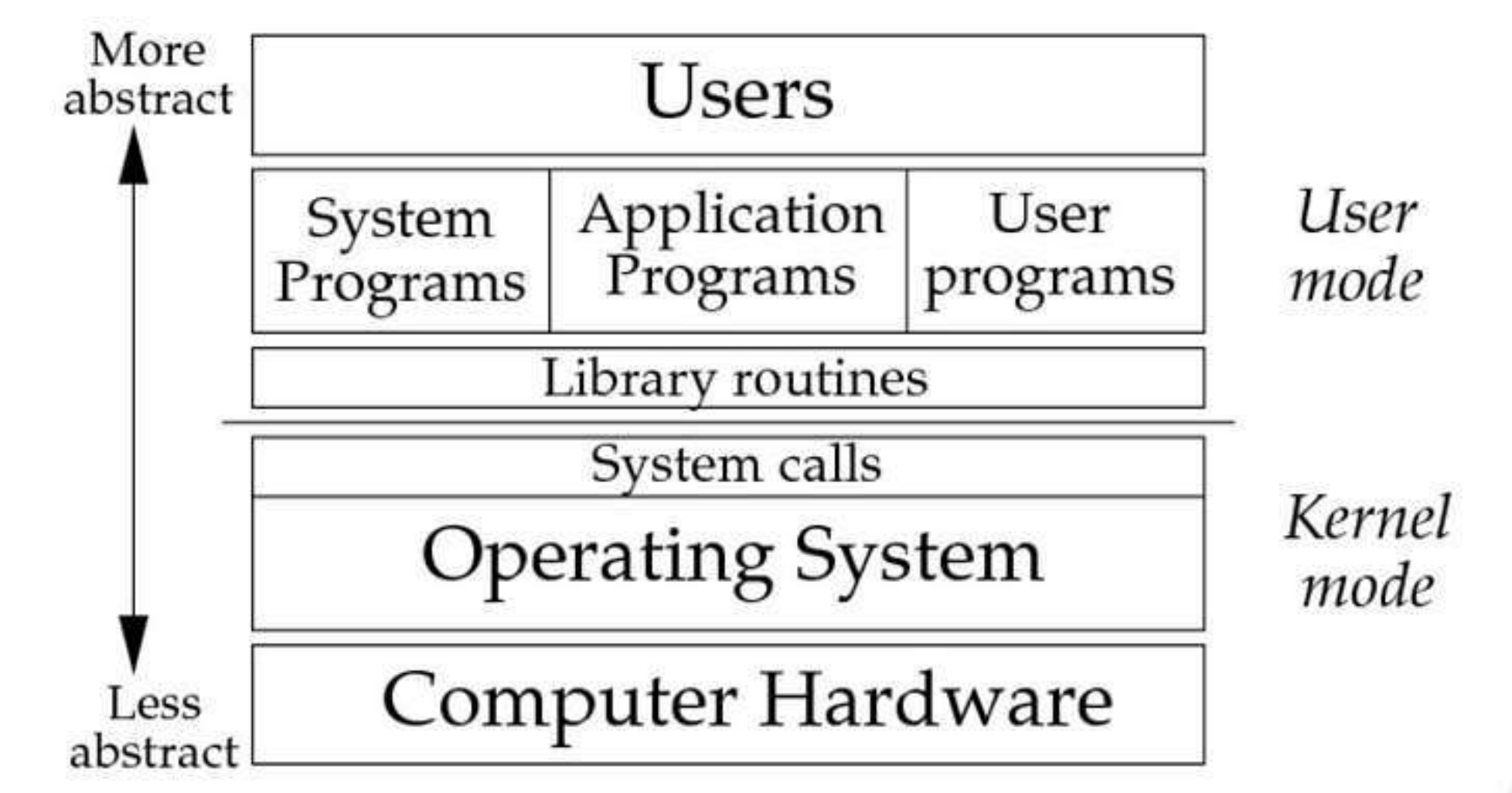
PROCESS CONTROL BLOCK (PCB)

- **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.

PROCESS CONTROL BLOCK (PCB)

- **Accounting Information**
  - This includes the amount of CPU used for process execution, time limits, execution ID etc.
- **IO Status Information**
  - This includes a list of I/O devices allocated to the process.
  - The architecture of a PCB is completely dependent on Operating System and may contain different information in different operating systems.

DIRECT EXECUTION



DIRECT EXECUTION

- To make a program run as fast as one might expect, OS developers came up with a technique: **limited direct execution,** aka. run the program directly on the CPU.
- The operating system runs in **kernel mode**. In kernel mode, the software has complete access to all of the computer's hardware, and can control the switching between the CPU modes. Interrupts are also received in the kernel mode software.
- There is also the **user mode**. In this mode, direct access to the hardware is prohibited. For example, when running in user mode, a process can't issue I/O requests; otherwise exception is raised.

THANK YOU

(DEEMED TO BE UNIVERSITY)

Team – System Design & Introduction to Cloud