Operating System Concepts

SunBeam Institute of Information & Technology, Hinjwadi, Pune & Karad.

Trainer: Akshita Chanchlani
Email: akshita.chanchlani@sunbeaminfo.com

Operating System Concepts

# Process Management

- When we say an OS does process management it means an OS is responsible for process creation, to provide environment for an execution of a process, resource allocation, scheduling, resources management, inter process comminication, process coordination, and terminate the process.

Q. What is a Program?

-**User view:** Program is a set of instructions given to the machine to do specific task.

-**System view:** Program is an executable file divided into sections like exe header, bss section, data section, rodata section, code section, symbol table.

Operating System Concepts

Q. What is a Process?

User view:

- Program in execution is called as a process.
- Running program is called as a process.
- When a program gets loaded into the main memory it is reffered as a process.
- Running instance of a program is reffered as a process.

System view:

- Process is a file loaded into the main memory which has got bss section, rodata section, code section, and two new sections gets added for the process:

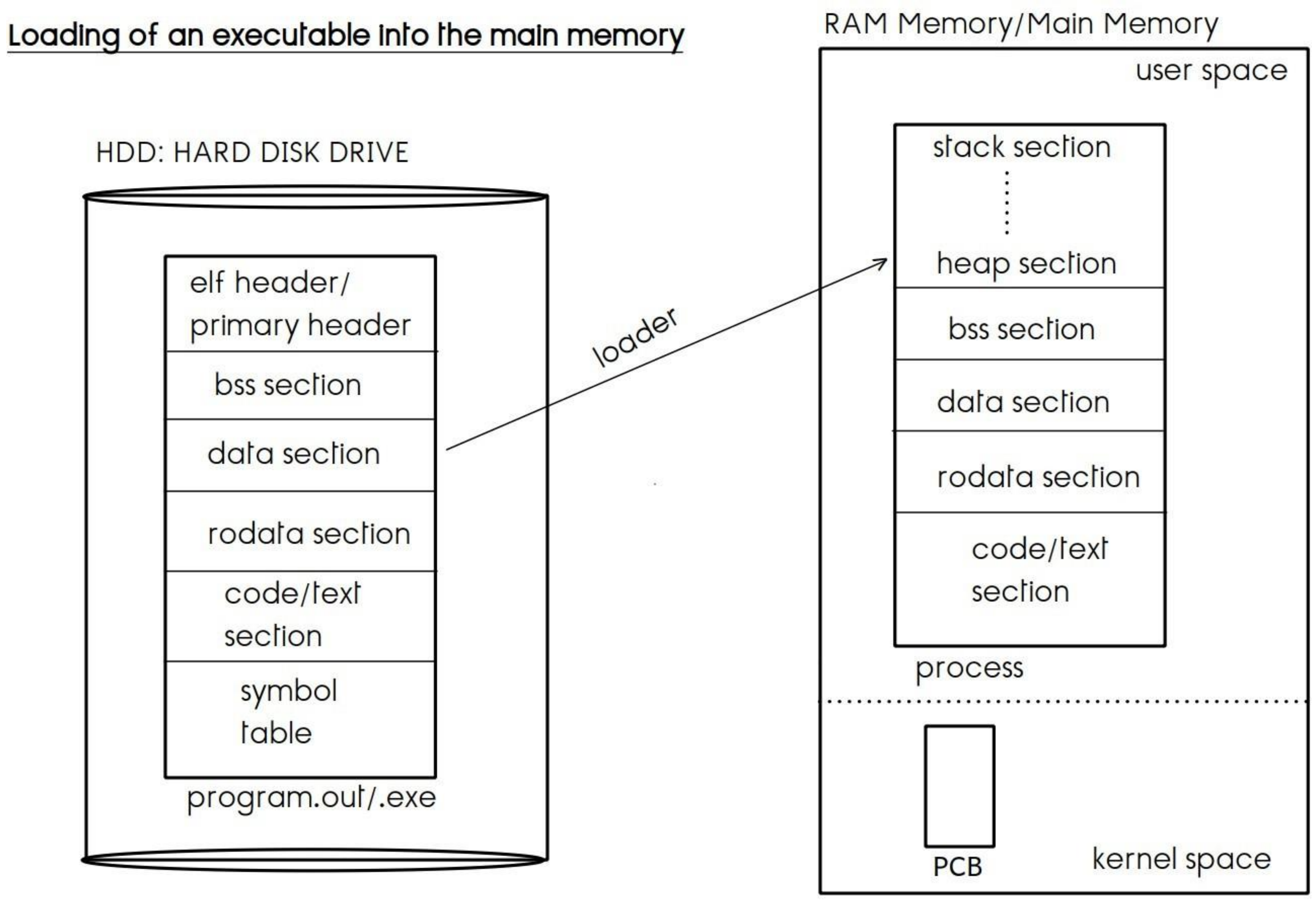
stack section: contains function activation records of called functions.

heap section: dynamically allocated memory

Process and Program

- A **process** is an instance of a program in execution.
- Running program is also knows as **Process.**
- When a program gets loaded in to memory is also known as **Process.**
- A **Program** is a set of instructions given to the machine to do specific task.
  - Three types of Programs:
    - User Programs (c/java program)
    - Application Programs (ms office)
    - System Programs (device drivers, interrupt handlers etc)

Operating System Concepts



Memory Layout of Program and Process

Program Consist of
exe header/primary header
Block started by symbol (bss) section (un initialized static / global variables)
Data section (initialized static / global variables)
Rodata Section(Constant/literals)
code/text section (contains executable instructions)
Symbol Table

Process Consist of
Skipped
Block started by symbol (bss) section (un initialized static / global variables)
Data section (initialized static / global variables)
Rodata Section(Constant/literals)
code/text section (contains executable instructions)
Skipped
Stack Section
Heap Section

Operating System Concepts

-As a kernel, core program of an a OS runs continuosly into the main memory, **part of the main memory which is occupied by the kernel reffered as kernel space and whichever part is left is reffered as an user space**, so main memory is divided logically into two parts: **kernel space & user space.**

-User programs gets loaded into the user space only.

Process Control Block/ Process Descriptors

- When an execution of any program is started one structure gets created for that program/process to store info about it, for controlling its execution, such a structure is known as PCB: Process Control Block.
- Per process one PCB gets created and PCB remains inside the main memory throughout an execution of a program, upon exit PCB gets destroyed from the main memory.
- It has information about process like:
  - process id – pid
  - process state - current state of the process
  - memory management info
  - CPU scheduling info
  - Program Counter -- address of next instruction to be executed
  - Exit status
  - Execution Context
  - I/O devices info etc...

A Two State Process Model

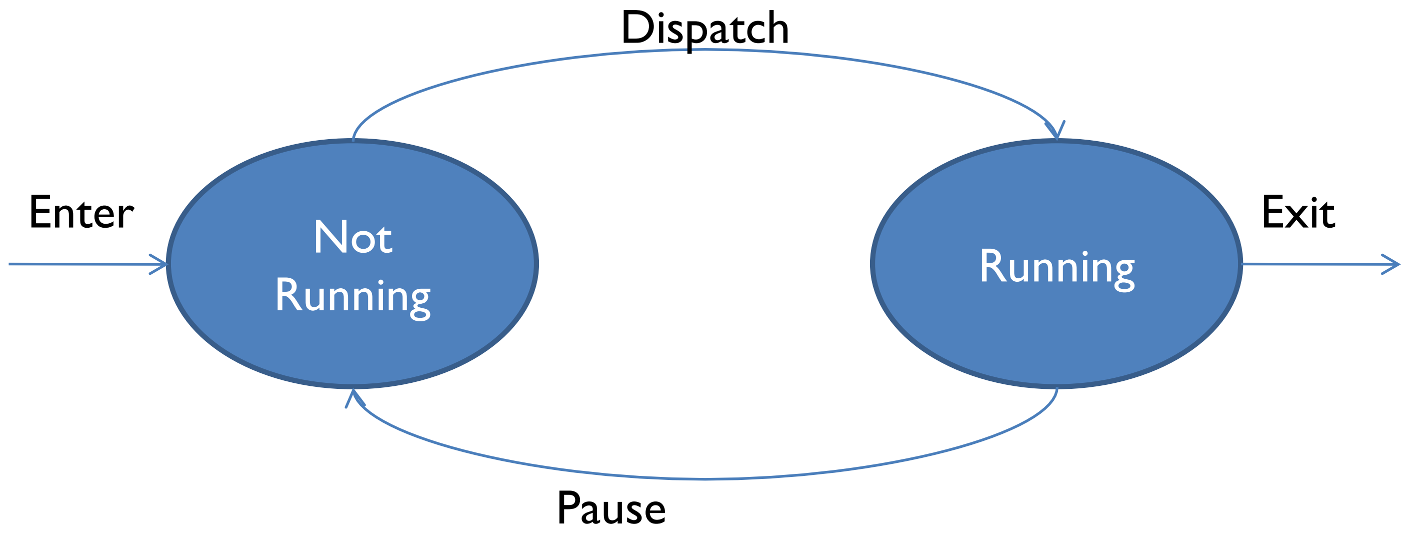


Fig. Two State Process Diagram

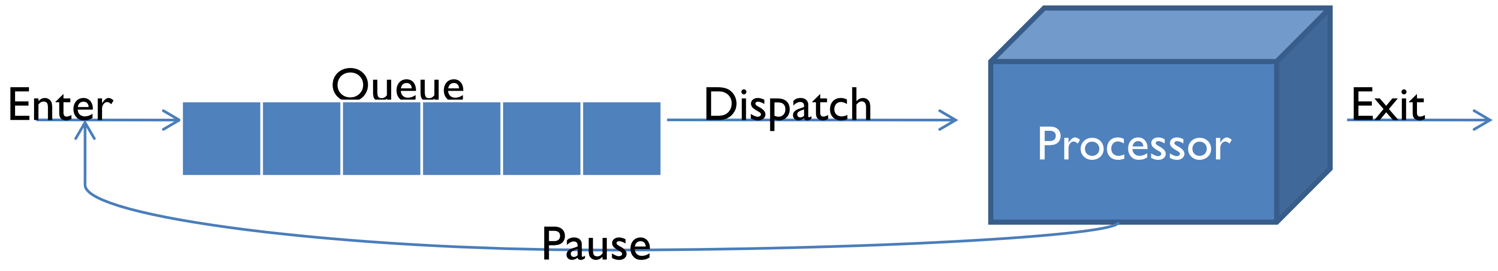


Fig. Queuing Diagram

Five State Process Diagram

# Process States:

-Throughout execution, process goes through different states out of which at a time it can be only in a one state.

-States of the process:

1. New state: upon submission or when a PCB for a process gets created into the main memory process is in a new state.

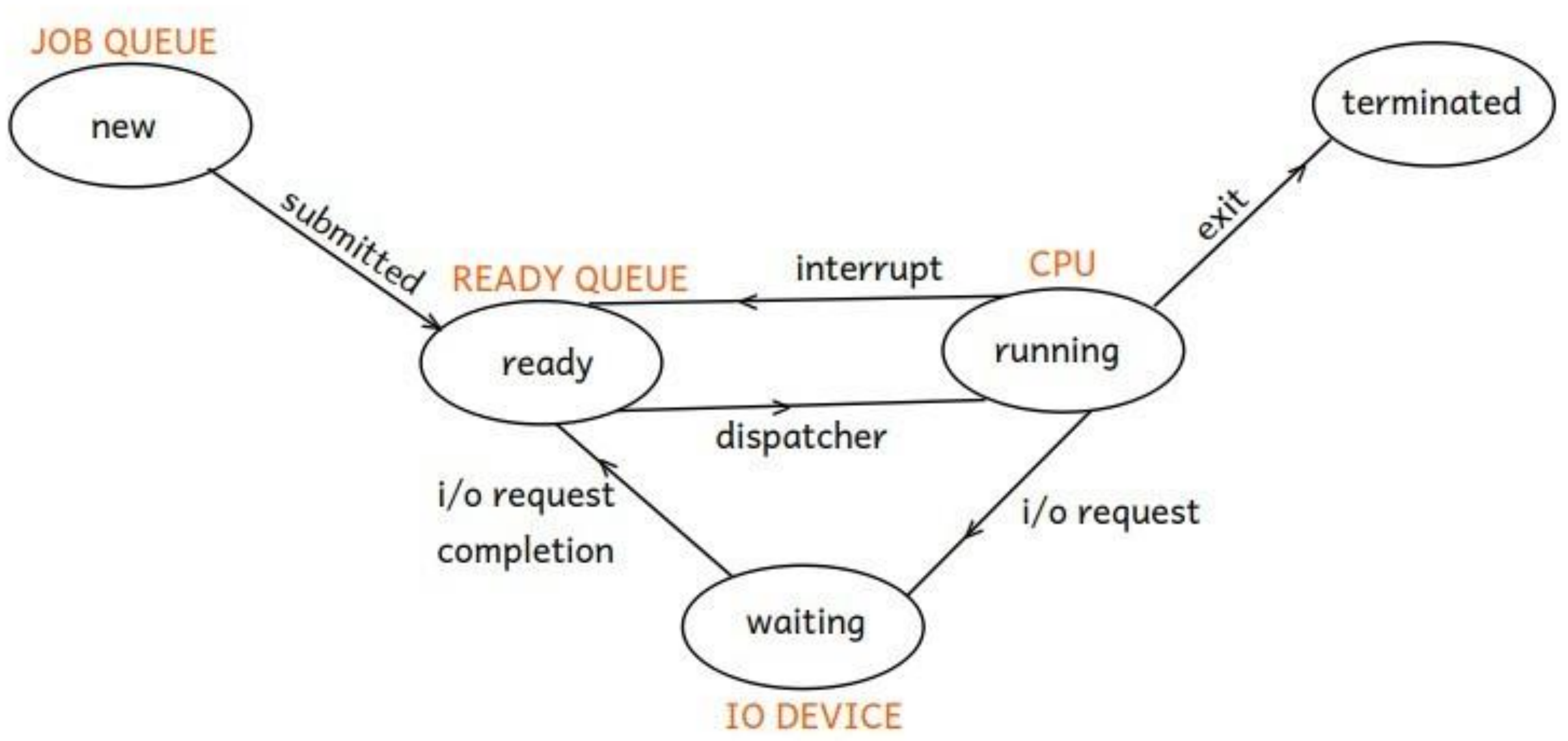
2. Ready state: after submission, if process is in the main memory and waiting for the CPU time, it is in a ready state.

3. Running state: if currently the CPU is executing any process then state of that process is considered as a running state.

4. Waiting state: if a process is requesting for any i/o device then state of that process is considered as a waiting state.

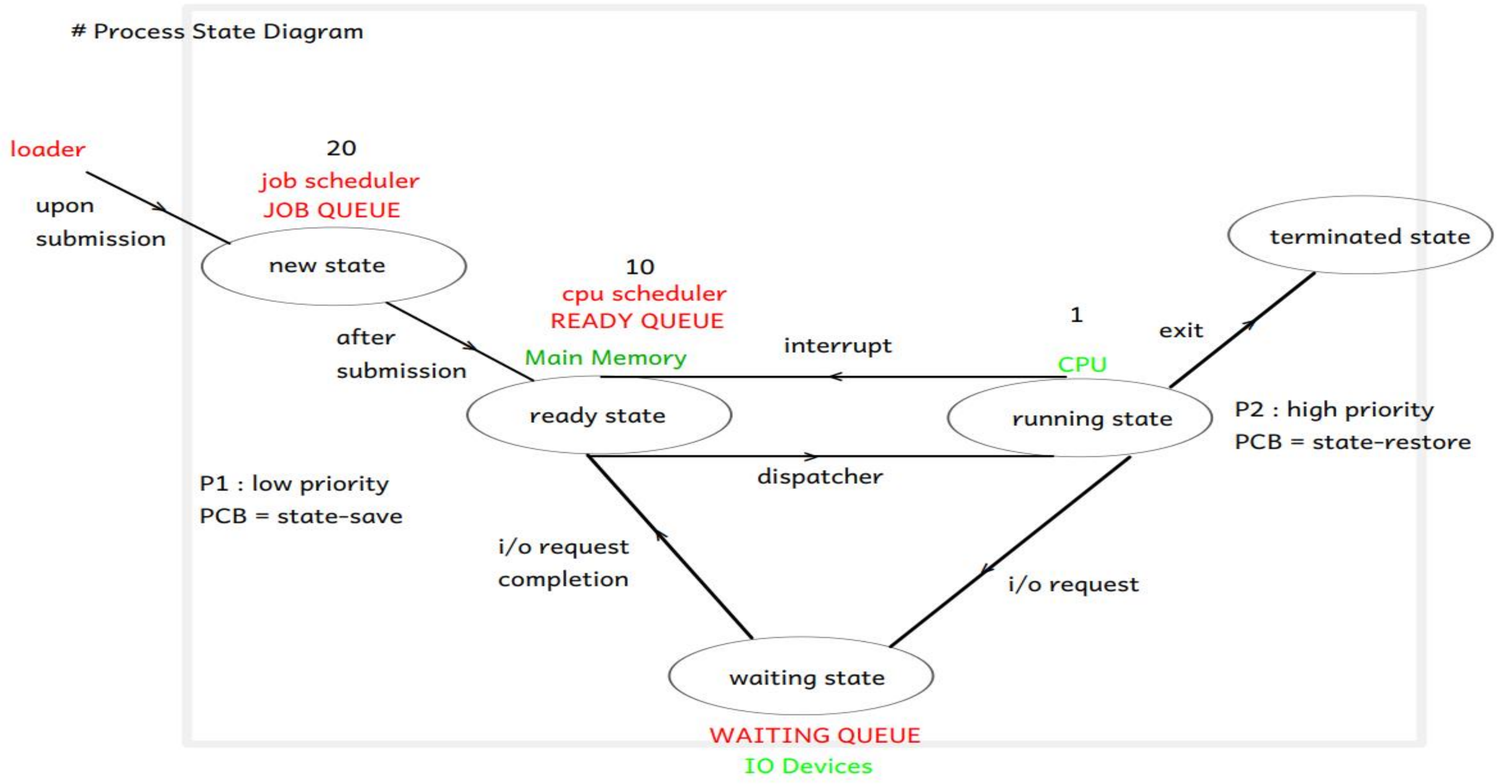
5. Terminated state: upon exit, process goes into terminated state and its PCB gets destroyed from the main memory.

Five State Process Diagram



PROCESS STATE DIAGRAM

Five State Process Diagram



Data Structures Maintained by Kernel at the time of process Execution

# Process States:

-To keep track on all running programs, an OS maintains few **data structures** reffered as **kernel data structures:**

1. Job queue: it contains list of PCB's of all submitted processes.

2. Ready queue: it contains list of PCB's of processes which are in the main memory and waiting for the CPU time.

3. Waiting queue: it contains list of PCB's of processes which are requesting for that particular device.

Process May be in one of the state at a time

New

- when program execution is started or upon process submission process
- when a PCB of any process is in a job queue then state of the process is referred as a new state.

Ready

- When a program is in a main memory and waiting for the cpu
- when a PCB of any process is in a ready queue then state of the process is referred as a ready state.

Running

- When a CPU is executing a process

Exit

- when a process is terminated

Waiting

- when a process is requesting for any i/o device then process change its change from running to waiting state
- When a PCB of any process is in a waiting queue of any device

Schedulers

- **Job Scheduler/long term schedulers :**

  It is a system program which selects/schedules jobs/processes from job queue to load them onto the ready queue.
- **CPU Scheduler/Short term schedulers**
  - It is a system program that selects / schedules process from ready queue to load into CPU
  - selects which process should be executed next and allocates CPU
- **<u>Dispatcher</u>**
  - It is a system program that loads a process onto the CPU that is scheduled by the CPU scheduler.
  - Time required for the dispatcher to stops execution and one process and starts execution of another process is called as **"dispatcher latency".**

Context Switch

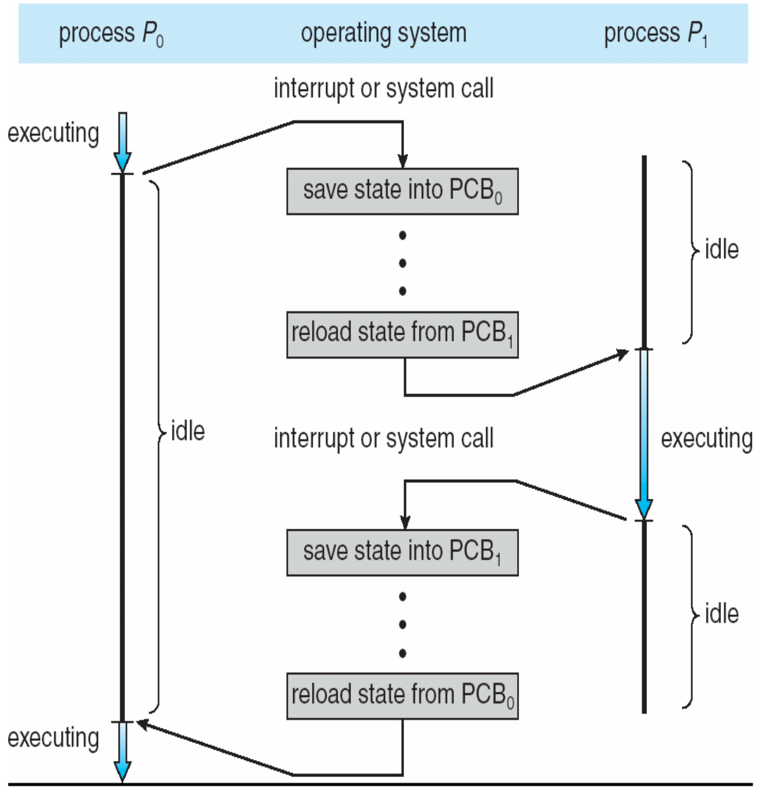
- When CPU switches to another process, the system must save the state of the old process and load the saved state for the new process via a **context switch**.
- **Context** of a process represented in the PCB.
- Context-switch time is overhead; the system does no useful work while switching
  - The more complex the OS and the PCB -> longer the context switch.

Context Switch = state-save + state-restore

"state-save" -- to save execution context of suspended process into its PCB
"state-restore" -- to restore execution context of the process which is scheduled by the cpu scheduler onto the cpu registers.

When a high priority process arrived into the ready queue, low priority process gets suspended by means of sending an interrupt, and control of the CPU gets allocated to the high priority process, and its execution gets completed first, then low priority process can be resumed back, i.e. the CPU starts executing suspended process from the point at which it was suspended and onwards.

Context Switching



CPU scheduler should get called in following four cases:

1.running --> terminated
2.running --> waiting
3.. running --> ready
4. waiting --> ready

Operating System Features

1. multi-programming: system in which more than one processes can be submitted/an execution of more than one programs can be started at a time.

-**degree of multi-programming:** no. of programs that can be submitted into the system at a time.

2. multi-tasking: system in which, the CPU can execute more than one programs simultaneously/concurrently, the speed at which it executes multiple programs simultaneosuly it seems that it executes multiple programs at a time.

At a time the CPU can execute only one program

-**time-sharing:** system in which CPU time gets shared among all running programs.

3. multi-threading: system in which it seems that the CPU can execute more than one threads which are of either same process or are of different processes simultaneously/concurrently.

4. multi-processor: system can run on a machine in which more than one CPU's are connected in a closed circuit.

5. multi-user: system in which multiple users can loggedin at a time.