
CODE DIGITAL
HARDWARE DESIGN
INTERNET
SCREEN
DATA
SERVICE
GRAPHIC
INPUT
APPLICATION
OS
APP
WIRELESS
USER
NETWORK
DEVELOPMENT
INFORMATIONCONCEPT
COMPUTING
CONCEPTUAL
MEMORY
BACKGROUND
DESKTOP
PROGRAMMING
OPERATING
SYSTEM
TECHNOLOGY
ALLOCATION
PROGRAM
DEVICE
COMPUTER
SOFTWARE
STORAGE
MOBILE
BUSINESS
WEB
COMMUNICATION
SYMBOL
OUTPUT

Jump2Learn
The Online Learning Place

OPERATING SYSTEM - II

CHAPTER 1 : PROCESS MANAGEMENT

AUTHORS

Mr. Yatin Solanki
M.C.A., Ph.D.(Pursuing)

DOLAT USHA INSTITUTE OF APPLIED SCIENCES, VALSAD

Dr. Jaimin Shukla
M.C.A., M. Phil., Ph.D.(Pursuing)

SUTEX BANK COLLEGE OF COMPUTER APPLICATIONS & SCIENCE, AMROLI

Website : www.jump2learn.com | **Email :** info@jump2learn.com | **Instagram :** www.instagram.com/jump2learn
Facebook : www.facebook.com/Jump2Learn | **Whatsapp :** +91-909-999-0960 | **YouTube :** Jump2Learn

PROCESS MANAGEMENT

- Process management is an integral part of any modern-day operating system (OS).
- The OS must allocate resources to processes, enable processes to share and exchange information, protect the resources of each process from other processes and enable synchronization among processes.

PROCESS MANAGEMENT SYSTEM:

- Business process management (BPM) is a discipline involving any combination of modeling, automation, execution, control, measurement and optimization of business activity flows, in support of enterprise goals, across systems, employees and customers within and beyond the enterprise boundaries.

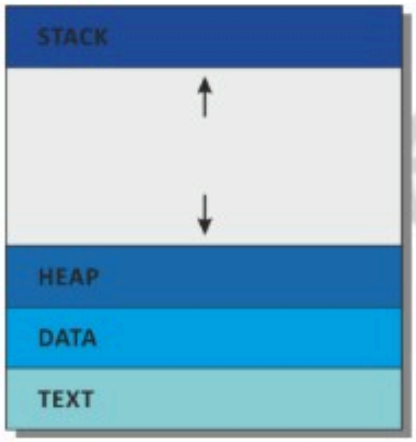
1.1 PROCESS CONCEPT

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

A process is defined as an entity which represents the basic unit of work to be implemented in the system.

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.
DATA	THIS INCLUDES THE CURRENT ACTIVITY REPRESENTED BY THE VALUE OF PROGRAM COUNTER AND THE CONTENTS OF THE PROCESSOR'S REGISTERS.
TEXT	THIS SECTION CONTAINS THE GLOBAL AND STATIC VARIABLES.

PROGRAM

A program is a piece of code which may be a single line or millions of lines. A computer program is usually written by a computer programmer in a programming language. For example, here is a simple program written in C programming language –

```
#include <stdio.h>

int main() {
  printf("Hello, World! \n");
  return 0;
}
```


A computer program is a collection of instructions that performs a specific task when executed by a computer. When we compare a program with a process, we can conclude that a process is a dynamic instance of a computer program.

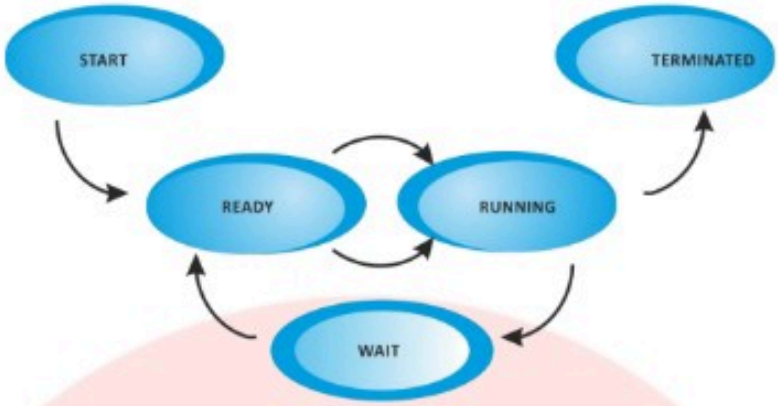
A part of a computer program that performs a well-defined task is known as an **algorithm**. A collection of computer programs, libraries and related data are referred to as a **software**.

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.

1. **Start** : This is the initial state when a process is first started/created.
2. **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.
3. **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.
4. **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.
5. **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.



OS-II 1.2 PROCESS LIFE CYCLE

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 in the table –

PROCESS STATE	THE CURRENT STATE OF THE PROCESS I.E., WHETHER IT IS READY, RUNNING, WAITING, OR WHATEVER.
PROCESS PRIVILEGE	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 REGISTER	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.
IO STATUS INFORMATION	THIS INCLUDES A LIST OF I/O DEVICES ALLOCATED TO THE PROCESS.

1.4 PROCESS CONTROL BLOCK

The architecture of a PCB is completely dependent on Operating System and may contain different information in different operating systems. Here is a simplified diagram of a PCB –

1.5 PCB

The PCB is maintained for a process throughout its lifetime, and is deleted once the process terminates.

1.2 PROCESS SCHEDULING

- The process scheduling is the activity of the process manager that handles the removal of the running process from the CPU and the selection of another process on the basis of a particular strategy.

- Process scheduling is an essential part of a Multiprogramming operating systems. Such operating systems allow more than one process to be loaded into the executable memory at a time and the loaded process shares the CPU using time multiplexing.

PROCESS SCHEDULING QUEUES:

The OS maintains all PCBs in Process Scheduling Queues.

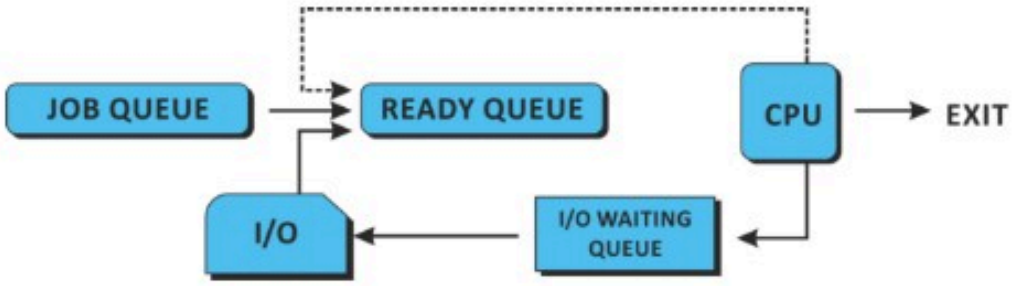
The OS maintains a separate queue for each of the process states and PCBs of all processes in the same execution state are placed in the same queue.

When the state of a process is changed, its PCB is unlinked from its current queue and moved to its new state queue.

The Operating System maintains the following important process scheduling queues –

JOB QUEUE – This queue keeps all the processes in the system.
READY QUEUE – This queue keeps a set of all processes residing in main memory, ready and waiting to execute. A new process is always put in this queue.

DEVICE QUEUES – The processes which are blocked due to unavailability of an I/O device represent this queue.



1.6 PROCESS SCHEDULING QUEUE

The OS can use different policies to manage each queue (FIFO, Round Robin, Priority, etc.). The OS scheduler determines how to move processes between the ready and run queues which can only have one entry per processor core on the system; in the above diagram, it has been merged with the CPU.

TWO-STATE PROCESS MODEL:

Two-state process model refers to running and non-running states which are described below –

Running:

When a new process is created, it enters into the system as in the running state.

Not Running:

Processes that are not running are kept in queue, waiting for their turn to execute. Each entry in the queue is a pointer to a particular process. Queue is implemented by using linked list.

When a process is interrupted, that process is transferred in the waiting queue.

If the process has completed or aborted, the process is discarded. In either case, the dispatcher then selects a process from the queue to execute.

SCHEDULERS

Schedulers are special system software which handle process scheduling in various ways.

Their main task is to select the jobs to be submitted into the system and to decide which process to run.

Schedulers are of three types –

- Long-Term Scheduler
- Short-Term Scheduler
- Medium-Term Scheduler

LONG TERM SCHEDULER:

It is also called a **job scheduler**. A long-term scheduler determines which programs are admitted to the system for processing.

It selects processes from the queue and loads them into memory for execution. Process loads into the memory for CPU scheduling.

The primary objective of the job scheduler is to provide a balanced mix of jobs, such as I/O bound and processor bound.

It also controls the degree of multiprogramming. If the degree of multiprogramming is stable, then the average rate of process creation must be equal to the average departure rate of processes leaving the system.

On some systems, the long-term scheduler may not be available or minimal. Time-sharing operating systems have no long term scheduler. When a process changes the state from new to ready, then there is use of long-term scheduler.

SHORT TERM SCHEDULER:

It is also called as **CPU scheduler**. Its main objective is to increase system performance in accordance with the chosen set of criteria. It is the change of ready state to running state of the process.

CPU scheduler selects a process among the processes that are ready to execute and allocates CPU to one of them.

Short-term schedulers, also known as dispatchers, make the decision of which process to execute next.

Short-term schedulers are faster than long-term schedulers.

MEDIUM TERM SCHEDULER:

Medium-term scheduling is a part of **swapping**. It removes the processes from the memory. It reduces the degree of multiprogramming.

The medium-term scheduler is in-charge of handling the swapped out-processes.

A running process may become suspended if it makes an I/O request. A suspended processes cannot make any progress towards completion.

In this condition, to remove the process from memory and make space for other processes, the suspended process is moved to the secondary storage.

This process is called **swapping**, and the process is said to be swapped out or rolled out. Swapping may be necessary to improve the process mix.

1.3 SCHEDULING CRITERIA

- There are several different criteria to consider when trying to select the "best" scheduling algorithm for a particular situation and environment, including:
  - **CPU utilization** - Ideally the CPU would be busy 100% of the time, so as to waste 0 CPU cycles. On a real system CPU usage should range from 40% (lightly loaded) to 90% (heavily loaded.)
  - **Throughput** - Number of processes completed per unit time. May range from 10 / second to 1 / hour depending on the specific processes.
  - **Turnaround time** - Time required for a particular process to complete, from submission time to completion. (Wall clock time.)
  - **Waiting time** - How much time processes spend in the ready queue waiting their turn to get on the CPU.
    - (**Load average** - The average number of processes sitting in the ready queue waiting their turn to get into the CPU. Reported in 1-minute, 5-minute, and 15-minute averages by "uptime" and "who".)
  - **Response time** - The time taken in an interactive program from the issuance of a command to the ***commence*** of a response to that command.
- In general one wants to optimize the average value of a criteria (Maximize CPU utilization and throughput, and minimize all the others.) However some times one wants to do something different, such as to minimize the maximum response time.
- Sometimes it is most desirable to minimize the ***variance*** of a criteria than the actual value. I.e. users are more accepting of a consistent predictable system than an inconsistent one, even if it is a little bit slower.

1.4 SCHEDULING ALGORITHMS

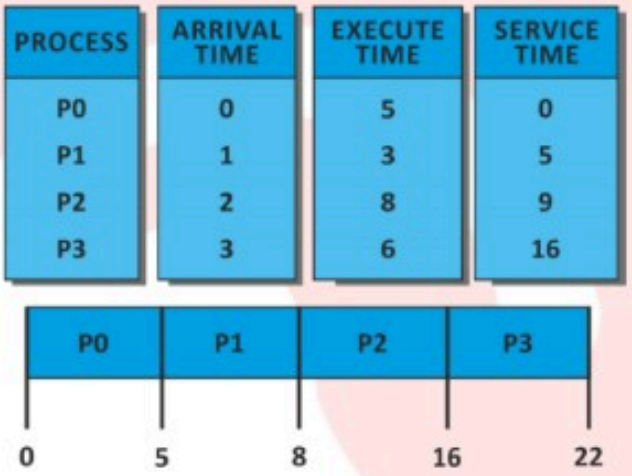
- A Process Scheduler schedules different processes to be assigned to the CPU based on particular scheduling algorithms.
- There are six popular process scheduling algorithms: –

1. First-Come, First-Served (FCFS) Scheduling
2. Shortest-Job-Next (SJN) Scheduling
3. Priority Scheduling
4. Shortest Remaining Time
5. Round Robin(RR) Scheduling
6. Multiple-Level Queues Scheduling

- These algorithms are either non-preemptive or preemptive. Non-preemptive algorithms are designed so that once a process enters the running state, it cannot be preempted until it completes its allotted time, whereas the preemptive scheduling is based on priority where a scheduler may preempt a low priority running process anytime when a high priority process enters into a ready state.

FIRST COME FIRST SERVE (FCFS):

- Jobs are executed on first come, first serve basis.
- It is a non-preemptive, pre-emptive scheduling algorithm.
- Easy to understand and implement.
- Its implementation is based on FIFO queue.
- Poor in performance as average wait time is high.



1.7 FIRST COMEFIRST SERVE (FCFS)

Wait time of each process is as follows –

PROCESS	WAIT TIME : SERVICE TIME - ARRIVAL TIME
P0	0 - 0 = 0
P1	5 - 1 = 4
P2	8 - 2 = 6
P3	16 - 3 = 13

Average Wait Time: (0+4+6+13) / 4 = 5.75

SHORTEST JOB NEXT (SJN):

- This is also known as **shortest job first**, or SJF
- This is a non-preemptive, pre-emptive scheduling algorithm.
- Best approach to minimize waiting time.
- Easy to implement in Batch systems where required CPU time is known in advance.

- Impossible to implement in interactive systems where required CPU time is not known.
- The processer should know in advance how much time process will take.

PROCESS	ARRIVAL TIME	EXECUTE TIME	SERVICE TIME
P0	0	5	3
P1	1	3	0
P2	2	8	14
P3	3	6	8

Gantt Chart:

Wait time of each process is as follows –

PROCESS	WAIT TIME : SERVICE TIME - ARRIVAL TIME
P0	3 - 0 = 3
P1	0
P2	14 - 2 = 12
P3	8 - 3 = 5

Average Wait Time: (3+12+5) / 4 = 5

PRIORITY BASED SCHEDULING:

- Priority scheduling is a non-preemptive algorithm and one of the most common scheduling algorithms in batch systems.
- Each process is assigned a priority. Process with highest priority is to be executed first and so on.
- Processes with same priority are executed on first come first served basis.
- Priority can be decided based on memory requirements, time requirements or any other resource requirement.

PROCESS	ARRIVAL TIME	EXECUTE TIME	PRIORITY TIME	SERVICE TIME
P0	0	5	1	9
P1	1	3	2	6
P2	2	8	1	14
P3	3	6	3	0

1.8 PRIORITY BASE SCHEDULING

Wait time of each process is as follows –

PROCESS	WAIT TIME : SERVICE TIME - ARRIVAL TIME
P0	9 - 0 = 9
P1	6 - 1 = 5
P2	14 - 2 = 12
P3	0 - 0 = 0

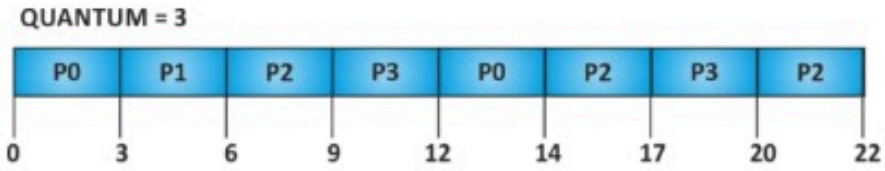
Average Wait Time: (9+5+12+0) / 4 = 6.5

SHORTEST REMAINING TIME:

- Shortest remaining time (SRT) is the preemptive version of the SJN algorithm.
- The processor is allocated to the job closest to completion but it can be preempted by a newer ready job with shorter time to completion.
- Impossible to implement in interactive systems where required CPU time is not known.
- It is often used in batch environments where short jobs need to give preference.

ROUND ROBIN SCHEDULING:

- Round Robin is the preemptive process scheduling algorithm.
- Each process is provided a fix time to execute, it is called a **quantum**.
- Once a process is executed for a given time period, it is preempted and other process executes for a given time period.
- Context switching is used to save states of preempted processes.



Wait time of each process is as follows –

PROCESS	WAIT TIME : SERVICE TIME - ARRIVAL TIME
P0	(0 - 0) + (12 - 3) = 9
P1	(3 - 1) = 2
P2	(6 - 2) + (14 - 9) + (20 - 17) = 12
P3	(9 - 3) + (17 - 12) = 11

Average Wait Time: (9+2+12+11) / 4 = 8.5

MULTIPLE-LEVEL QUEUES SCHEDULING:

Multiple-level queues are not an independent scheduling algorithm. They make use of other existing algorithms to group and schedule jobs with common characteristics.

- Multiple queues are maintained for processes with common characteristics.
- Each queue can have its own scheduling algorithms.
- Priorities are assigned to each queue.

For example, CPU-bound jobs can be scheduled in one queue and all I/O-bound jobs in another queue. The Process Scheduler then alternately selects jobs from each queue and assigns them to the CPU based on the algorithm assigned to the queue.