(Process Management)

2016 February 10 Wednesday

Process Concept :-

A process is the instance of a computer program that is being executed by one or many threads.

=) It contains the program code and its activity.

=) Depending on OS a process may be made up of multiple threads of execution that execute instructions concurrently.

=) Computer program is a passive collection of instructions often being loaded from the disk onto memory.

=) Several processes may be associated with the same program.

=) Multitasking the method to allow multiple processes to share CPU and other system resources.

=) A process is a program at the time of execution which include the process stack for the purpose of store temporary data.

* Process Vs Program

Process. Program

① It consist of instruction ① It consist of list of instruction in machine code. eg. in any programming language.

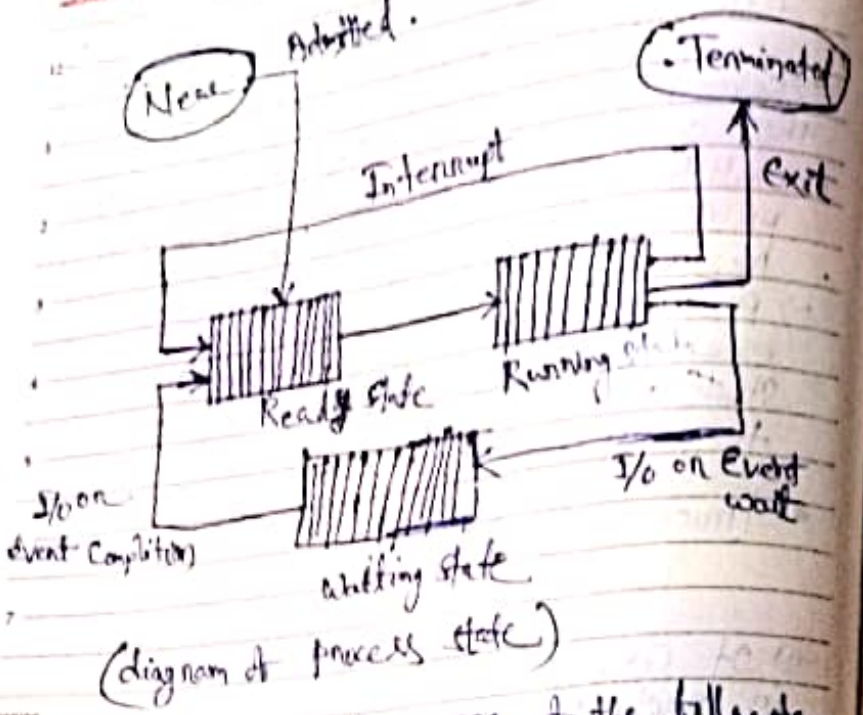
② Process is dynamic object. ② program is static object

Namrata Naik

Process

a) It resides in main memory.
b) Process span of time is limited.
c) It is a Active entity.
d) It results in secondary storage devices.
e) Program span of time is unlimited.
f) It is a passive entity.

Process State:-



(diagram of process state)

Each Process may be in one of the following states:-

1) New:- The process is being Created.

2) Running:- The process is being executed.
3) Waiting:- The process is waiting for some event to occur.
4) Ready:- The process is waiting to be assigned to a processor.
5) Terminate:- The process has finished execution.

Flow Diagram:-

New → Ready:- The OS create a process and prepare the process to be executed then the OS moved the process into ready queue.

Ready → Running:- When it is a time to select process to run, the OS selects one of the job from the ready queue and move the process from ready state to running state.

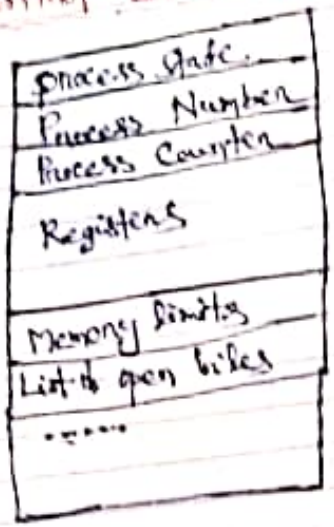
Running → Ready:- When the time slot of the process expired ex:- if any interrupt signal receive by processor then the OS shifted running process to ready state.

Running → Waiting:- If the process need an event occurs are an I/O device requires and OS does not provide immediately then

the model do waiting state by the OS.

Waiting → Ready:- A process in the blocked state is moved to ready state when it get the event or I/O device for which it has been waiting occurs.

Process Control Block (PCB)



⇒ A PCB is a data structure used by computer OS to store and the information about a process, it is also known as process description.

⇒ When a process is created the OS create the corresponding PCB.

⇒ The information in a process control Block is updated during the transition of

process state.

⇒ When the process terminates, its PCB is returned to the pool from which new PCBs are drawn.

⇒ Each process has a single PCB.

⇒ Though the structure of PCB is system dependent, common elements of PCB mainly fall in three main categories:-

1. Process Identification.
2. Process State.
3. Process Control.

Interacting Processes:-

⇒ Process interaction is a model of managing parallel or concurrent processes by determining how data between these processes is exchanged and how the processes are synchronized with each other.

⇒ The purpose of managing interaction between process is to ensure that there is a safe framework for managing parallel processes that are functionally link to each other.

⇒ This provide the ability to use data about interacting processes perform necessary operations and tasks.

⇒ Process interaction is the necessary condition for parallel processing.

Interprocess Communication (IPC)

are of two types:

→ Processes may be:
1. Independent Processes
2. Cooperating Processes

→ A process is independent if it cannot affect or be affected by the other processes executing in the system. Processes that do not share data with [illegible] any other process are independent.
→ A process is co-operating if it can affect or be affected by the other processes executing in the system.

→ Co-operating processes require an interprocess communication mechanism.

→ There are two fundamental models of IPC.

(1) Shared Memory
(2) Message Passing

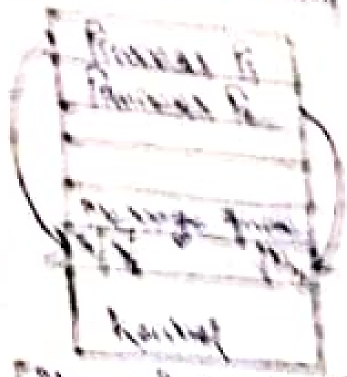
→ In a shared memory model, a region of memory that is shared by processes [illegible]. They exchange information by reading and writing data to the shared region.

→ In the message passing model, communication takes place by means of messages exchanged between the cooperating

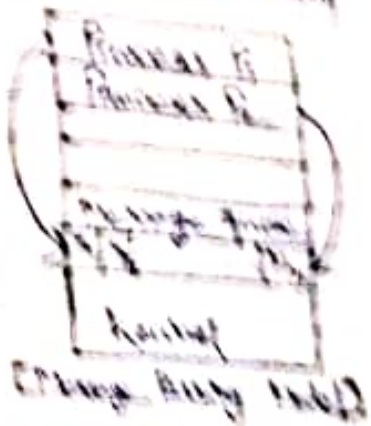
processes.

(Message passing model allows multiple processes to read and write data to the message queue without being connected to each other. Messages are stored on the queue until their recipient retrieves them.

→ Message queues are quite useful for interprocess communication and are used by most operating systems.



(Message Passing model)

Namrata [illegible]
Namrata [illegible]

Implementation Issue of Process —

⇒ To implement the process model, the operating system maintain a table (within one entry) called as process table. The table is called as Process table.

⇒ The entry contains the information about the process state, its program counter, stack pointer, memory allocation, the status of its open files, its accounting and the scheduling information.

⇒ In this table also store when the process is switched from running state to ready state or blocked state, so that it can be restarted later as if it had never been stopped.

⇒ Process Model is implemented by process table and process control block which keep track all information of process.

⇒ At the time of creation of a new process OS allocate a memory for load a process code.

⇒ The state of process is started as 'new' in its PCB and when this process move to ready state is also changes in PCB.

Evening

The various queue used for this which is implemented as linked list.

There are many following queue —

① Ready queue :- This queue is used for storing the process with state ready.

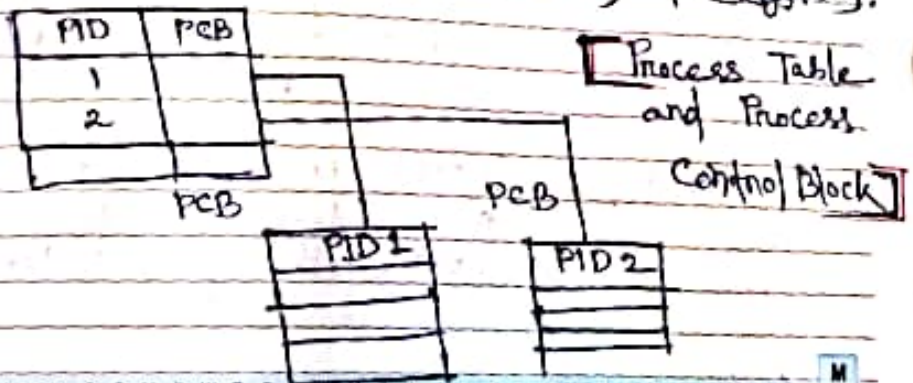
② Block queue :- This state store the process which need to wait (as an input, output devices.

③ Suspended Queue :- It is used for storing the blocked process that have been suspended.

④ Free Process Queue :- It is used for the information of empty space in the memory. [When new PCB Create]

Note

The Process table is an array of PCB that means logically a contain a PCB for all of the current processes in the system.



Evening

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 OS.

Process Scheduling Queues:-

⇒ The OS maintains the following important process scheduling queues:-

⇒ The OS maintains the following important process scheduling queues:-

① Job Queue → This queue keeps all the processes in the system.

② Ready queue → This queue keeps a set of all processes residing in memory, ready and waiting to execute. A new process is always put in this queue.

③ Device queue → The processes which are block due to unavailability of an I/O device.

⇒ 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 device queues.

Schedulers:-

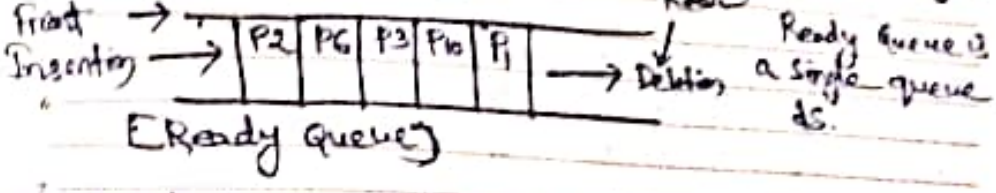
⇒ Schedulers are special system software which handle process scheduling in various ways.

⇒ Schedulers are of three types:-

① Long-Term Scheduler:-

⇒ It is also called a job scheduler.

⇒ The long term scheduler is to select the jobs from the pool of jobs and loaded these jobs into main memory (Ready queue of RAM).



② Short Term Scheduler:-

⇒ The function of the short term scheduler is, select a job from the ready queue and gives the control of the CPU to that process with the help of 'Despatcher'.

⇒ So it is also called CPU Scheduler.

⇒ The method of selecting a process from the ready queue (by the short term Scheduler) is depending on the CPU Scheduling algorithm.

① Dispatcher — A dispatcher is a module which connect the CPU to the process selected by the short term scheduler.

⇒ The main function of the dispatcher is switching the CPU from one process to another process.

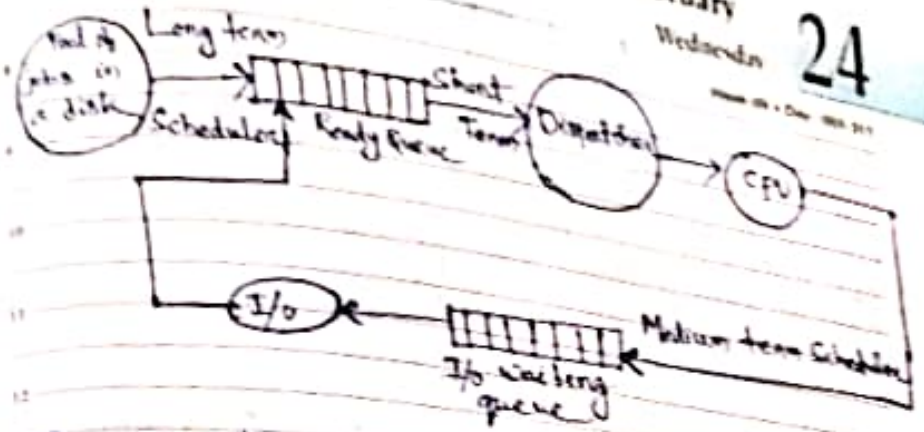
⇒ Another one is jumping to the proper location in the user program and ready to start execution.

Medium term Scheduler :—

⇒ If a process request an I/O in the middle of the execution, then the processor removed from the main memory and loaded into waiting queue.

⇒ When the I/O operation completed, then the job moved from waiting queue to ready queue.

⇒ These two operations performed by medium term scheduler.



[Queueing diagram for the CPU Scheduling]

CPU SCHEDULING ALGO :—

⇒ CPU Scheduling Algo. decides which of the processes in the ready queue is to be allocated the CPU.

① First-Come, First-Served Scheduling (FCFS)

- Arrival time = 0

Process	Burst time (in milliseconds)
P_1	5
P_2	24
P_3	16
P_4	10
P_5	3

⇒ It is the simplest CPU scheduling algo.

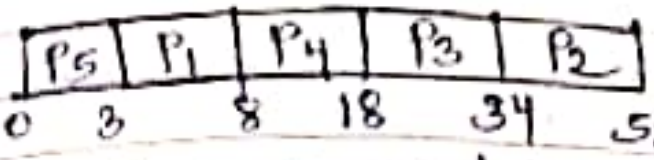
⇒ The criteria of this algo. is the process that request first the CPU first is holds the CPU first.

Shortest Job first Scheduling (SJF)

AT = 0

Process	CPU Burst time (Milli Sec)
P_1	5
P_2	24
P_3	16
P_4	10
P_5	03

The criterion of this is which process have smallest CPU burst, CPU assign that process next.



Avg. turn around time (FT − AT)

Turn around time for $P_1 = 8 - 0 = 8$
$P_2 = 58 - 0 = 58$
$P_3 = 34 - 0 = 34$
$P_4 = 18 - 0 = 18$
$P_5 = 3 - 0 = 3$

Avg. turn around time $= \frac{8+58+34+18+3}{5}$

$= \frac{121}{5} = \boxed{24.2}$ milli Sec.

AVG. waiting time (ST − AT)

Evening Waiting time for $P_1 = 3 - 0 = 3$
$P_2 = 34 - 0 = 34$
$P_3 = 18 - 0 = 18$
$P_4 = 8 - 0 = 8$
$P_5 = 0 - 0 = 0$

Namrata Nath

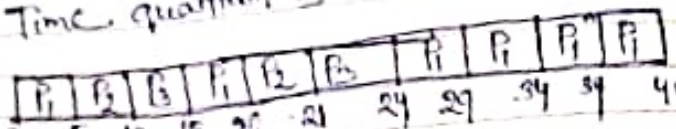
Avg. waiting time = $\frac{31+3+18+8+0}{5} = \frac{63}{5} = \boxed{12.6}$ millisec

Round Robin Scheduling Algorithm:-

This algo. is designed especially for time sharing systems. In this algo. the CPU switches between the processes. When the time quantum expired, the CPU switches to another job.

Process	CPU Burst time (in millisec)
P_1	30
P_2	06
P_3	08

Time quantum is 5 millisec. (5ms)



Avg. turn around time :- (FT−AT)

Turn around time for $P_1 \rightarrow 44-0 = 44$

Turn around time for $P_1 \rightarrow 44-0 = 44$

$P_2 \rightarrow 21-0 = 21$

$P_3 \rightarrow 24-0 = 24$

The avg. turn around time →

$\frac{44+21+24}{3} = \frac{89}{3} = \boxed{29.66}$ millisec.

Avg. waiting time = (ST−AT)

Waiting time for $P_1 = 0+(15-5)+(24-20)$

$= 0+10+4$

$= 14$

$P_2 = (5-0)+(20-10) = 5+10 = 15$

$P_3 = (10-0)+(21-15) = 10+6 = 16$

Avg. waiting time →

$\frac{14+15+16}{3} = \frac{45}{3} = \boxed{15}$ millisec.

Priority Scheduling :-

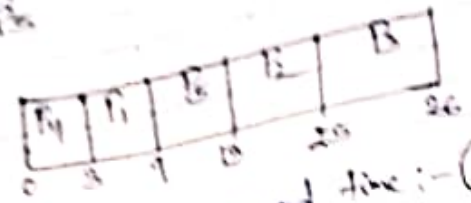
⇒ Priorities are generally some fixed range of numbers such as 0 to 4095.

⇒ The CPU is allocated to the process with the heightest priority.

⇒ Equal priority processes are scheduled in FCFS order.

Notes

Process	CPU Burst time (milliseconds)	Priority
P1	6	2
P2	8	4
P3	7	3
P4	3	1
P5	4	3



The avg. turn around time :- (T.T - A.T)

Turn around time for P1 = 9 - 0 = 9
P2 = 25 - 0 = 25
P3 = 26 - 0 = 26
P4 = 3 - 0 = 3
P5 = 13 - 0 = 13

Avg. turn around time :-

$\frac{9+25+26+3+13}{5} = \frac{76}{5} = 15.2$ msec.

The avg. waiting time = (BT - AT)

waiting time for P1 = 3 - 0 = 3
P2 = 13 - 0 = 13
P3 = 25 - 0 = 25
P4 = 0 - 0 = 0
P5 = 9 - 0 = 9

Notes

Avg. waiting time = $\frac{3+13+25+0+9}{5}$

$= \frac{50}{5} = 10$ msec.

Process Synchronization :-

- It is the task phenomenon of co-ordinating the execution of processes in such a way that no two processes can have access to the same shared data and resources.
- It is a procedure that is involved in order of execution of co-operative processes.
- In order to synchronize the processes there are various synchronization mechanism).
- Process Synchronization is mainly needed in a multiprocess system when multiple processes try to gain access to the same shared resources or any data at the same time.

Critical - Section :-

In a system consisting of 'n' processes and each process having a segment of code in which the process may change common variables, updating a table, writing a file and so on. This segment

Alernate Nath

A Code is said to be Critical Section.

Solution:- Sol" should satisfy the following three requirements:-

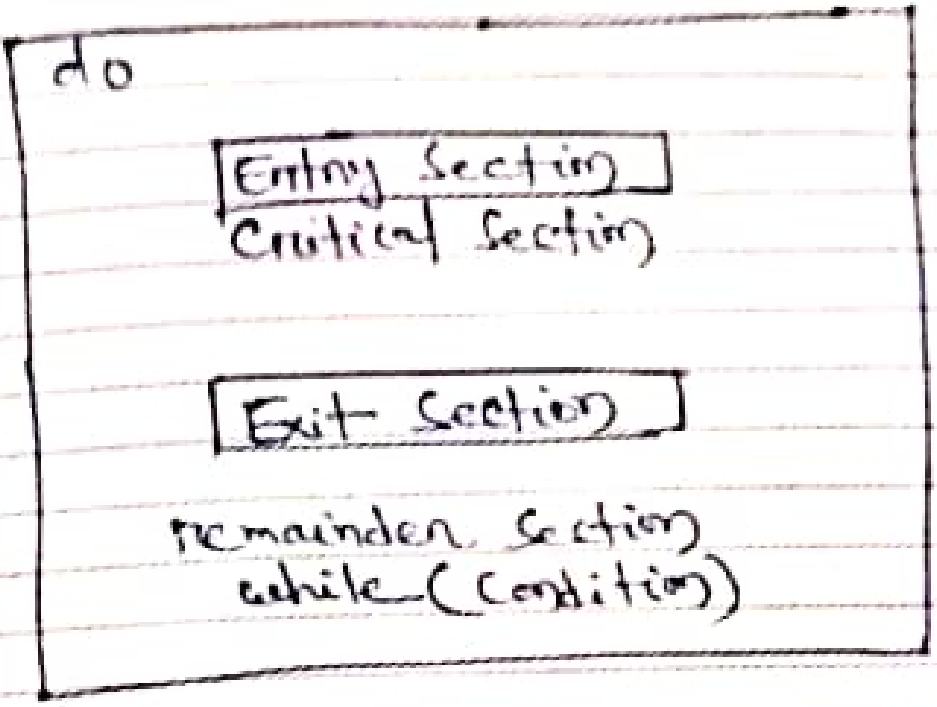
① Mutual Exclusion:-> only one process can execute their critical sections at any time. No other processes can be executing in their critical sections. #1

② Progress:-> If a process is executing in its critical sections and there exist some processes that wish to enter their critical sol", then only those processes that are not executing in their remainder section can participate in the decision of which will enter its critical section next, and this selection cannot be postponed indefinitely. #2.

③ Bounded Waiting:-
There exist a bound on the number of times that other processes are allowed to enter their critical sections after a process has made a request to enter its critical section and before that request is granted.

Shared	P1	P2
a. Shared variable: x=5;	1. int x = Shared;	1. int y = Shared;
• memory	2. x = x+1;	2. y = y-1;
• method	3. sleep(1);	3. sleep(1);
• Resources ← CPU ← Printer	4. Shared = x;	4. Shared = y;

Namrata Naik



*1 Requirements for Mutual Exclusion →

→ Mutual Exclusion must be enforced, only one process at a time is allowed into its Critical section among all processes that have critical sections for the same resource or shared object.

→ A process that halt in its Critical Section must do so without interfering with other process.

→ It must not be possible for a process requiring access to a Critical Section to be delay indefinitely, no deadlock or starvation can be allowed.

SEMAPHORES:

The solution of a critical section problem is a semaphore.

=> This is a synchronization tool.

=> A semaphore is a variable used to control access to a common resource by multiple processes.

=> It avoids critical section problems in a concurrent system such as a multitasking operating system.

=> This integer variable, apart from initialization, is accessed only through two standard atomic operations:

* wait
* signal.

P(S) ↓ wait

wait (S) : Decrement value of the argument S, as soon as it would become non-negative.

wait (S) : While S<=0 do no-op
S = S-1

V(S) ↓ signal

signal (S) : Increments the value of argument S is an individual operation.

signal (S) : S = S+1.

=> If the semaphore is allowed to take the values 0 and 1 is said to be binary semaphore.

=> It is very convenient for inter process synchronization.

Principle of Concurrency:-

Both interleaved and overlapped processes can be viewed as concurrent processes and they both present the same problems.

The relative speed of execution can not be predicted and depend on the following:

* The activities of other processes.
* The way OS handle interrupts.
* The scheduling policies of the OS.

Advantages:-

1) Better resource utilization.
2) Better average response time.
3) Better performance.

Disadvantages:-

* Blocking
* Deadlock.
* Starvation
* Need to protect multiple appl^n from one another.

02 March Wednesday 2016

SEMAPHORES:-

The Solution to a Critical-Section problem is a Semaphore.
This is a Synchronization tool.
A semaphore is a variable used to control access to a common resources by multiple processes.
It avoid critical section problem in a concurrent system such as a multitasking operating system.
This integer variable apart from initialization is accessed only through two standard atomic operation.
* wait * signal.

wait(S): Decrement value of its argument S, as soon as it would become non-negative.

wait(S): While S<=0 do no-op
S=S-1

signal(S): Increments the value of argument S is an individual operation.

signal(S): S=S+1.

=> If the semaphore is allowed to take the values 0 and 1 is said to be binary Semaphore.

Evening

=> It is very convinient for inter process Synchronization.

Namrata Naik

03 March Thursday 2016

Principle of Concurrency:-

* Both interleaved and overlapped processes can be viewed as concurrent process, and they both present the same problems.

The relative speed of execution can not be predicted and depend on the following

* The activities of other processes.
* The way of handle interrupts.
* The Scheduling policies of the OS.

Advantages:-

1. Better resource utilization.
2. Better average response time.
3. Better performance.

Disadvantages:-

* Blocking
* Deadlock
* Starvation
* Need to protect multiple appl^{n} from one another.

Evening

Namrata Naik