TUTORIALSDUNIYA.COM

Operating Systems Notes

Delhi University

Download TutorialsDuniya App from PlayStore

Visit https://www.tutorialsduniya.com for Notes, Tutorials, Programs, MCQs, Quiz etc.

- **FaceBook: https://facebook.com/tutorialsduniya**
- **YouTube: https://www.youtube.com/user/tutorialsduniya**
- **LinkedIn: https://www.linkedin.com/company/tutorialsduniya**
- **InstaGram: https://www.instagram.com/tutorialsduniya**

Operating System :-

Operating system is a set of programs used to provide interface between the computer user and computer hardware.

Operating system also controls the computer hardware and provide coordination among these hardware devices.

Ex:- print a document.

Main objectives considered to design an operating system are :-

↳ Convenience

In this the operating system provides interaction to the user to access a computer easily and quickly.

↳ Efficiency

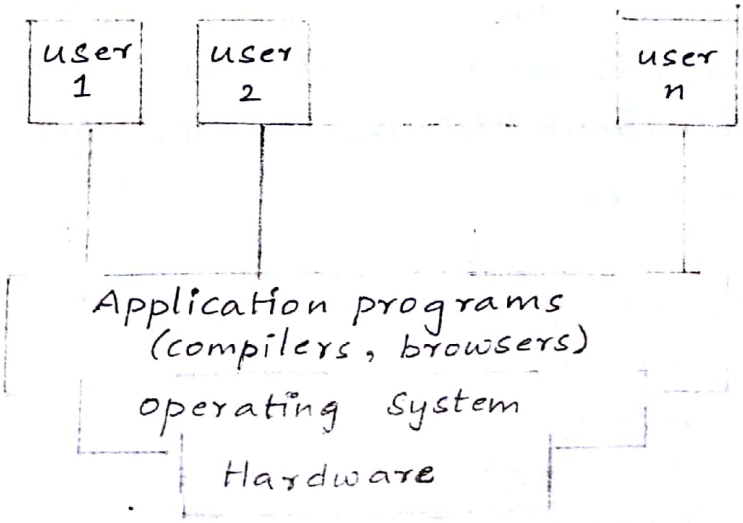
The operating system can do large work in less number of time.

↳ Ability to Evolve

In this the operating system provides a way for future enhancements and modifications.

Role of Operating System :-

We need to discuss about logical components of a computer to know the actual role of an operating system.



Hardware :-

It consists of CPU, memory, I/o devices, USB port, LAN port etc. All these computer hardware used by operating system to solve a user task.

Operating System :-

It is used to provide interface between application program and hardware. Here the operating system need to provide coordination among devices various hardware devices to solve a user task.

Application programs :-

These provides a way to the user to solve given task using computer system.

Users :-

These are end users of computer hardware. To know in detail about operating system role we need to discuss operating system in two point of views. They are

↳ User point of view
↳ System point of view

(1) User point of view :-

In this the role of OS can be varied based on user situated or worked

↳ User sit/work with stand alone PC

In this the user feel about OS, it provides convenience. The OS is used to complete user task conveniently and quickly. But the user don't bother about resource utilisation and system performance.

↳ User connected to main frames through dummy terminal.

In this the user can observe that the operating

System provides resource sharing among multiple users connected to the main frame.

↳ User work with PC connected to network

In this user can observe that os allows to access the services provided by network like network connections, access data by a remote system connected to the network.

(2) System point of view :-

In this the computer system mainly concentrate on efficiency that is high performance. The system always try to provide continuous service to the users. For this continuity of services it must be concentrated on resource validation. The operating system acts as resource manager and it performs following two activities.

① Resource Allocation :-

In this the operating system plays a vital role while allocating system resources to the processor or programs execution.

② Resource Control :-

In this the os specifies which resource can do this work.

Memory Hierarchy :-

In computer systems memory can be used to store data and programs. All the memory devices are put in a hierarchical order based on following criteria.

↳ Size of memory device
↳ Accessing speed of memory device
↳ Cost per 1 bit of memory.

Size can be increased from top to bottom, speed

and cost can be decreased from top to bottom.

Memory devices can be classified based on volatile ability into two categories.

↳ Volatilable devices

In this content can be lost when the power supply is OFF.

↳ Non-Volatilable devices

In this content cannot be lost when the power supply is OFF.

Memory devices can be divided into two categories based on accessibility

↳ Random Access devices

In this data can be directly accessed from memory device.

↳ Sequential Access devices

In this data can be accessed in a sequential manner.

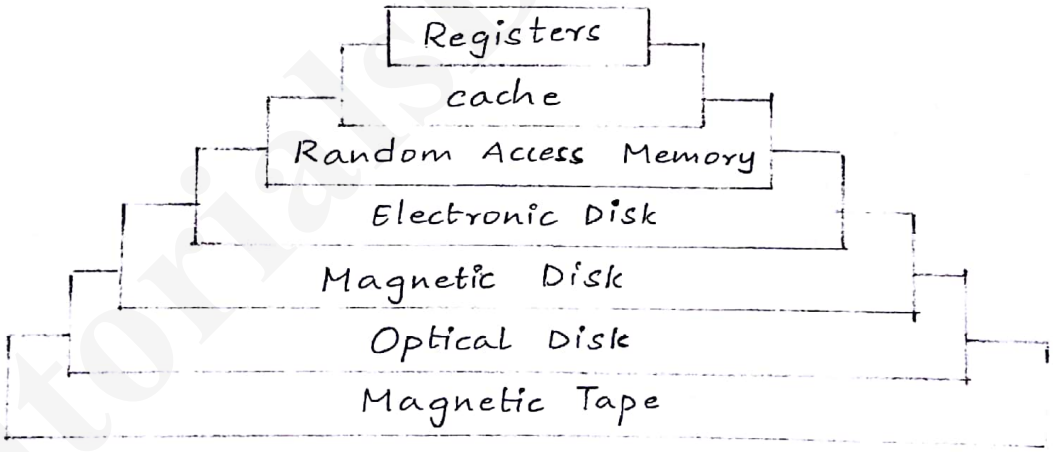


Fig: Memory Hierarchy

Registers :-

Registers are used by the CPU for its internal purpose while executing programs. In registers there are two types.

↳ General purpose Registers

Used to store input data while performing arithmetic operations. Ex:- Accumulator

↳ Control Registers

Used to store status of the program execution.

Ex:- PC, IR, MAR, MDR.

Cache Memory :-

It is placed between registers and main memory. It is used to store data. recently used by the processor. It increases accessing speed.

Random Access Memory :-

It is also called as main memory. It is used to store program and data from secondary memory before starting the program execution. In this data can be accessed directly by specifying the address.

Electronic Disk :-

It is also called as flash memory. It acts as RAM until the power supply is ON. Once the power supply is OFF it uses internal backup power supply [battery] to store the current data in magnetic disk. So it is called as volatile and Non-volatile memory device.

Magnetic Disk :-

It is also called as secondary memory. It is used to store all the programs. It divides into sectors and tracks to store the data. It is a non-volatile device. In general it is used to provide battery.

Optical Disk :-

It is called as removable memory. It is used to carry data along with the user. It is manufactured using optical technology.

Ex:- CD-ROM, DVD

Magnetic Tape :-

It is the oldest form of memory device. On this the data can be read/write in a sequential order. In this the tape can be divided into two tracks horizontally and stores the data in track 1 and track2 separately. Track 1 data can be accessed in forward direction and track2 data can be accessed in reverse direction.

Computer Architecture :-

A computer's system architecture can be categorized depending upon the availability of general purpose ~~registers~~. processors.

- ↳ Single processor systems
- ↳ Multi processor systems.

Single processor systems :-

On a single processor system, there is one main CPU capable of executing a general-purpose instruction set, including from user-processors.

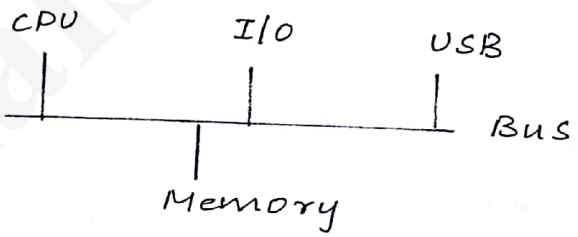


Fig : Uniprocessor system

- ↳ Less throughput compared to multiprocessor systems.
- ↳ Low reliability
- ↳ Resource sharing is not possible
- ↳ More cost.

Multi-processor Systems :-

On a multi-processor system, there will be two or more number of general purpose processors.

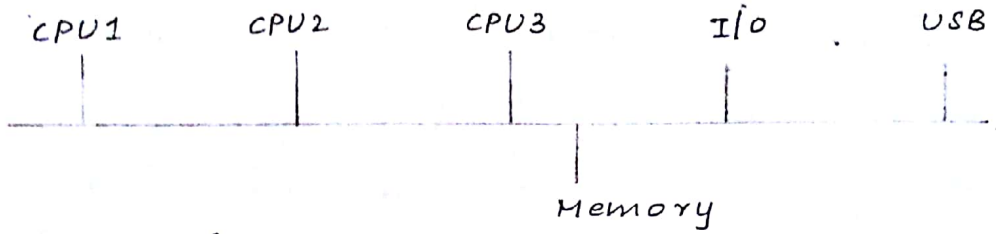


Fig : Multi-processor system.

There are two types of multi-processor systems. They are

↳ Asymmetric Multi-processor System.
↳ Symmetric Multi-processor system.

Asymmetric Multi-processor Systems :-

One among all the processors acts as master. other CPU's acts as slaves and perform the operations instructions instructed by the master and at the same time monitors the master processor. After the failure of the master the processor that monitors the master first will become Master.

Symmetric Multi-processor System :-

Here there is no master-slave relation. All the processors performs their tasks and monitors their neighbouring processors.

↳ More throughput
↳ More reliability
↳ Less cost
↳ Resource sharing is possible.

Clustered Systems :-

Gathering of multiple uni-processors together called clustered systems.

These systems also categorized into two types. They are

↳ Asymmetric clustered system
↳ Symmetric clustered system.

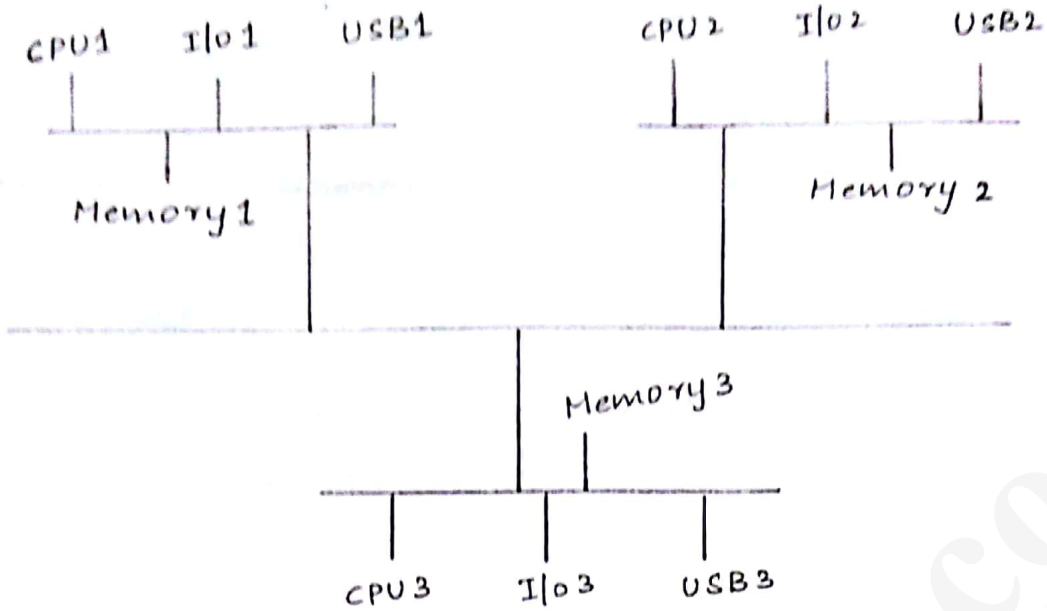


Fig : clustered systems

Asymmetric clustered systems :-

One among the uni-processors acts as master and one uni-processor only monitors the master and the remaining uni-processors acts as slaves and perform the operations instructed by master. After the failure of master the uni-processor that is monitoring the master becomes the master.

Symmetric clustered Systems :-

All the uni-processor systems performs their tasks and at the same time monitors the neighbouring uni-processor systems.

Operating System Structure :-

Operating system task is to perform the user's program. Operating system structure is based on no. of process.

↳ Single processing operating system
↳ Multi-programming operating system
↳ Multi-Tasking operating system (OR) Time shared systems.

Single processing Operating system :-

In single processing operating system only one program is available in memory if there is any interrupt the CPU will be idle. A single user cannot keep either the CPU or the I/o devices busy at all the times.

Multi-programming Operating System :-

Multi-programming increases CPU utilization by organizing jobs (code and data) so that the CPU always has one to execute. The operating system keeps several jobs in memory simultaneously. This set of jobs can be a subset of the jobs kept in the job pool which contains all the jobs that enter the system. Since the number of jobs that can be kept simultaneously in memory is usually smaller than the number of jobs that can be kept in the job pool. The operating system picks and begins to execute one of the jobs in memory. Eventually, the job may have to wait for some task, such as an I/o operation, to complete. In a multi-programmed system, the operating system simply switches to, and executes, another job. When that job needs to wait, the CPU is switched to another job, and so on. Eventually, the first job finishes waiting and gets the CPU back. As long as atleast one job needs to execute, the CPU is never idle.

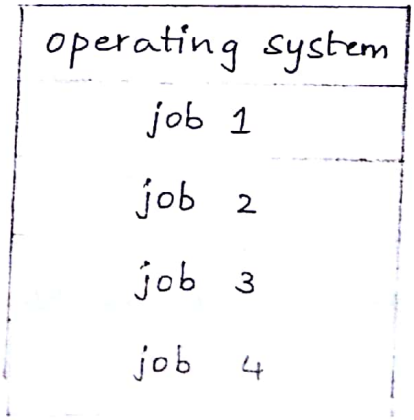


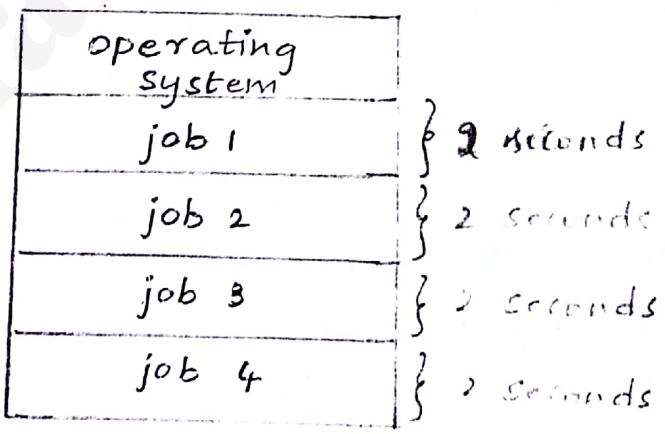
Fig: Memory layout for a multiprogramming system

Multi-Tasking Operating System / Time-shared Systems:-

Time sharing (or multi tasking) is a logical extension of multiprogramming. In time-sharing systems, the CPU executes multiple jobs by switching among them, but the switches occur so frequently that the users can interact with each program while it is running.

A time shared operating system uses CPU scheduling and multiprogamming to provide each user with a small portion of a time-shared computer. Each user has atleast one separate program in memory. A program loaded into memory and executing is called a process. The time that is allocated for single process is called Quantum.

Also, time sharing systems provide a mechanism for protecting resources from inappropriate use and it may ensure that jobs do not get stuck in a deadlock, forever waiting for one another.



Operating system operations:-

Operating system can execute mutiple programs In multi-Programmi os the error occured in one program can effect other user program some times effects the system programs also due to the sharing of single resource. To avoid this draw back we must ensure that design a proper operation of operating system. The proper operation of os is provided by the following two ways.

Dual mode operation:-

In this the operating system controls the cpu and provides two modes of process execution. They are.

↳ User mode:-

It is used to execute user process only

↳ Kernel mode / os mode / system mode / Priviliged mode:-

In this mode a cpu executes privilaged programs / os programs

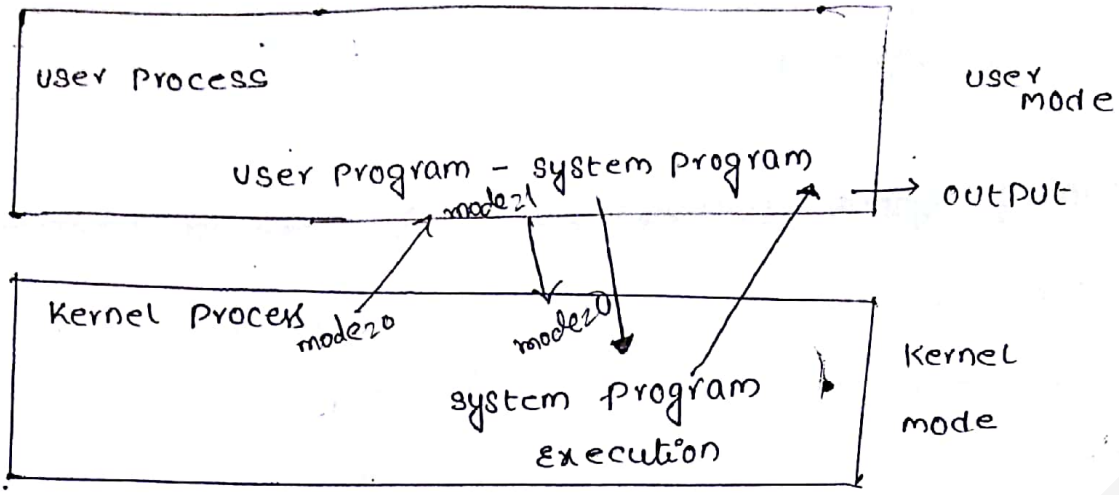
* By default the cpu kept in user mode at the starting of computer system
* A computer system follows a hardware approach to differentiate between user mode and kernel mode.

Hardware provides a bit register called as mode.

Ie

Mode = 0 → Kernel mode

Mode = 1 → User mode



The os initially executes an application program in user mode. In application mode It needs to execute a system Program then the cpu converts into kernel mode and executes the system programming

After system program execution It returns into usermode and executes the remaining application program.

2,

Timer mode Operation:-

This ensures that it prevents the following 3 activities

* Enter into a infinite loop for longer period
* Not returns the program control to os
* Not access the operating system services from user programming

In this the operating system used the concept of counter. counter is a integer initialized to fixed or variable values. The counter value is decremented by every clock 'T.' when The counter value becomes '0' then os generates an interrupt to the process. once interrupt occured during the Process execution the CPU returns the control to kernel mode.

Distributed systems:-

A set of physically seperated hetrogenous computer systems through a network.

Distributed system provides following benefits:-

* Speed of process execution with the concept of parallel execution of concurrency.
* share the resources available at one system to onther system
* provides reliability.

It a distributed system considers the concept of network is called as network distributed system.

It provides the following two features

* Share the resources that are unique to one system
* Provides comm. blw any two process that are executed on two seperated computer system.

Intra Process:-

The program execution is done on only one system

Inter Process:-

The o/p of program executed in one system is given as i/p to another process in the second system.

* In order to provide communication between processors.

*

This communication can be implemented by sending or receiving messages from one process to another process.

The N/w distributed systems are classified based on the distances between the systems.

* SAN [Small Area N/w]

It represents collection of systems in a room

* LAN [local Area N/w]

It represents collection of systems in an organization.

* MAN [metropolitian Area N/w]

It represents collection of systems in a city

* WAN [wide Area N/w]

It represents collection of systems throughout the world.

Special Purpose Systems:-

These systems are need to perform limited activites or functions. There are 3 types. They are

Realtime embedded system:-

embedded system is a s/w embedded into a real time s/w. It is used to control the operation of electronic devices.

ex:- washing machines, DVD players.

Real - time system:-

It has a rigid time constraint to process the task execution It must complete the task at specified time

ex:- Fuel injection, Remote bombs.

multimedia system:-

It can handle or processes multiple medias Image, videos, audio's. Now a days all the os are multi-media systems.

Handled system:-

These are also called as personal digital assistants (PDA) These are portable and easy to handle with hand.

Ex:- Mobiles, Laptops etc...

The size is very small when compared to PC

It has 3 drawbacks compared PC

* It provides less memory space
* It doesn't includes high speed processer
* It doesn't provide that much & user interaction as provided by PC.

Operating System Services / functions of Operating System:-

Operating system provides two sets of services one set of services provided for the convenience of programer, to make the Program development task easier. In this set it provides following 6 services.

* User Interface:-

It Provides interaction with the compiter system. In us it Provides two types of user interfaces.

a. character user interface:-

In this the os allows the user to enter commands using keyboard only.

(b) Graphical user interface:-

In this the os allows the user to use pointing devices to display menu's and select suitable options.

* Program execution:-

In this os Provides an environment to start up the Program execution. Before starting of program execution it must be loaded into main memory and linked with all the library file

* File system management:-

In this os can perform the following operations related to file system.

a) Create or open a new file
b) Read or write data on files
c) Modifies file content
d) Display status information of file
e) Delete a file.

TutorialsDuniya.com

Get FREE Compiled Books, Notes, Programs, Books, Question Papers with Solution* etc of following subjects from https://www.tutorialsduniya.com.

- C and C++
- Programming in Java
- Data Structures
- Computer Networks
- Android Programming
- PHP Programming
- JavaScript
- Java Server Pages
- Python
- Microprocessor
- Artificial Intelligence
- Machine Learning
- Computer System Architecture
- Discrete Structures
- Operating Systems
- Algorithms
- DataBase Management Systems
- Software Engineering
- Theory of Computation
- Operational Research
- System Programming
- Data Mining
- Computer Graphics
- Data Science

- Compiled Books: https://www.tutorialsduniya.com/compiled-books
- Programs: https://www.tutorialsduniya.com/programs
- Question Papers: https://www.tutorialsduniya.com/question-papers
- Python Notes: https://www.tutorialsduniya.com/python
- Java Notes: https://www.tutorialsduniya.com/java
- JavaScript Notes: https://www.tutorialsduniya.com/javascript
- JSP Notes: https://www.tutorialsduniya.com/jsp
- Microprocessor Notes: https://www.tutorialsduniya.com/microprocessor
- OR Notes: https://www.tutorialsduniya.com/operational-research

(*) I/O operations:-

In this os can attach or deattch I/O devices to the user program and controls I/O operations.

a) sending data to the I/O devices [monitor / printer]

b) Recive data from the I/O devices [key board]

c) os can provides interaction with I/O devices to the user progra

* Memory management:-

In this os provides services either allocate memory to store data or deallocate memory dynamically.

* communication:-

In this the os provides services to implement communicatio blw any two processes that are either executed on the same system or two different systems. In this the communication i done by sending or recieving data packets or messages. It als uses shared memory concept to provide communication blw two processes.

* Error Detection:-

In this os provides debugging service. os can detec errors in user program and display the information about errors to the programmer.

a) The error occured due to the failure of system resourc or power failure is called as Hardware error.

The error occured due to logic incorrect is called s/w erro

ex: Invalid memory address, devided by zero, ... etc

Second set of services:-

Another set of services provided for the purpose of syste to increase system's efficiency

Resource Allocation:-

In multi-programming environment, two or more programs are executed by allocating cpu time among these programs in a optimised manner. In this os can make decisions about resource allocation to process and he collect resources from the process

Accounting:-

In this os can maintain accounts about resource allocation ie, how many resources are allocated to which process or user. Their type of record keeping is useful

* to develop statistics about resource utilisation
* to make resource allocations in future

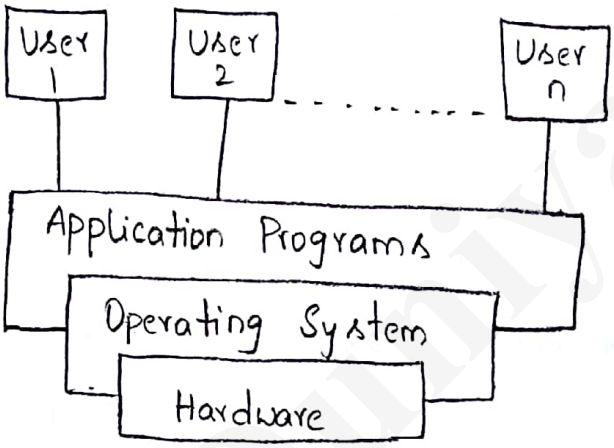
Protection and security:-

In this os provides protection and security on computer resources.

* Control the resource accessing among authorized user is called as protection
* Prevents the accessing of system resources by unauthorized users called as security.

Operating System Structure :-

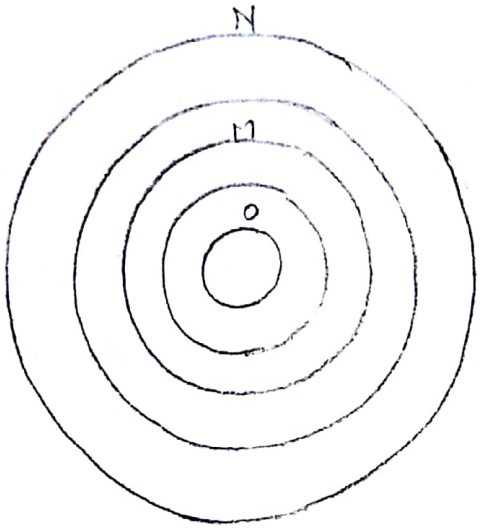
At the earliest days of computer OS available as a set of programs. It is called as Monolythic operating system. MS-DOS can use this type of structure. In this OS size is very simple and it perform limited set of functionalities.



If OS size becomes large it is complex to access a specific service from the set of all the services. To reduce the complexity the designers need to design OS in appropriate way for the proper operation of OS. The designers can use two approaches to modularize the OS.

1. Layered Approach :-

In this OS programs can be partitioned into small parts. Each part can be called as a layer. Each layer consists of set of functions and data to perform particular operation.



The Zeroth layer represents the CPU scheduling. N^{th} layer represents users of Computer System. Layer M implemented by using services provided by its lower level layers. Upper level layer of layer M access the services provided by layer M.

Benefits:-

1. It is easy to construct
2. It is simple to debug and verify the errors.

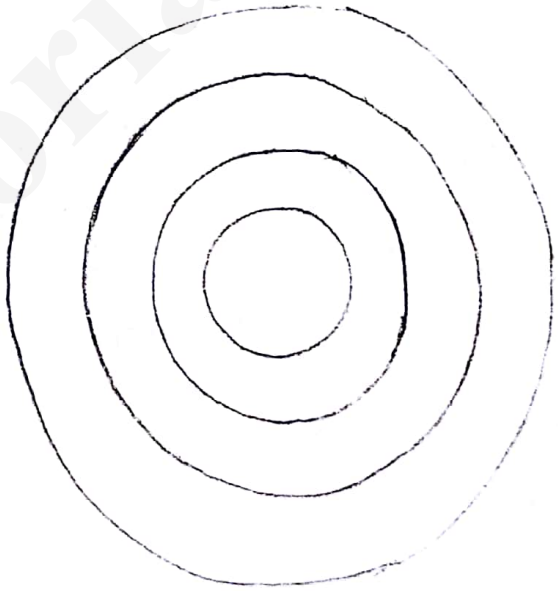
In this Zeroth Layer Compile and Correct all the errors first. Then 1st layer develop its programs using Zeroth layer programs. If there occurs in error in layer1 program it is compulsorily an error in layer 1 but not in layer 0.

Drawbacks:-

1. It is a complex task for the designer to design a layer appropriately by including features need by all the upper level layers.
2. In this efficiency is less due to a layer may need to interact with all the middle layers to communicate with zeroth layer (CPU).

Micro kernal Approach:-

To increase efficiency compared to layered approach we can use micro kernal approach. In this OS divided into 2 sets. First set contains essential programs of OS used to boot a OS. Second set of non-essential programs of OS used for process execution.



In this Microkernal provides a functionality i.e., communication between user and corresponding server. A user must interact with microkernal for service rather than directly communicate with server.

Benefits:-

1. It is easy to extend OS by adding new features to the system space instead of kernel.
2. It is easy to make changes in the kernel. This size is small and it is seperate from system programs.
3. It is easy to port kernal from one type of configuration to another.
4. It provides high security and reliability.

Drawbacks:-

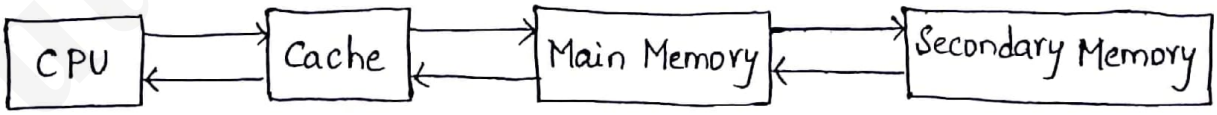
1. In this also efficiency is decreased to some extent due to functionality of the kernel i.e., provides the communication between user and server.

Cache Memory:-

Cache memory improves computer system performance.

1. Cache is a memory device placed between CPU and main memory.
2. Cache is high speed memory device with limited size. So, a program before start its execution placed in main memory. At execution time the program and data placed in cache memory because the CPU need not wait until data read from main memory.

The CPU first look in cache memory for the required data. If it is found there use it, otherwise the CPU look in main memory for the data and the data is available there, CPU copies the data into cache memory by assuming that the current used data will be required in the near future.



A cache memory provides fast accessing of data compared to main memory and secondary memory. So, a cache memory improves system performance.

Operating System Generation:-

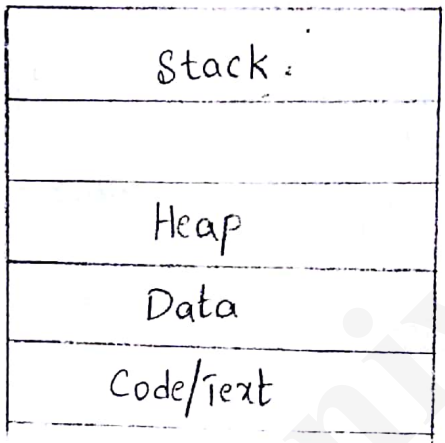
In general OS is available to run at any sight on any class of computer with any type of hardware configurations. This general operating system can be configured (or) generated suitable for particular configuration is called Operating System generation. The operating system generation can be done by using a program SYSGEN (System generation) This program can ask the administrator following for his to know. the details about hardware configuration of the computer.

1. What is the type of CPU?
2. How much memory space is available and how many partitions are needed?
3. How many I/O devices available? Type of I/o devices
4. What are the OS features to be used etc.....

These details are used by the SYSGEN to make modifications in the source code of OS to initialize values, modify variables etc. After do all the required modifications in the source code it recompiles the OS

Process :-

Process means program in execution. OS can allocate required memory space for program execution. The process have a predefined structure as shown below:



In this process memory space is divided into 4 segments.

1. Code/Text Segment :- Used to store actual program. It is fixed size segment. Its size cannot be changed when the process executed.

2. Data Segment :- It is used to store data values related to the program like variable initializations etc.

3. Stack Segment :- It is used to store status of currently executed program before starts the execution of subprocess in middle. The status information is content of accumalator, program counter, instruction register, MAR, MDR and all the

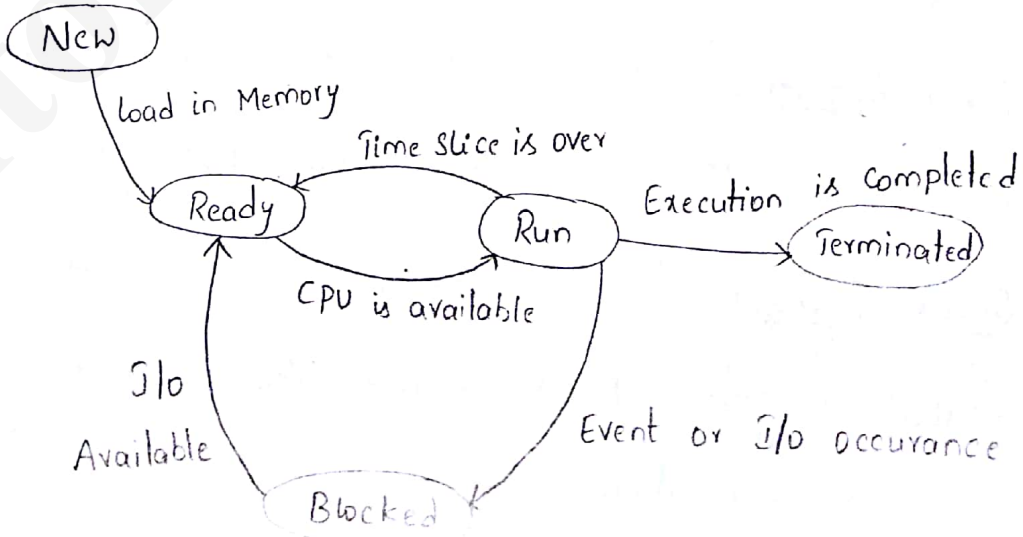
general purpose registers. Once sub process execution is completed the information stored in stack is retrieved and continuous the remaining process execution.

4. Heap Segment :- It is used to store temporary data. It is also used by data segment to store extra data and also used by stack segment to store status information of a process if stack segment is full.

Process State Diagram :-

Process States :-

A process may change its state during process execution. Process state describes position of the process at particular time. A process possible to placed in following states during its execution.



1. New State:- A process created newly put in new state.

2. Ready State:- A process entered into ready queue by allocating memory.

3. Run State:- A processor is to be allocated to a process.

4. Terminate State:- A process completes its execution.

5. Blocked State:- A process to which the required I/O is not available (or) receive an event (or) signal.

Navigations:-

1. New - Ready:- The process will go from new state to ready state when it is load in memory.

2. Ready - Run:- The process will go from ready state to run state when CPU is available.

3. Run - Ready:- The process will go from run state to ready state when time slice is over.

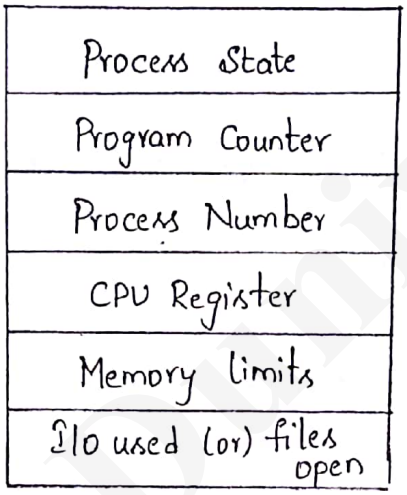
4. Run - Blocked:- The process will go from run state to blocked state when an event or I/O occured.

5. Run - Terminated:- The process will go from run state to terminated when its execution is completed.

6. Blocked - Ready:- The process will go from blocked state to ready state when I/O is available.

Process Control Block (PCB):-

It is used to store status information of a process. It is used by operating system to control the process execution. It is also called as Job Control Block. A process control block contains following information of process:



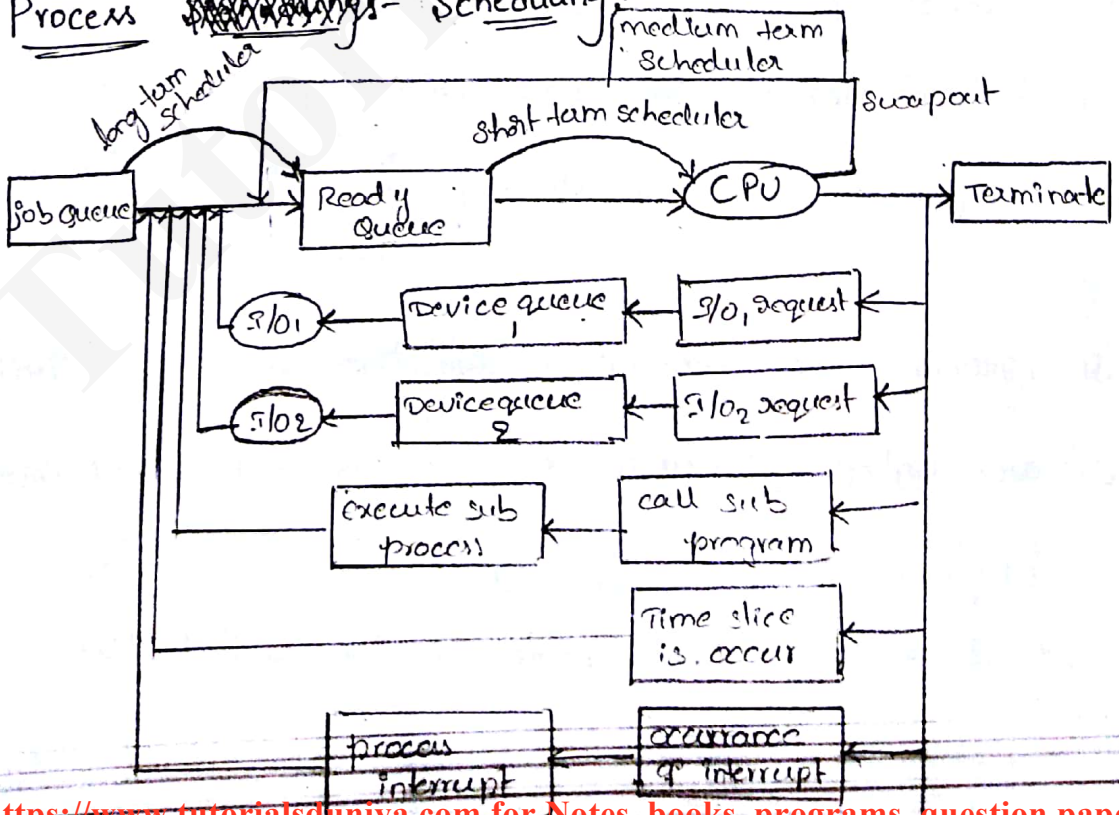
1. Process State:- It represents current state of the process is in execution. State may be New, ready, running, blocked or terminate.
2. Program Counter:- It represents number of currently executed instructions in the program.
3. Process Number:- It represents identification no. of process. At the time of process execution OS can decide unique number to a process for identification. It is called Process number.

4. CPU Registers:- It is used to store content of CPU registers at the time of stops the process execution. The CPU registers are accumulators, stack pointers, General purpose registers.

5. Memory Limits:- It is used to store start, end addresses of a memory area in main memory allocated to a process for its execution. This information is used by OS to avoids the process from accessing data out of its limitations.

6. I/O Used (or) Files Open:- In this OS can save what are the I/O devices used by the process and type of I/O devices. What are the files opened in process execution. It is used to avoid process execution without release the I/O devices and close the opened files.

Process ~~Scheduling~~ Scheduling:-



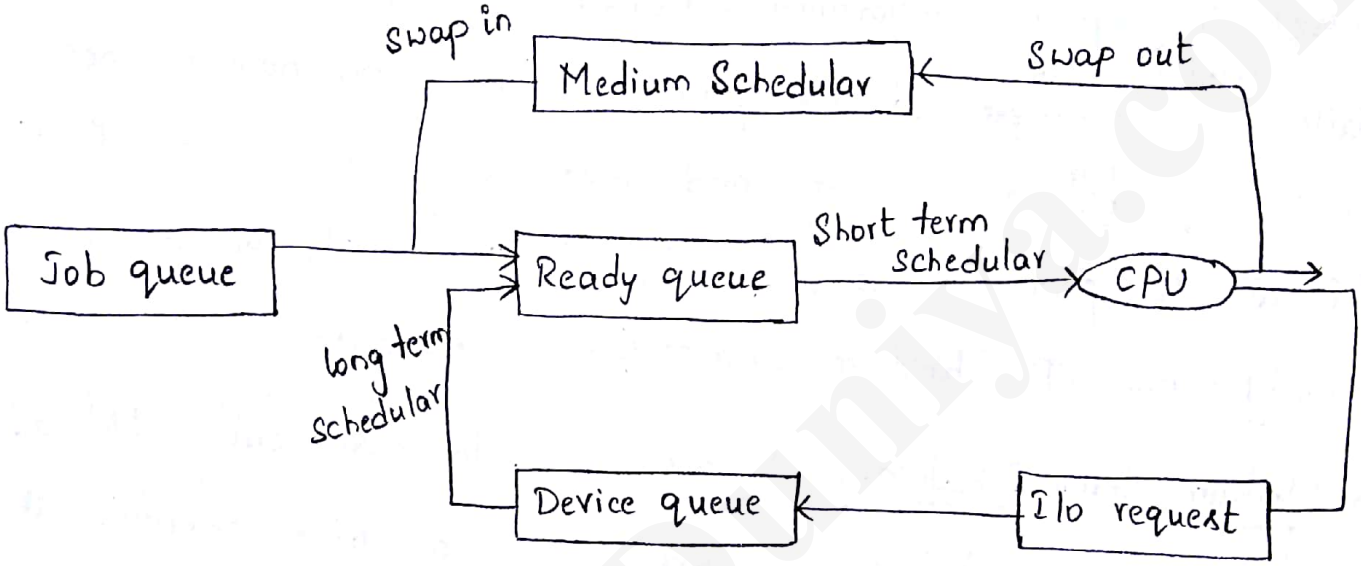
In multitasking (or) multiprogramming more than one program loaded into main memory to makes availability of CPU time efficiently. In these systems one process is scheduled (selected) for execution at a time.

Scheduling Queues :- In process execution, a process need to visit 3 types of scheduling queues.

1. Job Queue :- The process entered into the system can be placed in job queue.
2. Ready Queue :- The jobs which are loaded into main memory and ready for execution are placed in ready queue.
3. Device Queue :- Jobs which are wait for the availability of an I/o device are placed in device queue. In a system, each I/o device associates with seperate device queue for place the process waiting for the I/o device only.

In general the scheduling algorithm. queues are implemented as linked list. Each node in the linked list saves the PCB of one process. The queue header contains address of first and last node of the linked list.

Process Schedulers:- During the process execution, it must be scheduled among these 3 process scheduling queues. The scheduling can be performed by appropriate schedular. There are mainly 3 types of schedulers available.



1. Short-term Schedular:- It is used to schedule a process from ready queue for execution by CPU. It make scheduling a process everytime with less amount of time b/w two selection i.e., milliseconds to micro seconds.

2. Long-term Schedular:- It is used to schedule a process from job queue and load it in main memory i.e., in ready queue. In this the schedular makes selection after minutes of time from the first selection. Long term schedular select processes carefully to balance

the system. There are 2 types of processes.

CPU-Bound Processes:- Spent most of its time for the computation.

I/O-Bound Processes:- Spent most of its time for I/O Operation

The long term schedular intermix the 2 types of processes in equal proportionality. Otherwise system balanced is failed. for example, all the processes are CPU bound, then device queues are empty and increases the burden of CPU. If all the processes are I/O bound, then ready queue is empty. and CPU have no processes to execute.

3. Medium-term Schedular:- It is used to swap out a blocked process from main memory and after sometime reenters it into ready queue for execution. This process is called swaping.

Context Switch:- If one process execution is stopped in the middle, then OS can save context of such process in stack segment. and switch to another process execution. After sometime retrieve context of current process and continuous the remaining program execution. This process is called context switching.

Multithreading:-

Thread is a light weight process. Thread is used to execute module of a process. A thread have its own program counter, CPU registers, stack segment. All the threads belong to a single process shares code segment, data segment, memory (heap) of that process as shown below:

In multithreading multiple threads are executed parallely. Multithreading allows an application program to perform two (or) more activities simultaneously. Ex:- Word Processor. Read characters entered by the keyboard, check the spelling of the words, do editing etc....

Multithreading provides following benefits compared to single thread of execution.

1. Responsiveness:- In this if any one thread is blocked the remaining thread continues the execution and gives response to the user.

2. Resource Sharing:- In this multiple threads share resources of a process like code, data, memory and files opened in that process.

3. Economy:- In this there is no need to context switch because every thread have a separate program counter, SP & CPU registers. So, consider the concept of context switch.

4. Utilization Of Multiprocessor Architecture:- Multithreading provides a possibility to utilize the multiprocessor architecture efficiently. In this multiple threads are executed parallely on multiple processors. A thread is implemented at two levels.

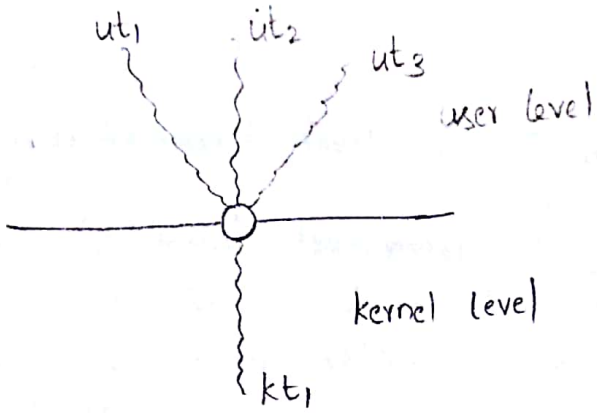
1. User level Threads
2. kernel level Threads

1. User Level Threads:- These are created by the user at user space.
2. kernel Level Threads:- These are created and managed by kernel in kernel space.

A user level thread must need the support of kernel level thread for its execution. So, there are 3 types of relations are available between user level threads and kernel level threads. They are:

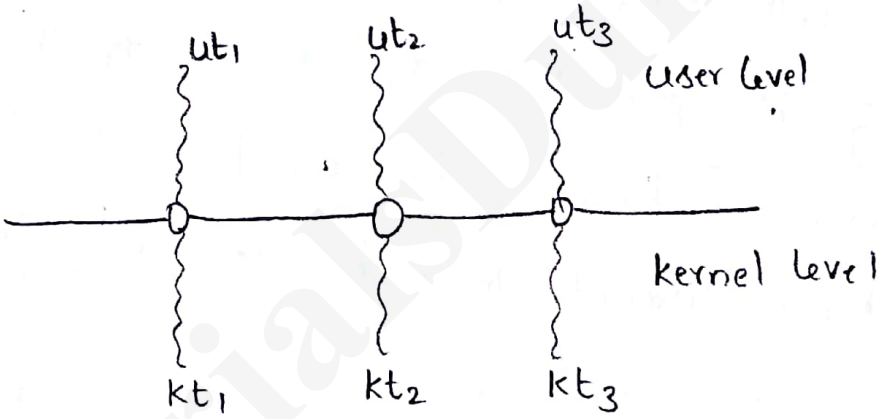
1. Many to One
2. One to One
3. Many to Many

Many to One:- In this two (or) more user level threads are mapped with one kernel level thread. In this one kernel level thread provide support to execute multiple user level threads.



In this the one kernel level thread blocked in its execution then all the user level threads are blocked. So, it provides less reliability.

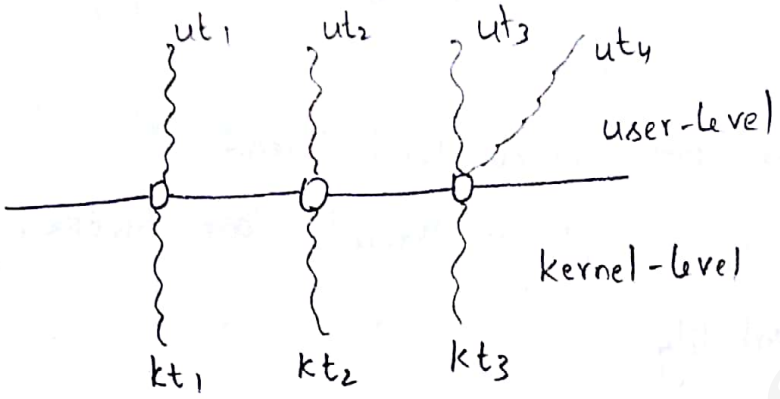
One-to-One:- In this one user level thread maps with one kernel level thread.



In this if a kernel level blocks then only corresponding user-level thread is blocked. All the remaining threads continues its execution. So, it provides high reliability. But in this the kernel needs to create a seperate thread to handle one user level thread. So, the burden to the kernel is increased.

Many-to-Many:-

In this many user level threads are mapped with many (less than no. of user level threads) kernel level threads.



In this if one kernel thread blocked another kernel thread will handles the user-level thread. So, in this reliability is provided. The burden to the kernel level is increased because the kernel need not to create a seperate kernel thread to each user level thread.

Thread Scheduling :-

User level threads are executed with the support of kernel level threads.

Three type of mappings are available between user level threads and kernel level threads.

In many-one and many-many mappings two or more user level threads mapped with single kernel thread. So a kernel thread need to schedule one user level thread at a time. In this competition is rised among all the threads belongs to a single process per a kernel thread. It is called "process contention scope" [PCS]. The PCS follows priority scheduling algorithm.

The kernel level thread need to get the control of CPU for its execution. So there occurs competition among all the kernel level threads for CPU. This is called "System contention Scope" [SCS]

In this the CPU follows multi-level priority scheduling.

Scheduling algorithm in Solaris :-

It follows multilevel queue scheduling based on primitive priority queue scheduling algorithm. But in this highest priority value have highest priority. In this the processes are categorised into following categories.

- ↳ Real time processes
- ↳ System processes
- ↳ Time shared processes
- ↳ Interactive processes

TutorialsDuniya.com

Get FREE Compiled Books, Notes, Programs, Books, Question Papers with Solution* etc of following subjects from https://www.tutorialsduniya.com.

- C and C++
- Programming in Java
- Data Structures
- Computer Networks
- Android Programming
- PHP Programming
- JavaScript
- Java Server Pages
- Python
- Microprocessor
- Artificial Intelligence
- Machine Learning
- Computer System Architecture
- Discrete Structures
- Operating Systems
- Algorithms
- DataBase Management Systems
- Software Engineering
- Theory of Computation
- Operational Research
- System Programming
- Data Mining
- Computer Graphics
- Data Science

- Compiled Books: https://www.tutorialsduniya.com/compiled-books
- Programs: https://www.tutorialsduniya.com/programs
- Question Papers: https://www.tutorialsduniya.com/question-papers
- Python Notes: https://www.tutorialsduniya.com/python
- Java Notes: https://www.tutorialsduniya.com/java
- JavaScript Notes: https://www.tutorialsduniya.com/javascript
- JSP Notes: https://www.tutorialsduniya.com/jsp
- Microprocessor Notes: https://www.tutorialsduniya.com/microprocessor
- OR Notes: https://www.tutorialsduniya.com/operational-research

Windows XP :-

It is also use thread based primitive priority scheduling algorithm. In this the processes are divided into two categories.

→ Variable processes

Its priority level is rangle from 1-15

→ Real time processes

Its priority level is range is from 16-31.

→ Memory management related processes have the priority '0'.

LINUX :-

It also uses multi-level queue scheduling algorithm. In this it shares the CPU time to a class of processes based on the process priority. It can give small CPU time to the high priority processes and large CPU time to the low priority processes. In this processes are divided into two categories.

↳ Real time processes

Its priority range is from 0-99.

↳ User processes

Its priority range is from 100-199.

In this user defined processes can be modified using 'nice' command.

Also called as 'Nice processes'.

Process Communication :-

Processes are divided into two categories. They are

↳ Independent process

↳ Cooperative process.

→ Independent process :-

It is a process that can't effect or be effected by any other process. It also doesnot share any data with any other process.

→ Cooperative process :-

It is a process that can effect or be effected by any other process. It also share data with any other process. The following are the benefits of cooperative process.

① Resource Sharing :- It allows to share data or files among multiple processes.

② Computational Speed-up :- In this the process is divided into small dependent processes and execute all the small processes componently. So that the speed of process execution is increased.

③ Modularity :- It provides features to a design process a separate set of modules.

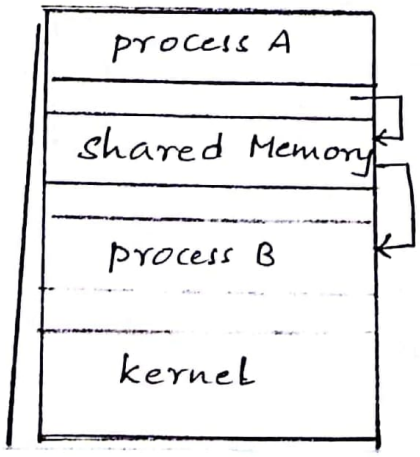
④ Convenience :- In this it is convenient to pass the data to other processes.

The communication between any two co-operative processes is called inter process communication.

We can use following two approaches to increment IPC [Inter process Communication].

① Shared Memory :-

In this a memory segment in main memory is created. This memory segment is shared by the two processes that want to communicate with each other.



In this there is no kernel intervention to implement IPC. The sender process write data into shared memory, want to send to the receiver.

The receiver process read data from the shared memory, want to read from sender. In this the memory segment is created by one process and attached by other process before performing read/write operations in the memory segment.

It have two limitations.

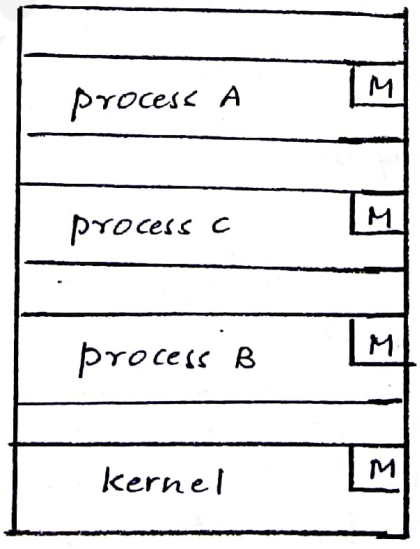
→ It is used to implement only communication between two processes of same host, it is complex to implement IPC between two different processes available on two different hosts.

→ By using it large amount of data can be passed at a time.

→ It implements communication at very high speed [equal to the memory accessing speed].

(2) Message Passing :-

In this IPC is implemented by passing a message from one process to another process.



In this each process is associated with message/mail box. A message box is an object that contains the messages to send/receive from other process.

In this the kernel intervention is compulsory to implement IPC. The sender process send message to the kernel, want to send to the receiver.

Kernel :- The kernel sends the message read from sender to the correct receiver. In this we can send small amount of data at a time because message size is limited.

In this the communication speed is low. It is also used to implement IPC between two processes of different systems. In this message passing technique we must need to know about type of logical link, between two processes, type of send and receive operations used, buffer capacity of mailbox.

Link Establishment :-

The logical link between two processes is established in two ways.

(1) Direct Communication :-

In this source and destination processes need to mention name of receiver or sender processes. In this the primitives of send and receive operations are

Send (M, A) - Send message M to process A only.

Receive (M, B) - receive message M from process-B only.

It is called as Symmetric Direct communication.

But if the receive operation doesnot mention the from address [sender process name] then it is called Asymmetric Indirect Communication.

In this the send and receive primitives are

Send (M, A)

Receive (M, id)

(2) Indirect Communication :-

In this the communication is implemented by using the concept of mailbox. A sender send messages to a

mailbox rather than to process B.

In this the send and receive primitives are as follows.

Send (M, A) - Send a message to the mailbox of process A

Receive (M, B) - Read a message from mailbox of process B.

Type of send and receive message / Synchronized and Non-Synchronized Operations :-

The send and receive operations are performed in two different modes.

(1) Blocked Send :-

In this the sender send a message to the receiver and wait until the message is received by the receiver.

(2) Non Blocked Send :-

In this a sender process send a message to the receiver and continues its execution without bother about the message is successfully received or not.

(3) Blocked Receive :-

In this the receiver process wait untill a message is received from the sender.

(4) Non-Blocked Receive :-

In this the receiver reseems its execution if there are no messages from the sender.

Types of mailbox capacity / Buffering capacity :-

There are three types of buffering capacities.

They are

- ↳ Zero capacity
- ↳ Bounded capacity
- ↳ Unbounded Capacity

(1) Zero capacity :-

In this the mailbox cant provide any storage to load the messages. So the sender need to wait untill message is received by the receiver.

(2) Bounded Capacity :-

In this mailbox provide storage to store limited or number of messages. So a sender process allow to send n number of messages continuously but mailbox is full the sender need to wait untill either space is available in the mailbox or one message is received by the receiver.

(3) Unbounded capacity :-

In this mailbox provides unlimited storage to store any number of messages. So the sender always need not to wait for receiving of the message by the receiver.

CPU scheduling algorithm:-

CPU scheduler:-

CPU scheduler select one process from ready queue and dispatch it for execution by the CPU. This CPU scheduler may need to select a process from ready queue in the following four situations.

* when the currently running process move to waiting state because of I/O unavailability or execute wait() system call.
* The currently running process move to ready state due to occurence of interrupt or time slice completion
* A waiting process moves to ready state due to availability of the required I/O
* The currently running process moved to terminate after completion of its execution

The CPU scheduling algorithm are basically divided into two categories based on the situation when a schedular selects a process.

Non-Premitive scheduling:- This type of scheduling is followed by the cpu schedular in occurance of situation-1 and 4.

Now, the scheduler need to select a new process from ready queue

In this, once CPU is allocated to a process, the process releases the CPU, either after completion of process execution or put the process in waiting state.

Premitive scheduling:-

In this, the scheduler follows situations 2 and 3 to select a process from the ready queue.

In this, once CPU is allocated to a process it release the CPU either time is completed or the required I/O is not available. It also release the CPU when high priority process is entered into the queue.

Scheduling Criteria:-

There are various CPU scheduling algorithms are available used to select one process from ready queue. we need to compare these scheduling algorithm. ~~For~~ to find the optimal CPU scheduling algorithm. ~~to~~ ~~for~~ For this comparison, we can use following criteria.

CPU utilization:-

In this, we consider ~~the~~ to keep the CPU as busy as possible. In general, it ranges from 0 to 100 But, in system point & view it ranges from 40 to 90 percent.

The implementation of the FCFS policy is easily manged with FIFO queue. when a process in enterd into the ready queue, its PCB is linked onto the tail of the queue.

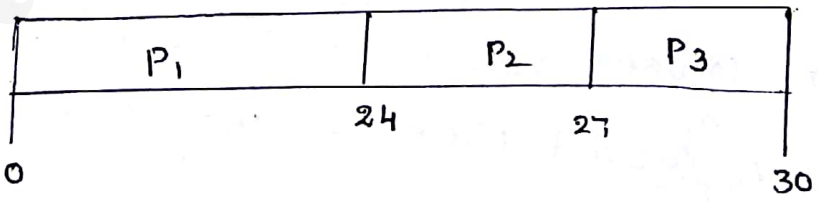
The code for FCFS scheduling is simple to write and understand.

The avg waiting time under FCFS policy, however, is often quite long. consider the following set of process that arrive at time 0, with the length of the CPU burst given milliseconds.

Process	Burst Time
P_1	24
P_2	3
P_3	3

Suppose that the processes arrive in the order: P_1, P_2, P_3

The Gantt chart for the schedule is



waiting time for $P_1 = 0$; $P_2 = 24$; $P_3 = 27$

Average waiting time: $(0+24+27)/3 = 17$

Suppose that the processes arrive in the order P_2, P_3, P_1

Through Put:- These the no. of processes executed per one unit of time. The unit of time may be one minute or one hour.

Respone time:-

The time interval when first response is displayed to the user regarding process execution

waiting time:-

It is the time period between arrival time of the process into ready queue to first response of the process.

Turn around time:-

It is time period between arrival time of the process into ready queue to the time interval when the process execution completed. It is also the seem of process waiting time and burst time.

The cpu scheduling algorithm which produce Produce highest values for cpu utilization and through put criteria, lowest values for responce time, waiting time and turn around time is the best one.

FIRST-COME, FIRST served scheduling

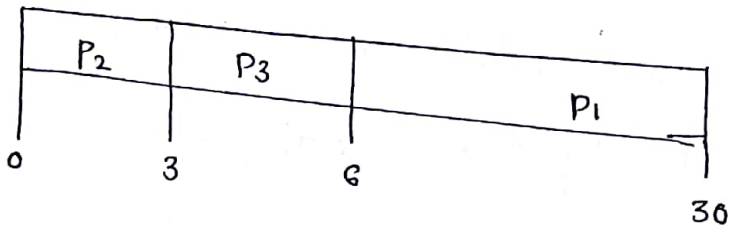
By far the simplest cpu-scheduling algorithm is the FCFS scheduling algorithm. with this scheme, the process that requests the cpu first is allocated the cpu first.

The Gantt chart for the schedule is: waiting time for $P_1 = 6$; $P_2 = 0$; $P_3 = 3$n

Average waiting time: $(6+0+3)/3 = 3$

much better than previous case

convoy effect short process behind long process



The FCFS scheduling algorithm is non preemtive. once the CPU has been allocatid to a process, that process keeps the CPU. until it releases the CPU either by terminating or by requesting I/O.

Shortest - Job - FIRST scheduling:-

A different approach to CPU scheduling is the SJF scheduling algorithm. This algorithm associates with each process the length of the process's next CPU burst. when CPU is available, it is assigned to the process that has the smallest next CPU burst. if the next CPU bursts of two processes are the same, FCFS scheduling is used to break the tie. Note that a more appropriate term for this scheduling method would be the shortest-next-cpu-burst algorithm, because scheduling depends on the length of the next cpu burst of a process, rather than its total length. we use the term SJF because most people and textbooks use this term to refer to this type of scheduling.

We define

$$\tau_{n+1} = \alpha t_n + (1-\alpha)\tau_n$$

The above formula defines an Exponential average. The value of t_n contains our most recent information; τ_n stores the past history. The parameter α controls the relative weight of recent and past history in our prediction. If $\alpha=0$, then

$$\tau_{n+1} = \tau_n.$$

and recent history has no effect.

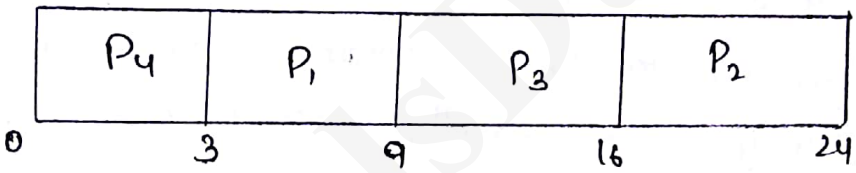
If $\alpha=1$, then $\tau_{n+1} = t_n$. and only the most recent CPU burst matters. more commonly, $\alpha = 1/2$. so recent history and past history are Equally weighted.

SJF algorithm can be either premitive or nonpreemitive

As an Example of SJF Scheduling, consider the following Set of processes, with the length of the cpu burst given in milliseconds.

Process	Burst time
P_1	6
P_2	8
P_3	7
P_4	3

Using SJF Scheduling, we would schedule these processes according to the following Gantt Chart:

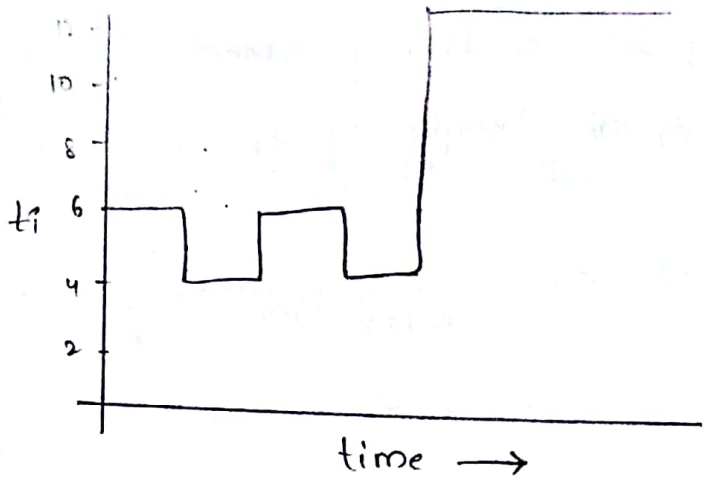


The waiting time is 3 milliseconds for process P_1, 16 milliseconds for process P_2, 9 milliseconds for process P_3, and 0 milliseconds for process P_4.

Thus, the average waiting time is

$$(3+16+9+0)/4 = 7 \text{ milliseconds}$$

The SJF scheduling algorithm is provably optimal, in that it gives the minimum average waiting time for a given set of processes.



cpu burst (ti)	6	4	6	4	13	13	13	...
"guess" (τ_i)	10	8	6	6	5	9	11	12 ...

prediction of the length of the next cpu burst.

Priority scheduling:

The SJF algorithm is a special case of the general priority scheduling algorithm. a priority is associated with each process, and the cpu is allocated to the process with the highest priority. Equal-priority processes are scheduled in FCFS order. an SJF algorithm is simply a priority algorithm where the priority (p) is the inverse of the next cpu burst. The larger the cpu burst, the lower the priority.

Here we discuss the scheduling in terms of high priority & low priority. priorities are generally indicated by some fixed range of numbers, such as 0 to 7 or 0 to 4,095.

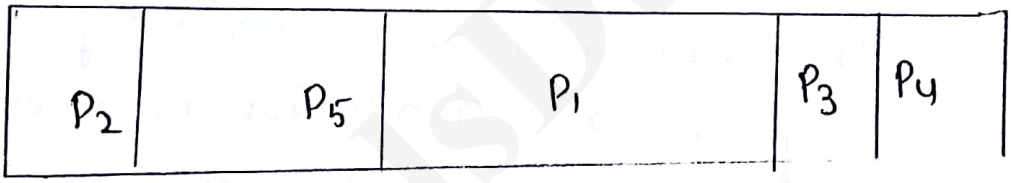
Example:

consider the following set of processes, assumed to have arrived at time 0, in the order $P_1, P_2 \ldots P_5$

with the length of the cpu burst given in milliseconds.

Process	Burst time	Priority
P_1	10	3
P_2	1	1
P_3	2	4
P_4	1	5
P_5	5	2

using priority scheduling, we would schedule these processes according to the following Gantt chart.



The average waiting time is 8.2 milliseconds

The priorities can be defined either internally or externally. Internally defined priorities use some measurable quantity or quantities to compute the priority of a process.

for example, time limits, memory requirements, the number of open files, the ratio of average I/O burst to average cpu burst have been used in computing priorities.

Round-Robin Scheduling:

Round-Robin scheduling algorithm is designed especially for time sharing systems. It is similar to FCFS scheduling, but preemption is added to switch between processes. A small unit of time, called a time, called a time quantum or time slice, is defined. A time quantum is generally from 10 to 100 milliseconds. The ready queue is treated as a circular queue. The CPU scheduling goes around the ready queue., allocating the CPU to each process for a time interval of up to 1 time quantum.

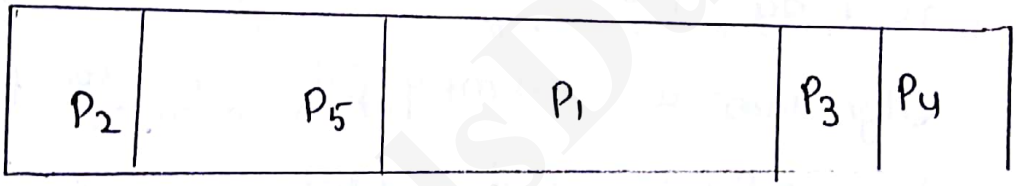
To implement RR scheduling, we keep the ready queue as a FIFO queue of processes. New processes are added to the tail of the ready queue. The CPU scheduler picks the first process from the ready queue., sets a timer to interrupt after 1 time quantum, and dispatches the process.

One of two things will then happen. The process may have a CPU burst of less than 1 time quantum. In this case, the process itself will release the CPU voluntarily. The scheduler will then proceed to the next process in the ready queue. Otherwise, if the CPU burst of the

with the length of the cpu burst given in milliseconds.

Process	Burst time	Priority
P_1	10	3
P_2	1	1
P_3	2	4
P_4	1	5
P_5	5	2

using priority scheduling, we would schedule these processes according to the following Gantt chart.



The average waiting time is 8.2 milliseconds

The priorities can be defined either internally or externally. Internally defined priorities use some measurable quantity or quantities to compute the priority of a process.

For example, time limits, memory requirements, the number of open files, the ratio of average I/O burst to average cpu burst have been used in computing priorities.

External properties are set by criteria outside the operating system, such as the importance of process, the type and amount of funds being paid for computer use, the department sponsoring the work, and other, often political factors.

Priority scheduling can be either premitive or non-premitive. when a process arrives at the ready queue. its priority is compared with priority of the currently running process. a primitive priority scheduling algorithm will preempt the cpu if priority of the ready arrived process is higher than the priority of the currently running process. A non premitive priority Scheduling algorithm will simply put the new process at the head of the ready queue.

a major problem with priority Scheduling algorithm is indefinite blocking or starvation. a process that is ready to run but waiting for the cpu can be considered blocked. therefore a solution to the problem of indefinite blockage of low-priority process is aging. Aging is a technique of gradually increasing the priority of processes that wait in the system for a long time.

TutorialsDuniya.com

Get FREE Compiled Books, Notes, Programs, Books, Question Papers with Solution* etc of following subjects from https://www.tutorialsduniya.com.

- C and C++
- Programming in Java
- Data Structures
- Computer Networks
- Android Programming
- PHP Programming
- JavaScript
- Java Server Pages
- Python
- Microprocessor
- Artificial Intelligence
- Machine Learning
- Computer System Architecture
- Discrete Structures
- Operating Systems
- Algorithms
- DataBase Management Systems
- Software Engineering
- Theory of Computation
- Operational Research
- System Programming
- Data Mining
- Computer Graphics
- Data Science

- Compiled Books: https://www.tutorialsduniya.com/compiled-books
- Programs: https://www.tutorialsduniya.com/programs
- Question Papers: https://www.tutorialsduniya.com/question-papers
- Python Notes: https://www.tutorialsduniya.com/python
- Java Notes: https://www.tutorialsduniya.com/java
- JavaScript Notes: https://www.tutorialsduniya.com/javascript
- JSP Notes: https://www.tutorialsduniya.com/jsp
- Microprocessor Notes: https://www.tutorialsduniya.com/microprocessor
- OR Notes: https://www.tutorialsduniya.com/operational-research

3. Synchronization

UNIT-III

Synchronization:- In Cooperative process there may be a situation for the occurance of data inconsistency when two processes executed parallely and perform modification operations on sharable resource.

Ex:- In production Consumer problem both producer and Consumer executed simultaneously. They need to modify the value of global variable "Counter" shared by both producer and Consumer procedures. In this the Counter value is either 4 or 6 but not 5.

To avoid data inConsistency use the Concept of Synchronization on co-operative processes. Synchronization is a mechanism to ensure that orderly execution of Cooperative processes while perform modifications on sharable resources. By using Synchronization it Provides data Consistency.

Consumer

```
While (true)
{
 while (counter == 0)
  ;
next Consumed = buf [out];
out = (out+1) % BUFSIZE;
Counter--;
}
```

Producer

```
While (true)
{
 While (Counter == BUFSIZE)
  ;
buf [in] = next produced;
in = (in+1) % BUFSIZE
Counter++;
}
```

```
reg 1 = Counter ;
reg 1 = reg 1 - 1 ;
Counter = reg 1 ;
```

} Internal operation {

```
reg 2 = Counter ;
reg 2 = reg 2 + 1 ;
Counter = reg 2 .
```

Critical Section Problem:- A System Contains 'n' no. of processes $P_1, P_2, \ldots P_n$ share a Common file or try to modify a Common variable. Then the System need to allow only one process to modify file Content and Variable Content, does not allow any other process until the Completion of first Process Execution. In this the Code in a process used to implement modification operation on a file or Variable is Called Critical Section. So the System design a protocol used to prevent any other process execute its Critical Section until Current process Completes the Critical Section execution. The protocol provides 2 features.

1. The process make request for the permission to enter into its Critical Section. The Code to implement this request is Called Entry Section.
2. The process need to release the resources when exit from the Critical Section. The Code to implement the exit operation is Called exit Section. The remaining Code in process is Called Remainder Section.

Peterson's Solution For Critical Section problem:-

Peterson provides a s/w solution to Solve the Critical Section problem Implement Synchronization but the peterson's problem is used to Synchronize b/w any 2 processes for example P_i, P_j are 2 Processes. In peterson problem we can use 2 data-structures.

int turn; → used to Specify which process turn is Currently enters into the Critical Section.

boolean flag [2];

→ flag is used to Specify the readyness (or) the interest of a process to enter into its Critical Section.

flag [0];

→ Specifies interest of P_i

flag [1];

→ Specifies interest of P_j.

P_i :-

```
do
{
    turn = i;
    flag[j] = true;
    while ((flag[j] == True) && (Turn == j))
    ;
Critical Section
    flag[j] = false;
} while (1);
```

→ The structure of a process Contains Critical Section is shown below:

```
do
{
    Entry Section
    Critical Section
    Exit Section
    Remainder Section;
} while
```

A Solution designed for Critical Section problem must Satisfies given 3 requirements.

1. Mutual Exclusion:- It allows only one process into execution.
2. Progress:- If anyone process not enter into Critical Section, then Select one process from the list of waiting Processes allows it into Critical Section.
3. Bounded wait:- There is a limited no. of times a process enter into its Critical Section when another process make a request for enter into Critical Section. A process Completes its execution with in a limited waiting time for the arrival of the request.

→ The Critical Section problem also occurs in os. programs. The Kernel can avoid Critical Section problem by executes the Kernel in non-primitive mode, while executes Critical Section i.e; the Kernel Cant be Pre emitted its Control until process is Completed.

Swap :-

It is also a hardware instruction used to implement mutual exclusion b/w two (or) more processes. It is also a atomic instruction. i.e; it is executed without any interrupts. It is used to interchanges the contents of given two parameters as shown below:

```
Swap (boolean * lock, boolean *key)
{
   boolean  * temp = * lock;
            * lock = * key;
            * key = * temp;
}
boolean lock = false;
```

Pi

```
do
{
  boolean key = true;
  while (key == True)
  Swap (* lock, * key)
  // Critical Section;
     lock = false;
  // remainder Section;
} while (True).
```

Pj

```
do
{
  boolean key = True;
   while (key == True)
  Swap (* lock, * key);
  // Critical Section;
     lock = false;
  // remainder Section;
} while (True).
```

In this we can use two variables. The global variable lock is shared by all the processes. It is initialized to "false". Key is a local variable to each process. It is initialized to 'True'. A process check key value and swaps the content of lock and key variables. If key is always true the process doesn't enter into Critical Section, if key value is false then process enter into Critical Section.

It is used to stores the previous value target and load a new value into the target. It returns old value (rv) to the calling function. It is used to implement Synchronization among multiple processes as shown below:

```
boolean lock = false;

Pi                                  Pj
do                                  do
{                                   {
while (Test and set (&lock))        while (Test and set (&lock))
  ; do nothing                        ; do nothing
// Critical Section;                // Critical Section;
   lock = false;                       lock = false;
} while (true)                      } while (true)
```

All processes share a global variable lock and is initialized with "False".

When one process Test and set address of is passed, it returns "FALSE" and set lock as "True" then it enters into the Critical Section and prevent all the remaining processes by set the lock value as "True". Once the process Completes Critical Section execution it sets the lock to a "False" value i.e., removes the lock. Any another process try to enters into Critical Section will enters into Critical Section and set the lock by set lock with "True".

Pj:-

```
do
{
    turn=j;
    flag[i]= true;
    While ((flag[i]==True) && (Turn==i))
    ;
    Critical Section.
    flag[i]= false;
}while(1);
```

Busy waiting:-

In this a process put in busy waiting state when try to enters into the Critical Section. In this State the process will test Continously the value of Semaphore. Due to this CPU time is wasted. To avoid this busy waiting we can modifies the Semaphore definition. Now a waiting list is added to each Semaphore. The waiting list is used to store all the Processes that are wait for the resource Controlled by Semaphore. Now we can use following definition of wait and Signal instructions.

```
Struct Semaphore
{
    Int val;
    Struct process *list;
}s;
Wait (Semaphore *s)
{
    s->val--;
    if (s->val<0)
    {
        add process to s->list;
```

```
    block();
    }
  }
Signal (Semaphore *S)
  {
    S->val++;
    if(S->val<=0)
    {
      Remove process 'p' from S->list;
      wakeup();
    }
  }.
```

In entry Section a process executes wait() to acquire a resource. In wait() the Semaphore value is decremented to 1 & check the availability of resources using S->val is less than 0. If it returns false resource is available & allocated to a process then the process enters into Critical Section. If the Condition returns true i.e., the Semaphore value is negative resource is not available. So the process add to the waiting list, associated with Semaphore. The process moves from ready State to blocked State using block() function. Now the Process execution is Suspended and gives the cpu control to another process. In this the absolute value of Semaphore represents total no. of processes available in the waiting queue.

Signal:- In exit Section a process executes Signal operation. The Signal returns a resource and check the waiting queue by using S->V<=0. If it returns false there is no waiting processes

Deadlock:-

In implementation of Synchronization using Semaphore there is a situation for the occurance of deadlock. Deadlock is a state when a waiting process wait for an event performed by another process in the waiting list.

P_i	P_j
wait (p);	wait (q);
wait (q);	wait (p);
⋮	⋮
Signal (p);	Signal (q);
Signal (q);	Signal (p);

Hardware Approach For Synchronization:-

Hardware approach makes easier to develop program and improves the system efficiency. There are two types of H/w instructions are provided available with all H/w Configurations.

1. Test and Set
2. Swap.

1. **Test and Set:-** It is a atomic instruction. It is executed without any interrupts. It is used to test value of its argument and set the argument with a new value as shown below:

```
Test and Set (Boolean * target)
{
    boolean rv = *target; (previous value of target
                           assigned to rv)
    *target = True;
    return rv;
}
```

on the queue & resource is given to any other proces. If it returns true there is waiting process in the queue. Now one processes is removed from queue & starts its execution by allocates the resource. Here we use wakeup() instruction to resume a waiting process execution. It moves the process from blocked to ready state. In this way using block & wake up instructions along with Semaphores we can avoid busy waiting of a process on a Semaphore.

There are 2 types of Semaphores available

1. Binary Semaphore :- It is used to Synchronize the accessing of single instance of a resource. Its value is either 0 (or) 1. If Semaphore value is 1 it means resource is available. If Semaphore value is 0 it means resource is not available.

2. Count Semaphore :- It is used to Synchronize the accessbility of multiple instances of one resource. Its value is initialized to total no. of instances of a resource. In this we can able to execute 'n' no. of processes enters into Critical Section at a time. Once S value becomes zero (0) all the remaining processes are put in waiting list & Semaphore value is decremented by 1.

and set the lock as "True" to prevent all the remaining processes enters into Critical Section. Once the process Completes Critical section execution and Set lock as "False" to allows any other process tries to enter into Critical Section.

Both test and set, swap instructions are used to only implement mutual exclusion. But they can't implemented bounded waiting. So, there are not implement Synchronization.

Software Approach:-

Semaphore:- It is a S/w to implement Synchronization. It reduces the Complexity in implementation of Synchronization. It is a integer variable. It is represented by 's'. It is used to Synchronizes the accessibility of a single resource. The Semaphore value can be modified by executing two types of operations on a Semaphore

(i) **Wait:-** It is represented by 'P' (proberen). proberen means to wait. It is used to test the Semaphore value. If value is less than (or) equal to '0' then put the process in waiting state otherwise decrements the Semaphore value.

(ii) **Signal:-** It is represented by V (verhogen). Verhogen means to increment. It is used to increment the Semaphore value by 1.

Both wait and Signal operations are atomic instructions. So, no two processes are allowed at a time to perform either wait (or) Signal operations on a Semaphore value.

Wait

```
wait (int s)
{
 while (s<=0)
  ; do nothing
   s--;
}
```

Signal

```
Signal (int s)
 {
   s++;
 }
```

Now we take a Semaphore called "mutex" and initialized to 1.

Implementation of Semaphore:-

Pi

```
do
{
 wait (mutex);
// Critical Section;
  Signal (mutex);
} while (true).
```

Pj

```
do
{
 wait (mutex);
// Critical Section;
  Signal (mutex);
} while (true).
```

In this when the 1st process executes wait operation with mutex value as 1 then the wait instruction decrements the semaphore value by i.e; mutex is 0. then allows the process into Critical section.

Now all the remaining processes that executes wait operation with mutex value '0' are put in busy waiting i.e:, Continously test the Semaphore value until greater than '0'.

Once process Pi Completes Critical section and execute Signal operation on mutex then the mutex value is incremented by 1. Now any

one of the process in waiting state are allowed to enter into Critical Section.

In Semaphore the drawback is busy waiting.

Bounded Butter problem:-

In this a buffer pool used to store 'n' no. of items. one buffer stores one item. Both producer and Consumer Can't enters into the butter pool at a time. only one process either producer or Consumer enters into butter pool. The producer produces an item & added it to butter pool only when the empty butter is available in butter pool. Consumer access the Butter pool to read an item only when an item is available in the butter pool. So we need to synchronize the usage of butter pool b/w producer & Consumer.

Semaphores are used to solve the bounded butter problem by implementing Synchronization b/w Consumer & producer as shown below:

In this we Can use a Semaphore mutex to Synchronization the accessibility of butter pool b/w producer & Consumer. It is initialized to Value 1.

Semaphores full used to represent total no. of full buffers in the pool. It is initialized to Zero.

The Semaphore empty represents no. of empty buffers in the pool. It is initialized to n.

Producer

```
do
{
// produce an item;
  wait (empty);
  wait (mutex);
// add item to buffer in pool;
  Signal (mutex);
  Signal (full);
} while (true).
```

Consumer

```
do
{
  wait (full);
  wait (mutex);
// Remove item from
     buffer pool;
  Signal (mutex);
  Signal (empty);
// Consume the item;
} while (true).
```

Producer:- First the producer checks empty Space is available or not by executing wait (empty). If empty space is available it again checks Consumer is already uses the Buffer pool or not by executing wait (mutex). If the Consumer is not using the Buffer pool. Now producer releases the lock applied on Buffer pool by executing Signal (mutex) & remove one item from the waiting Consumers list & intimates to read item from the Buffer pool.

Consumer:- Consumer first verify is there any item is available in the buffer pool by executing wait (full). Once item is available in the buffer pool then it again checks if buffer pool is used by producer or not by executing wait (mutex). If the buffer pool is free Consumer remove one item from the Buffer pool

After this unlocks the butter pool using signal (mutex) & intimates to the producer for add or item to the butter pool using signal (empty). Now the Consumer Consumes the item.

In this way Synchronisation is Implemented b/w producer & Consumer using Semaphores.

UNIT - 4

Memory Management

Loading :-

Data is transfer from secondary memory to main memory is called "Loading".

Store :-

Data is transfer from main memory to secondary memory is called "storing".

Swaping :- A process in main memory is swaped out into secondary memory & the same process is swaped into main memory after some time.

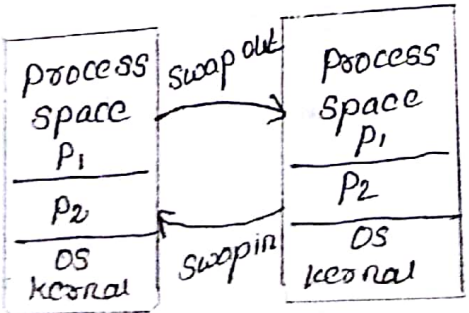
Context Switching :- If all the processes are blocked then the new process is bought into Execution from secondary memory & saves context values.

Main memory is divided into 2 blocks

(1) one is used by OS

(2) Another is used for all the processes

swaping ⟶ swapout + swapin



Swapped out means when the processes are not ready for execution.

→ When one blocked process is swaped out then swapin occurs.

→ The address on main memory is called physical address or original address.

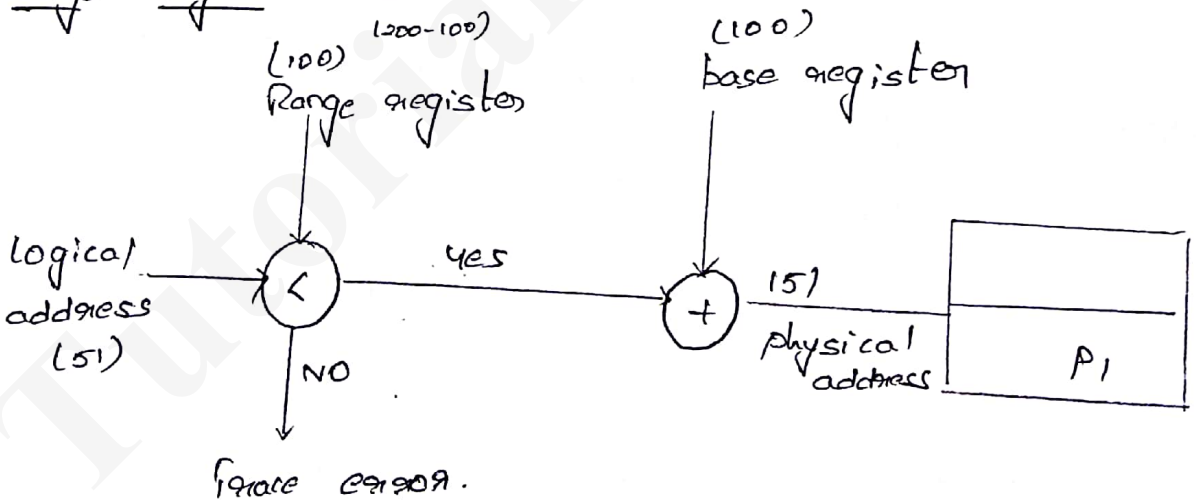
→ The address generated by cpu i.e pc is called as logical address.

→ Binding the logical address to a suitable physical address is called Address binding/mapping. It also provides protection to other processes.

Address mapping is done by using two registers.

1) Base register:- It is used to store lowest address or starting address in main memory.

2) Range register:- It is used to store size of the process.



→ If logical address is compared to range register, if it is less than, then it provides protection.

Conditions for swaping :-

i) A process can be swaped when it is idle i.e process can't be communicated with I/O devices.

ii) We can use Round Robin and priority algorithm.

-> In Round robin, whose process time quantum is completed it is swaped out.

-> In priority, whose process has lowest priority that is swaped out.

Dynamic loading :-

Loading is that the data is transferred from secondary memory to main memory.

storing is nothing but the data transferred from main memory to secondary memory.

The loading that is done at the runtime is called Dynamic loading.

-> If process size is greater than main memory size then the process is divided into 2 modules, that is done by the user and load main sub routines into main memory first and starts process execution. In the process execution if main subroutine calls secondary subroutine then loads the subroutine.

TutorialsDuniya.com

Get FREE Compiled Books, Notes, Programs, Books, Question Papers with Solution* etc of following subjects from https://www.tutorialsduniya.com.

- C and C++
- Programming in Java
- Data Structures
- Computer Networks
- Android Programming
- PHP Programming
- JavaScript
- Java Server Pages
- Python
- Microprocessor
- Artificial Intelligence
- Machine Learning
- Computer System Architecture
- Discrete Structures
- Operating Systems
- Algorithms
- DataBase Management Systems
- Software Engineering
- Theory of Computation
- Operational Research
- System Programming
- Data Mining
- Computer Graphics
- Data Science

- Compiled Books: https://www.tutorialsduniya.com/compiled-books
- Programs: https://www.tutorialsduniya.com/programs
- Question Papers: https://www.tutorialsduniya.com/question-papers
- Python Notes: https://www.tutorialsduniya.com/python
- Java Notes: https://www.tutorialsduniya.com/java
- JavaScript Notes: https://www.tutorialsduniya.com/javascript
- JSP Notes: https://www.tutorialsduniya.com/jsp
- Microprocessor Notes: https://www.tutorialsduniya.com/microprocessor
- OR Notes: https://www.tutorialsduniya.com/operational-research

Memory allocation strategies / memory management techniques

→ we discuss how to allocate memory.
→ Continious allocation → entire process loaded into continous memory area
→ continious allocation strategies are divided into 2 types
1) MFT – memory with fixed no. of tasks
2) MVT

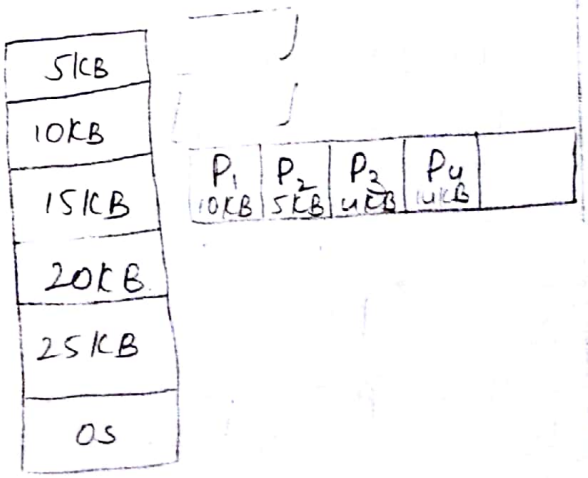
- MFT
  - MFT with equal size
  - MFT with unequal size
- MVT

user space is divided into fixed no of partitions by device manufacturer.
Each partition size is equal
→ A process is entered into a partition whose size is large enough to load a process.
→ The partition should be free.
→ The wasting of memory / unused memory is known as fragmentation.
  - Internal fragmentation
  - External fragmentation

→ If the fragmentation occurs with in the partition then it is called Internal fragmentation.
→ If the fragmentation occurs outside the partition it is known as external fragmentation.

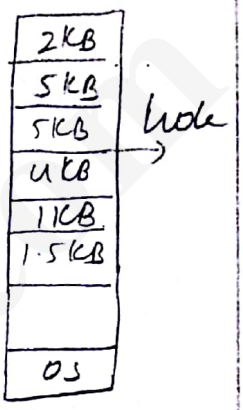
Int → 1 ext → 0

Drawback:-
1) Internal fragmentation
2) we can't able to load largest processes even entire memory is free. To overcome it uses MFT with unequal size
3) Each partition size is varied.
4) In this technique we don't follow single ready queue.
→ Here internal fragmentation is more and we take each block for each partition, If reduces to some extent & we can execute large processing.



CPU time is wasted
decreases system efficiency
we can't eliminate the fragmentation in MFT.

MVT:-
→ Manufactures doesn't divide memory into partitions.
→ At the time of process loading the memory is divided into partitions.
Hole:- The memory area already used by a process
Here we have external partition i.e., external fragmentation.
→ Before execution that part is divided into 2 subpartitions.
→ Here internal fragmentation is eliminated.



Drawback:-
1) External fragmentation.
→ we use technique called compaction → used to collect all the unused memory partitions into a large partition once a large partition is available it is used to load another process but compaction is burden to the CPU because all the used partitions are reallocated to upperside & holes are reallocated to loverside of the memory.
→ To reduce the external fragmentation we follow placement algorithum while a process load. there are 3 types of placement algorithums are available.
1) First fit:- In this process load into a partition i.e., first available in the search. Here the search starts from starting of the memory unit.
→ More external fragmentation is possible.
→ It takes less searching time & search until one partition is available.
2) Best-fit:- It can search for smallest partition i.e., big enough to load the process.
→ Here external fragmentation is less.
→ It takes very large time for searching.

Next-fit / worst-fit:-
It can search for largest partition to load the process by assuming that the remaining memory is suitable to load another small process completely.

Paging:-

To fully avoid the external fragmentation completely we use the technique called paging. The paging is non-contiguous memory allocation technique. In this entire process cannot be loaded in contiguous memory locations instead of the process loaded at various locations in the memory.

In this main memory is divided into fixed no. of blocks with equal sizes. Each block is called as frame. It has the physical address.

In this the process is divided into fixed no. of blocks with equal sizes known as pages. This points to logical address.

page size = frame size.

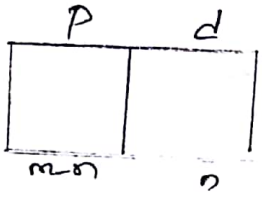
The process is loaded into main memory at the time of execution in non-contiguous frames. Here the external fragmentation is '0'.

At the time of loading, the operating system can save the information about process loading in a table known as Page Table. The page table contains 2 columns.

1) page no. 2) frame no.

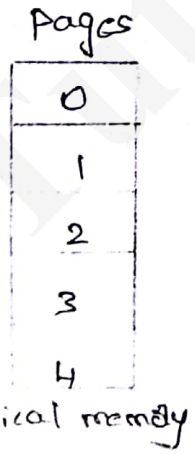
1) Page number :- It represents number of pages. It is used as an index into a page table.

2) frame number :- It represents which frame contains the corresponding page number.

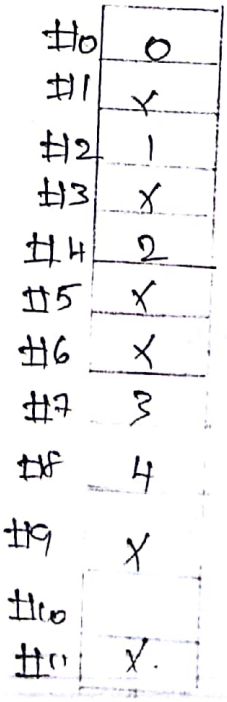


P → page number
d → displacement number.

→ Before loading, the operating system checks the availability of frame number.



Page No	Frame No
0	0
1	2
2	4
3	7
4	8



→ page table is used to access a page at process execution.

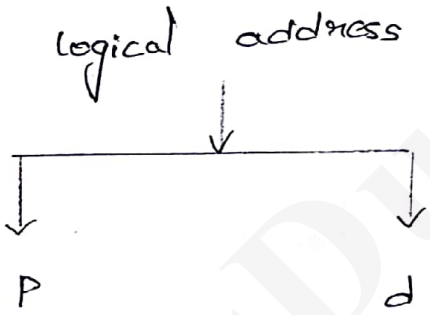
→ Here internal fragmentation may occur in the last page of loading.

→ Fragmentation range from 1 Byte to entire page or frame.

Hardware support :-

→ It is used in the address binding.

→ The logical address has 2 fields.

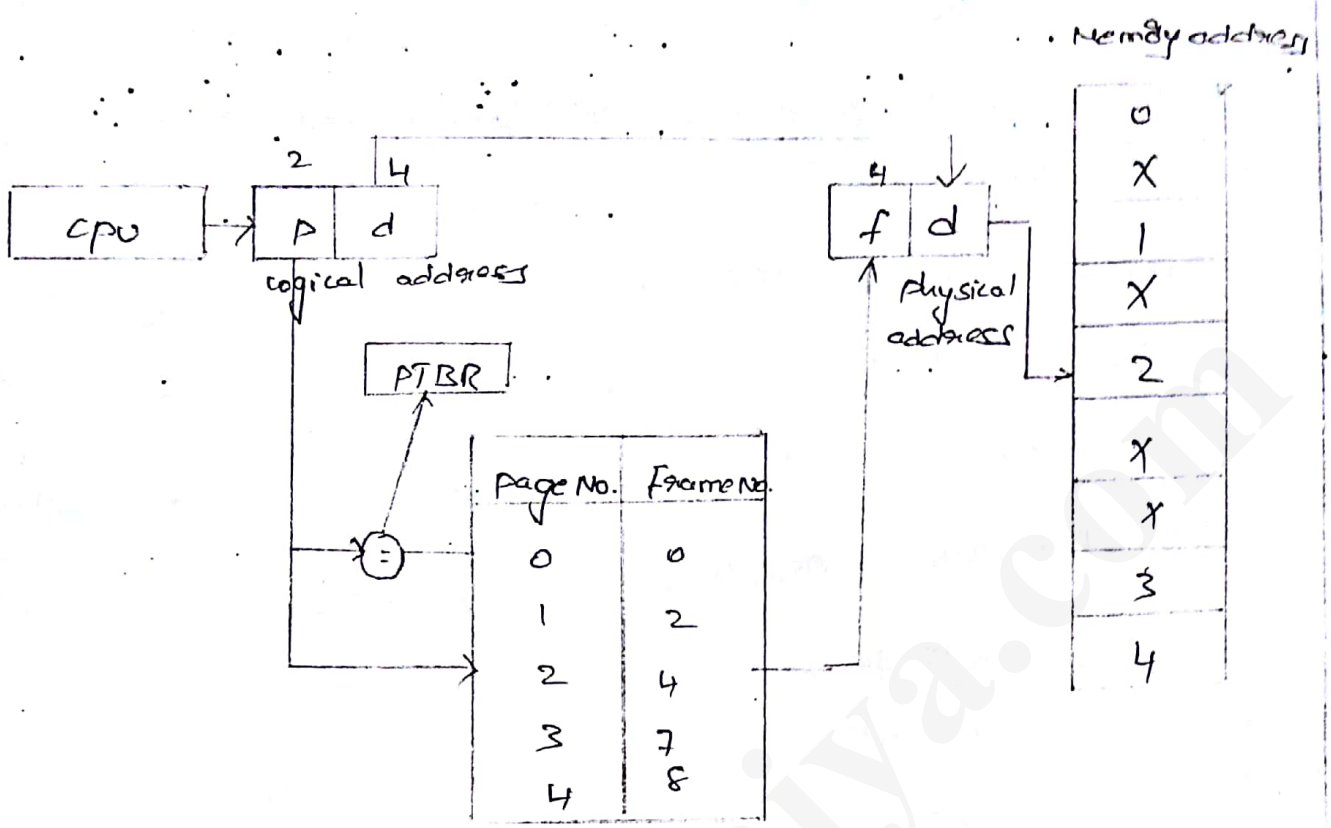


where p = page number

d = page offset / page displacement value.

→ Page number is used to add a page in the process.

→ page offset value is used to access particular instruction. and it is also used to address byte in that page.



To convert the logical address to physical address we use the page table. We take page number with key value and perform the search operation. It returns a frame number that contains a frame # and saved.

The displacement value is usually combined with the frame number

Page size:- In general the page size is varied from 512 bytes to 16 mega bytes. It varies depending on the system architecture.

In general the page size is 4 mega bytes

If page size is more then internal fragmentation is also more. If page size is less, internal fragmentation is also very less and increases the page table size.

→ It is very difficult to combine the page table.

Eg:- let us consider,

$m = 30$

$p = 2^n$

$d = 2^{m-n}$

$2^m = 2^n \times 2^{m-n}$

where

$2^n = p$

$2^{m-n} = d$.

→ cpu can maintain the page Table Base Register i.e PTBR, to store the location address. After this we add the page number.

PTBR value + pageno x 4 (32-bits)

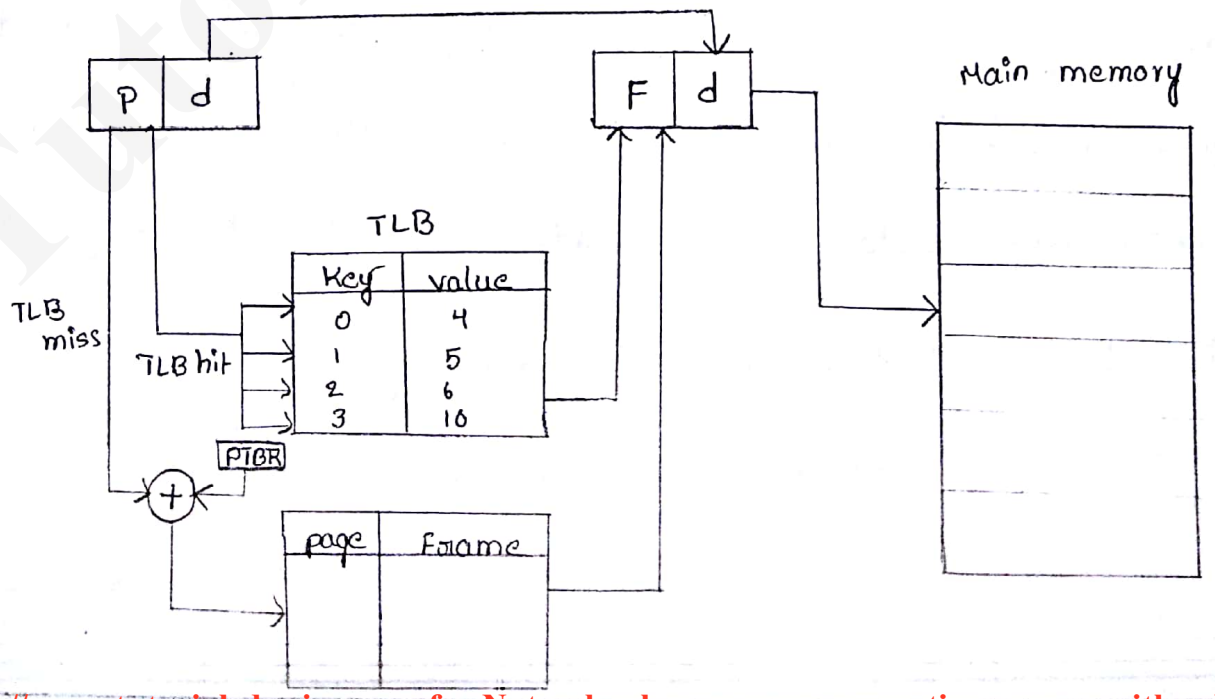
Here two times, the main memory is accessed. one for accessing the page table to find the frame number & second for accessing the original page.

TLB (Translation look a side buffer) :-

It is high speed, low size cache memory associated with paging hardware. The special feature of TLB is it check out all the entries at a time. TLB is also a table. It contains key and value in each entry. key is used to perform search. one key is matched then its corresponding value is returned. Here key is nothing but page number, value is nothing but frame number. It performs search operation on all entries at a time.

If page number is found in TLB then it is called TLB hit. Then TLB returns corresponding frame number.

If page number is not available in TLB then it is called TLB miss. It will search in page table and access the frame number.



TLB hit ratio :- The maximum number of times the page is available in the TLB is called as TLB hit ratio.

Effective Access time EAT = TLB hit + TLB miss

$= 0.8 \times 120 + 0.2 \times 220$

$= 96 + 44$

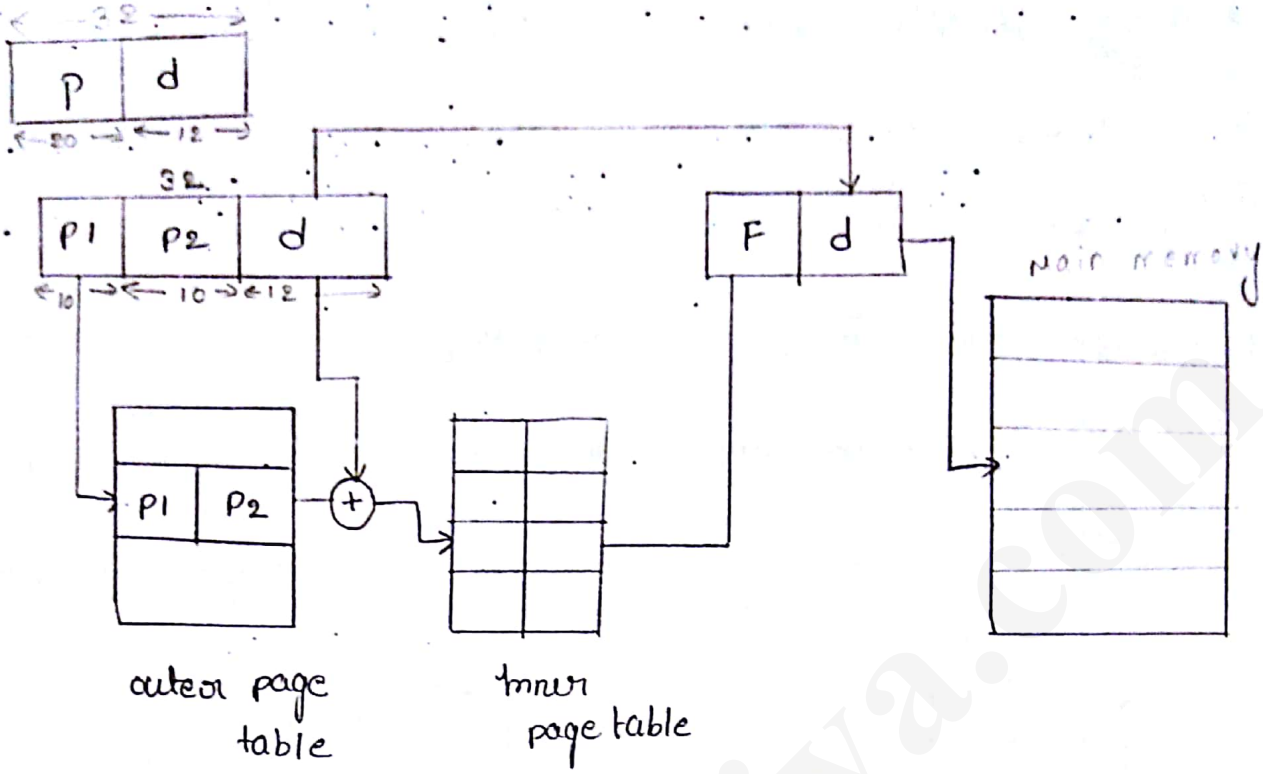
$= 140$

It is complex to maintain page table when page table size is large.

page table structures :-

We can maintain large page table in 3 page table structure. These structures are used to handle page table when it's size is very large i.e the address length is 32 bit or 64 bit.

1. Hierarchical page table :- In this the page table can also use paging technique to load into the main memory. In this another page table is defined at the time of outer page table loading. This is called inner page table. Here address length of page number is divided into 2 sub parts.



2. Hashed page table structure :-

In this page table can be defined by using hash value as page number in page table.

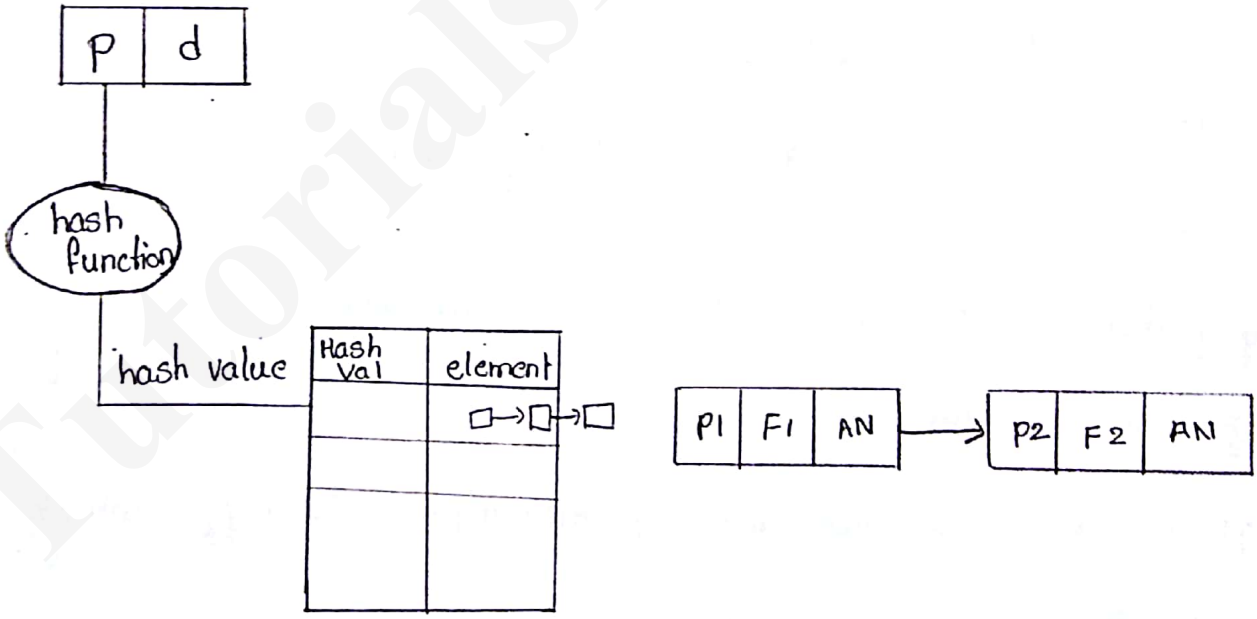


Fig: Hashed page table

Each element in hash table is a linked list of objects or nodes each node contains following information.

1: page number

2. Frame number

3. Address to the next element

3. Inverted page table :- In this page table can be defined for Entire main memory frames instead of all the pages in process.

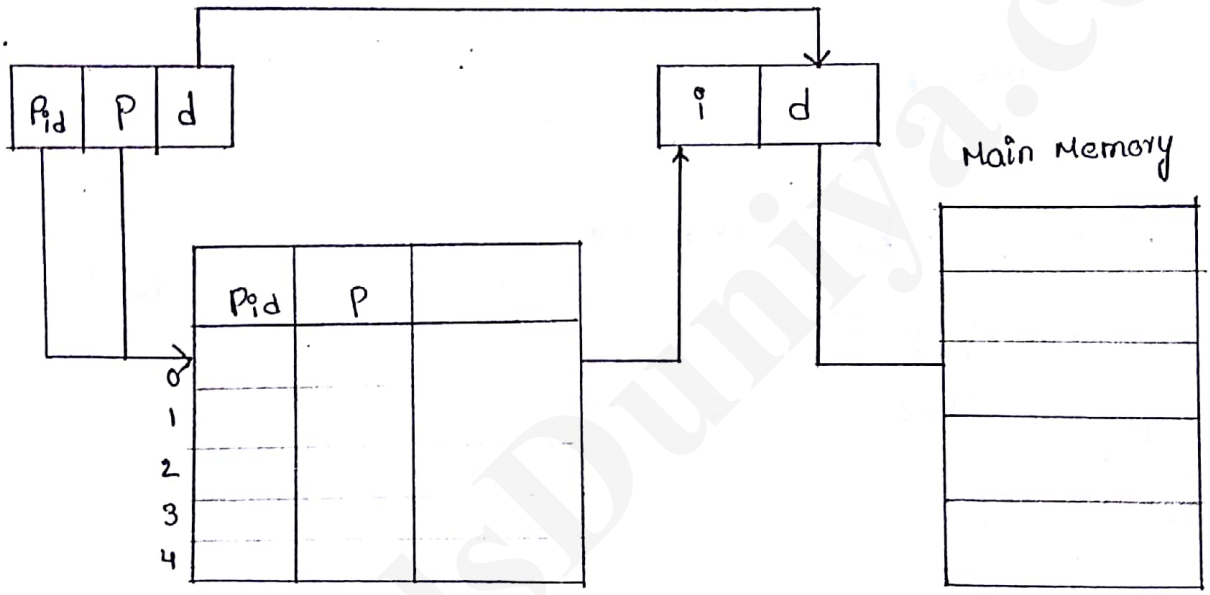


Fig: Inverted page table

Segmentation :- To avoid draw backs in paging, we are using the concept called pure segmentation.

→ It is non-continous memory allocation strategy which uses MVT

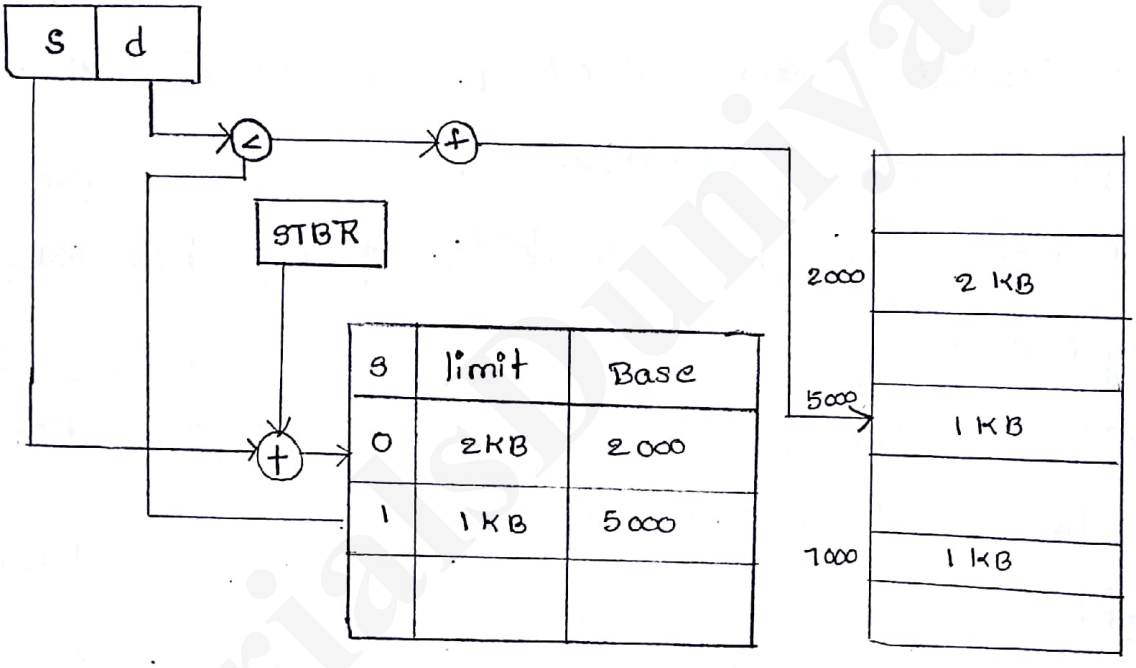
→ In this process is divided into segments like code segment, data segment etc.

→ sub programs are treated as one segment.

→ The main memory or physical memory can't be divided into frames.

→ In this os can construct a segmentation table while load the process into physical memory.

→ The segmentation table contains segmentation number, limit of segmentation and base of the segmentation.

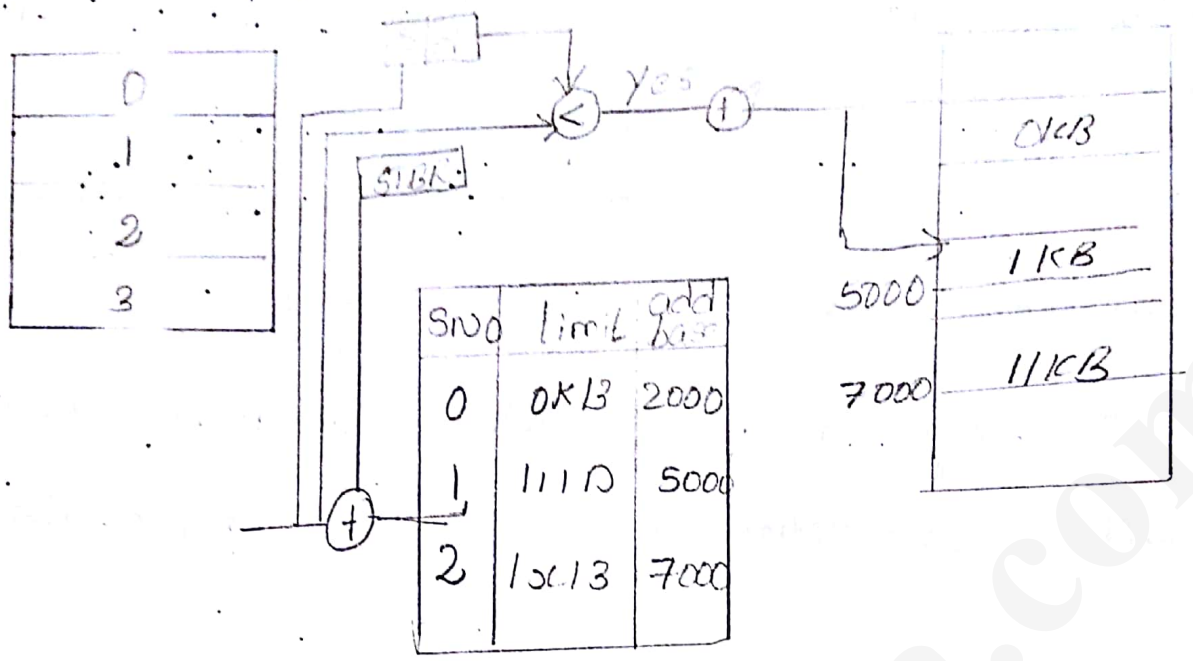


Paged segmentation :-

→ A process is first divided into segments. Each segment is divided into pages.

→ At the time of process loading we can construct 2 levels of table.

→ 1st level includes segment number, Base address of page table. Each segment has page table.



Virtual memory :- Load part of process in physical memory and start process execution.

→ A memory which is available logically but not available physically.

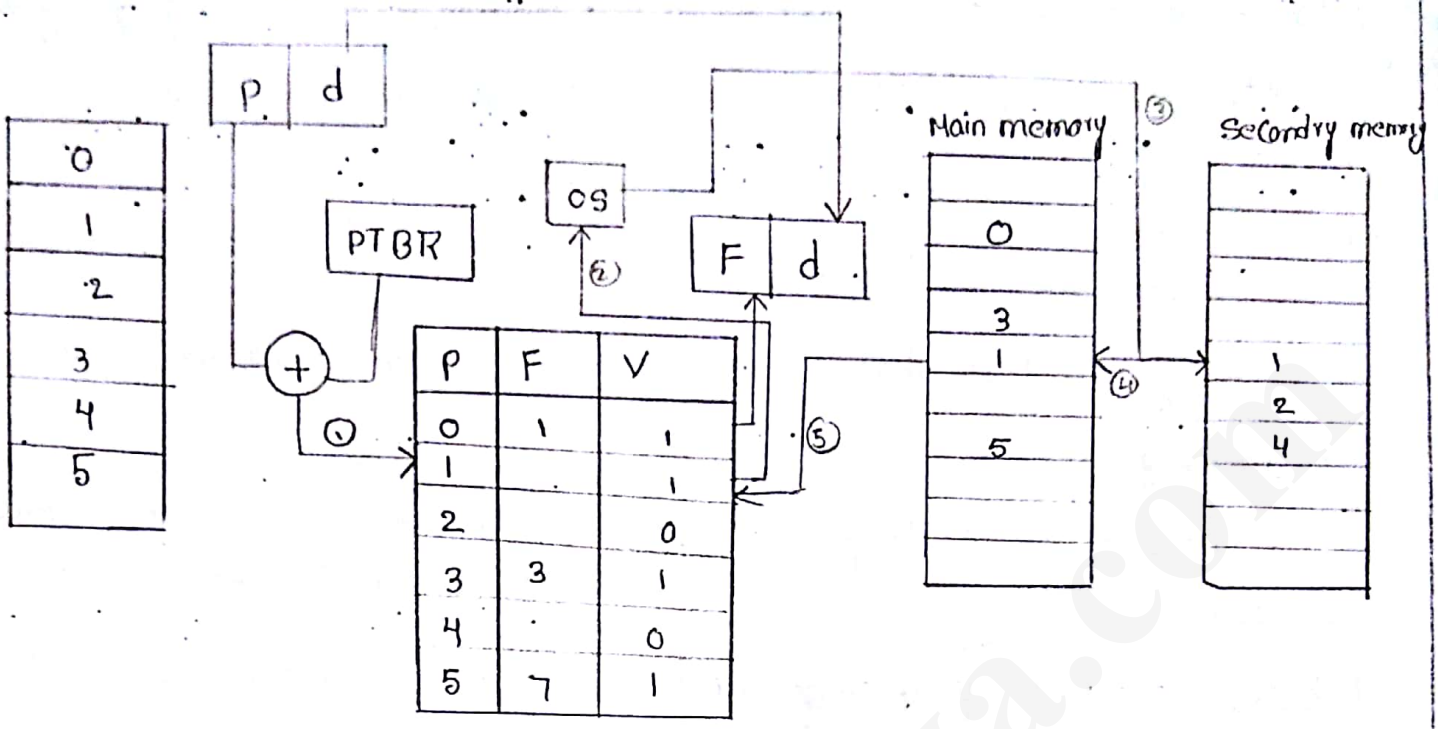
→ To start the process execution even though entire process is loaded into main memory.

Benefits :-

1. It allows to execute a process whose size is larger than physical memory.
2. It is used to increase the range of multi programming by loading more no. of process into physical memory.

paging with virtual memory :-

→ page table consists of page number(P) Frame number(F) and validity (V)



→ If required page is not available in main memory then it is called page fault

→ If demanded page is not available in main memory then we take it from secondry memory is called demand paging.

→ If a page is loaded before its requirement is called pre-paging.

Demand paging :- An alternative strategy is to initially load pages only as they are needed. This technique is known as demand paging and is commonly used in virtual memory system.

→ With demand-paged virtual memory, pages are only loaded when they are demanded during program execution.

→ A demand-paging system is similar to a paging system with swapping, where processes reside in secondry memory:

→ When we want to execute a process, we swap it into memory. Rather than swapping the entire process into memory.

→ Access to a page marked invalid causes a page-fault trap.

Steps in handling a page fault:-

1. We check an internal table for this process to determine whether the reference was a valid or an invalid memory access.
2. If the reference was invalid, we terminate the process.
3. We find a free frame
4. We schedule a disk operation to read the desired page into the newly allocated frame.
5. When the disk read is complete, we modify the internal table kept with the process and the page table to indicate that the page is now in memory.
6. We restart the instruction that was interrupted by the trap. The process can now access the page as though it had always been in memory.

page replacement algorithms :- To select one page for replacement. There are 3 page replacement algorithms.

(1) FIFO

(2) optimal

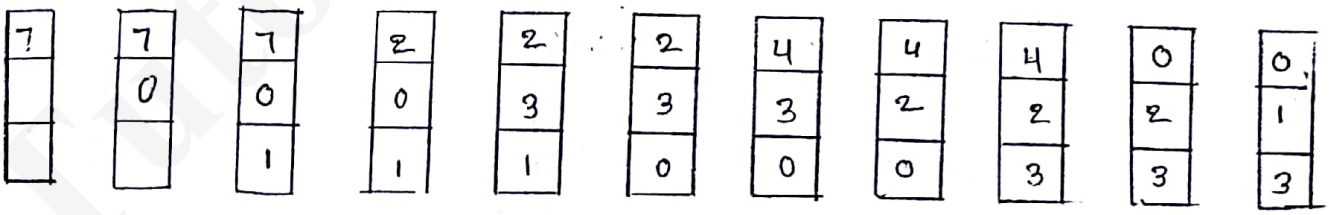
(3) LRU

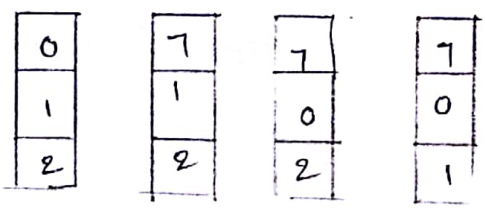
(1) FIFO page replacement algorithms :-

The simplest page-replacement algorithm is first-in, first-out (FIFO) algorithm. A FIFO replacement algorithm associated with each page the time when that page was brought into memory. When a page must be replaced, the oldest page is choosen. The FIFO page-replacement is easy to understand and program However, its performance is not always good.

Now, we consider the following reference string and apply FIFO page replacement algorithm

7 0 1 2 0 3 0 4 2 3 0 3 2 1 2 0 1 7 0 1





(2) optimal page replacement algorithm :-

→ An optimal page replacement algorithm has the lowest page-fault rate of all algorithms.

→ Replace the page that will not be used for the longest period of time.

→ Use of this page-replacement algorithm guarantees the lowest possible page fault rate for a fixed number of frames.

Ex:- Reference string

7 0 1 2 0 3 0 4 2 3 0 3 2 1 2 0 1 7 0 1

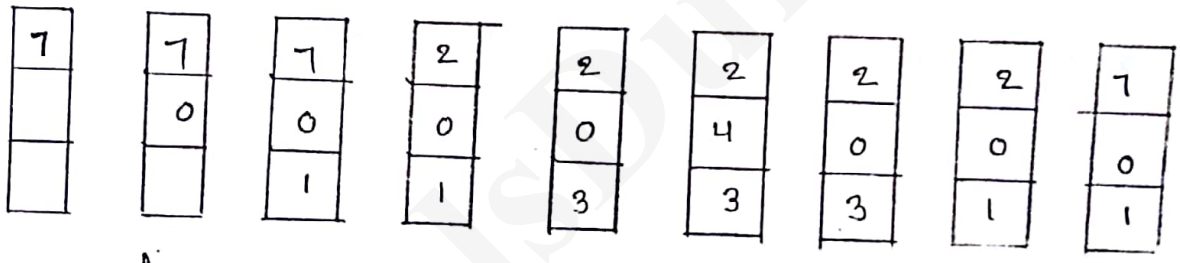


Fig: optimal page replacement algorithm.

(3) LRU page replacement algorithm :-

The key distinction between the FIFO and OPT algorithms is that the FIFO algorithm uses the time when a page was brought into memory, where as the OPT algorithm uses the time when a page is to be used. If we use the recent past as an approximation of the near future,

then we can replace the page that has not been used for the longest period of time. This approach is the least-recently-used (LRU) algorithm:

→ LRU replacement associates with each page the time of that page's last use. When a page must be replaced, LRU chooses the page that has not been used for the longest period of time.

Ex: Reference string

7 0 1 2 0 3 0 4 2 3 0 3 2 1 2 0 1 7 0 1

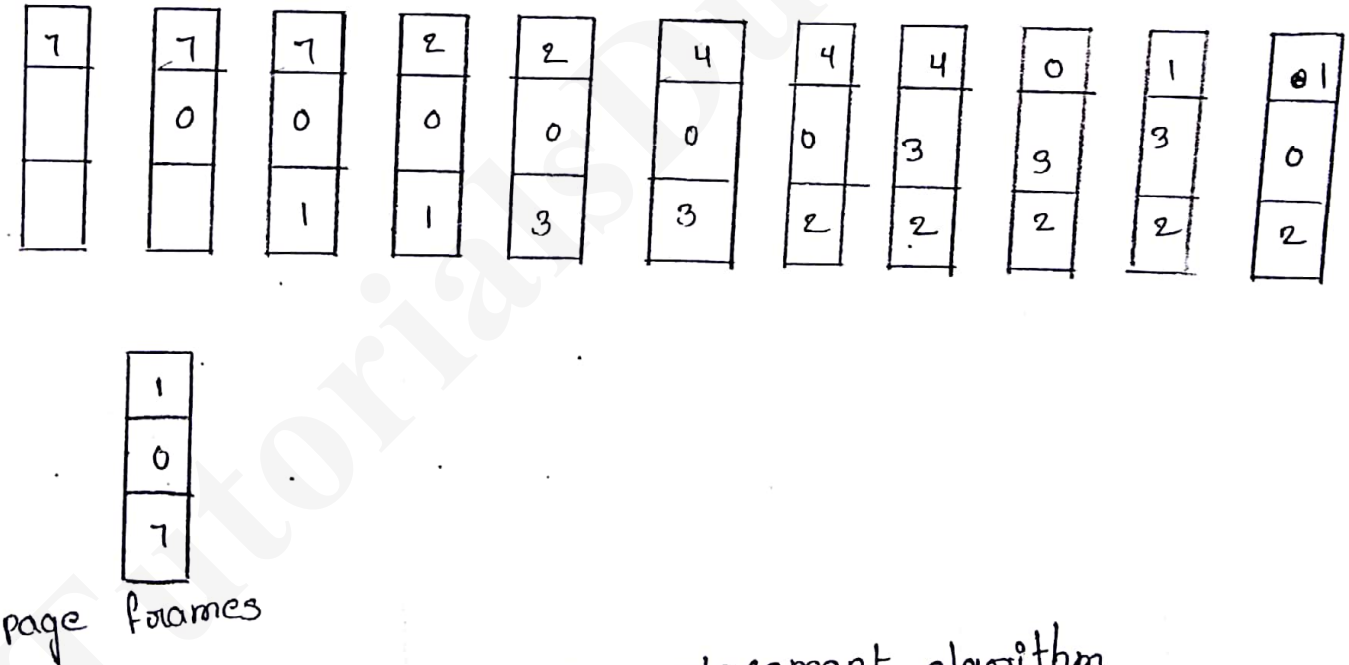


Fig: LRU page-replacement algorithm.

TutorialsDuniya.com

Get FREE Compiled Books, Notes, Programs, Books, Question Papers with Solution* etc of following subjects from https://www.tutorialsduniya.com.

- C and C++
- Programming in Java
- Data Structures
- Computer Networks
- Android Programming
- PHP Programming
- JavaScript
- Java Server Pages
- Python
- Microprocessor
- Artificial Intelligence
- Machine Learning
- Computer System Architecture
- Discrete Structures
- Operating Systems
- Algorithms
- DataBase Management Systems
- Software Engineering
- Theory of Computation
- Operational Research
- System Programming
- Data Mining
- Computer Graphics
- Data Science

- Compiled Books: https://www.tutorialsduniya.com/compiled-books
- Programs: https://www.tutorialsduniya.com/programs
- Question Papers: https://www.tutorialsduniya.com/question-papers
- Python Notes: https://www.tutorialsduniya.com/python
- Java Notes: https://www.tutorialsduniya.com/java
- JavaScript Notes: https://www.tutorialsduniya.com/javascript
- JSP Notes: https://www.tutorialsduniya.com/jsp
- Microprocessor Notes: https://www.tutorialsduniya.com/microprocessor
- OR Notes: https://www.tutorialsduniya.com/operational-research

 UNIT - 5

DeadLocks & I/O devices

Dead lock :- It is a problem occured in multi programming while applying synchronization.

In a multi programing system two or more processes try to access a shearable resource. In general a process follows the given steps to access a shearable resource.

1) **Request :-** In this the process need to request the o.s to acquring one instance of a resource.

2) **Usage :-** After acquires the resource the process executes operations by using the resource.

3) **Release :-** After using the resource the process need to release the resource and returning to the os.

The os maintaince information about system resourcess in a table called as system table. The system table contains following information.

1. How many types of resourcess available in the system.

   Ex :- cpu, memory, files, Ilo devices.

2. No. of instancess of each resource type.

3. If the resource is allocated to any other process or not. In case of the resource allocated to a process the system table stores PID of the process.

4. In this a process request 'n' no of resources i.e not exceeds the total no of resources available in the system.

→ If the required no of resources are not available then the process put in waiting state.

→ The process wake up from waiting state when the required resource is available.

→ A process can voluntarily releases the resource at the End of its execution.

Def:- In a system a waiting process start its execution when an event (releases the resource) performed by any other process that is already in waiting queue is called dead lock.

Ex:- In a system consists of two CD drives and one printer. Two process P1 & P2 are Executed P1 requires two CD drives and one printer for it's execution. P2 requires 1 CD drive and 1 printer for its Execution.

Now P1 first acquire 2 CD drives and wait for printer already allocated to process P2. process P2 acquire printer and wait for CD drive already allocated to process P1.

Characteristics of dead lock :-

Dead lock can arrise in a system when the following four conditions are occurs simultaniously in the system.

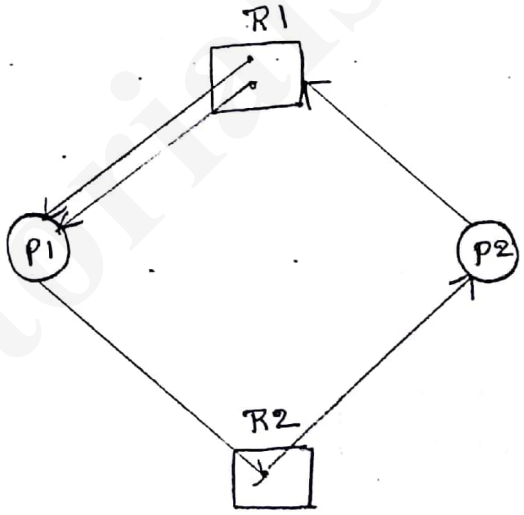
1) Mutual Exclusion :- If a sharable resource have only one instance, then it allows only one process to use the sharable resource and put the remaining process in waiting queue.

2) Hold and wait :- It ensures that a process hold the acquired resources and wait for the resources that are not available freely.

3) No preemition :- The process does not releases the acquired resources before completion of process execution.

4) Circular wait :- A system contain 'n' no of process {P1, P2, --- Pn}, P1 wait for P2, P2 wait for P3 --- Pn-1 wait for Pn and Pn wait for P1.

DAG or RAG (Resource allocation graph) :-



It is an alternative technique to represent dead lock diagrametically. In this a graph is drawn to describes a system model. The graph contains two models. i.e vertices and edges.

There are two types of verticess available.

1) process :- All the process in a system represented as circle.

2) Resourcess :- All the resourcess in the system represented as reactangles. The no. of instancess of a resource is represented by specifying dots in the react angle.

Edges :- There are two types

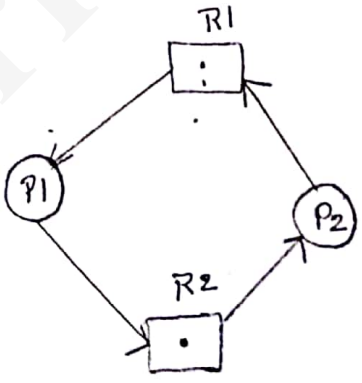
Requesting edge :- It is used to a process request a resource.

Ex :- P1 → R1

Assignment edge :- used to represent a resource assigned to a process.

Ex :- P1 ← R1

If the graph contains any cycles, there is a dead lock occurs in the system.



No. of process P = { P1, P2 }

No. of resourcess R = { R1, R2 }

Edges E = { P1 → R2, P2 → R1, P1 ← R1, P2 ← R2 }

Handle dead lock :- To handle dead locks we must follow three ways

1) prevention
2) Avoidence
3) Detection and Recovery

1) Dead lock prevention :- In this the system never Enters into the dead lock condition. To ensures this (not handle) or prevent the occurence of any one of the four necessary condition for dead lock occurence. Each condition

Mutual Exclusion :- Mutual exclusion is implemented on non-sharable resources.

(i) A resource is used by only one process at a time is called non sharable resources.

(ii) Mutual exclusion is not necessary for sharable resource.

(iii) A resource used by multiple process at a time is called sharable resources.

(iv) so, By using sharable resource in the system as possible we can prevents the occurence of mutual exclusion but some resources are intensively or by default non sharable. so for these type of resources mutual exclusion is hold.

Hold and wait :- In this the process is to be allocated the available resources and wait for required resources. To ensures that not hold this condition in the system we

can follow the given two solutions.

1) A process is to be allocated the resources if and only if all the required resources are available otherwise the process put in waiting state and no one resource is allocated to the process.

2) A process initially allocate available resources and start execution when the process need to request additional resource, the process must release all the resources currently allocated.

This solution ensures that a process release all its allocated resources before going to wait state.

Ex:- A process need to read data from CD drive and save it in a file available on hard disk print data in the file using printer.

Solution:- The process required CD drive, hard disk and printer to solve this task, but CD drive and hard disk are available. Now process read data from CD drive and save it on hard disk. After this the process releases both CD drive and file and put a request for file and printer.

No preemition:- This condition ensures that the process never returns allocated resources before completion of its execution.

→ To not hold this condition in the system we can use following protocol.

→ Their is another process in system request a set of resourcess that are already used by a process in waiting state. Then the system deallocates the resourcess from waiting process and allocate to the requested process.

Ex:-

If the process P1 acquires the resourcess R1 and R2 and starts the Execution. After some time P1 wants the resource R3, but it is not available then the P1 put into wait state, Then P2 request the resource R1 and R2, then it check for the availability of R1 & R2, if it is not available then put the process P2 in wait state.

Circular wait:- To not hold this condition in system it can follows given protocol.

→ In a system on all processes and all resourcess it can impose numbers in a sequential order.

→ The system can allocate resourcess to process P_j, its Execution is completed with all the resourcess allocated to all its pacior process in sequential order.

The dead lock prevention have following drawbacks.

1) In this resource utilization is low.

2) In this their is a possibility for occurence of starvation.

starvation means a process wait infinetly.

2) Dead lock Avoidence :-

In dead lock avoidence the system avoids the occurence of dead lock condition. It is implemented based on how resource requests are made. i.e The order of resources requested. Based on resource request order system will determine if the requested resources are granted then the system either enters into dead lock or not. If the system will not enters into dead lock then the resources are granted to the requested process. otherwise the requested process put in waiting state.

The system determines the dead lock state by using following information about resources availability of resources, allocated resources and need resources for each process.

Safe state :- A state is safe if it allocates requested resources to all the process and to ensures that the system will not enters into dead lock. A safe state produces a sequence of all the process executed it is called safe sequence.

Unsafe state :- It is also called as dead lock. A state which is not safe state. An unsafe state may leads to dead lock. All deadlocks are unsafe state. But An unsafe state is not always dead lock state.

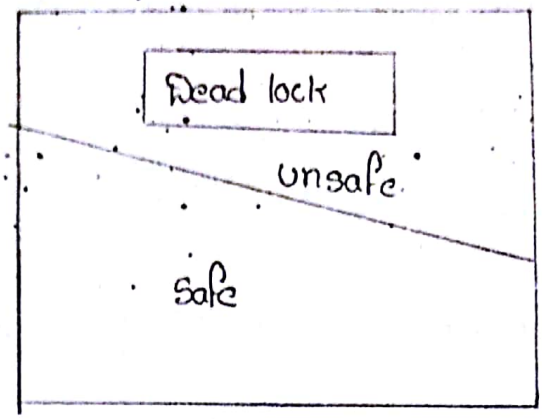


Fig: Control of state and unsafe state

Banker's Algorithm:- Dead lock avoidence can be implemented by using banker's algorithm. In this total requested resources are not allocated to a single process which leads to the dead lock of remaining process. In this algorithm we can use following data structures to represent a state of system.

We assume that a system consists of 'n' no of process and 'M' no of resources.

1) Available [m]:- It is used to represent availability of each resource type in system.

2) Max (n,m):- It is used to describe max no of resources requested by all the processes in system.

max (i,j)=k describes process Pi request for resource Rj type, k no of instances.

3) Allocation (n,m):- It is used to describes how many no of resources of each type allocated to each process

in system.

4) Need (n,m):- It is used to describe total no of resources need by a process to complete it's execution

need (n,m) = Max - Allocation

need (i,j) = k describes that k is no of instance of type R_j needed for the process completes its execution.

Safe state algorithm:-

step 1:- Take two vectors work of length 'm', finish of length 'n'. Initialize finish [i] = = false ∀ i=1 to m

work = Availability

step 2:- To find an 'i' such that

(a) finish [i] = = false

(b) need [i] ≤ work

If no search i is existed goto step 4

step 3:- finish [i] = true

work = work + allocation then

goto step 2

step 4:- If finish [i] is = = true ∀ i=1 to n then system is safe state.

Example :- A system consists of 5 process and 3 no of resources are as follows:

max :-

	A	B	C
P0	7	5	3
P1	3	2	2
P2	9	0	2
P3	2	2	2
P4	4	3	3

Allocation :

	A	B	C
P0	0	1	0
P1	2	0	0
P2	3	0	2
P3	2	1	1
P4	0	0	2

Need

	A	B	C
P0	7	4	3
P1	1	2	2
P2	6	0	0
P3	0	1	1
P4	4	3	1

Availability

A	B	C
3	3	2

(i) Work = Availability

	A	B	C
	3	3	2
P1	2	0	0
	5	3	2
P3	2	1	1
	7	4	3
P4	0	0	2
	7	4	5
P0	0	1	0
	7	5	5
P2	3	0	2
	10	5	7

The system is in safe state

The order of safe state sequence is {P1, P3, P4, P0, P2}

Resource Request Algorithm :- It is used to determine if the system is in safe state or not when a requested resources are granted to a process. In this the resources requested by a process or represented as request vector of size 'n'

step1 :- To test request <= need, if it is true goto step2, otherwise return an error the process Exceeds it's need.

step2 :- Test if request ≤ work then goto step3 otherwise stop the process execution untill requested no of resources are available.

step 3 :- In this the required requested resources are allocated to the process and do the following.

work = work - requested

allocated i = Allocated i + request

need i = need i - requesti

Now we can apply safty algorithm on the new state to determine the new state safe or not.

If the new state is safe then commits the transaction i.e goand the requested resources to the process. otherwise cancle or bault the transaction and restore the previous state.

Deadlock detection & Recovery :-

If the system can't follow either deadlock prevention (or) deadlock avoidance, then there is a possibility for the occurence of deadlock in the system.

Now the system can follows given 2 protocols.

1. Use the an algorithm for deadlock detection that occures in system.
2. Recovery from deadlock

Deadlock detection in 2 solutions available to detect deadlock occurence.

1. graph :- When system have single instance of each resource, deadlock condition is detected by drawing wait for graph.

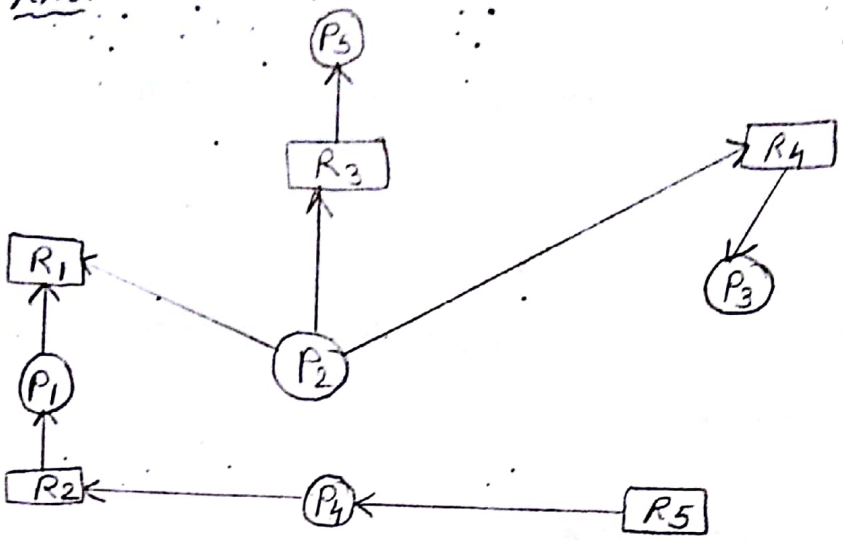
It is a graph derived from resource allocation graph. To derive wait for graph.

Given RAG we follows the given rules:

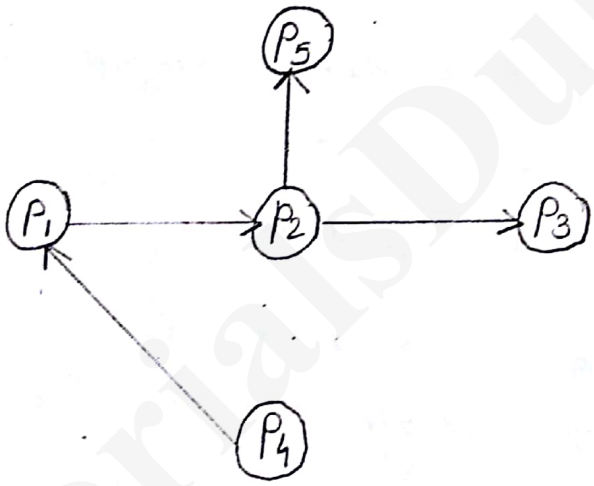
1. Remove vertices represent resources in RAG.
2. Cancel the edges connected to resources.
3. Draw an edge from $P_i \rightarrow P_j$ if and only if process P_i wait for a resource that is allocated to P_j (or) released by P_j.

Ex:- A System contain 5 resources $R_1, R_2 \cdots, R_5$

RAG



Wait for graph :-



If a cycle is found in wait for graph then deadlock is occured / detected, otherwise there is deadlock.

When cpu utilization below 40% deadlock.

The deadlock detection algorithm :-

In this algorithm is applied frequently it mostly when the cpu utilization rate is below 40%.

If the resources have multiple instances then the system can follow a detection algorithm. In this algorithm, we can also use as work as in Banker's algorithm. This algorithm is executed when for every request allocation.

In this 3 data structures are used.

1. Allocation
2. Request
3. Available

Algorithm :-

1. Take work[ ], finish[ ] of m, n Elements

   work[ ] = Available[ ]

   finish[i] = false for all i, 0 to (n-1)

   where,

   Alloc[i] != 0, otherwise finish[i] = true.

2. find an i such that

(a) Request$_i$ <= Work

(b) Finish[i] == false

If no such i not found, goto step-4

3. Finish[i] = true

Work = Work + Allocation

goto step-2

4. find Finish[i] == true for all is 0 to n-1 then there is no deadlock, otherwise deadlock occurs. Then process P_i is in deadlock state where Finish[i] = false.

Recovery from Deadlock :-

Once a deadlock is detected it is intima-ted to the operator. So that the operator handles deadlock manually. otherwise, deadlock is handled by the system automatically.

A system can recover from deadlock by following 2 alternatives

About the Execution of one (or) more deadlock processes :-

(i) Abort all the deadlocked processes at a time: In this increased burden to the system to execute all the processes from the starting.

(ii) Abort one deadlock process at a time until recover from deadlock.

Pre-emit resources of a deadlock process allocated to a deadlocked process. It in this the system can releases resources allocated to a deadlocked process. It follows 3 issues.

1. Select a victim (process):-
In this select one process for preemission of resources based on how many resources are allocated to the process.

2. Roll back:- If we preempt a resource from a process, it cannot continue with it's normal execution. It is missing some needed resource. We must roll back the process to some safe state and restart it from that state.

3. Starvation:- Every process is pre-Emitted only for fixed count

TutorialsDuniya.com

Get FREE Compiled Books, Notes, Programs, Books, Question Papers with Solution* etc of following subjects from https://www.tutorialsduniya.com.

- C and C++
- Programming in Java
- Data Structures
- Computer Networks
- Android Programming
- PHP Programming
- JavaScript
- Java Server Pages
- Python
- Microprocessor
- Artificial Intelligence
- Machine Learning
- Computer System Architecture
- Discrete Structures
- Operating Systems
- Algorithms
- DataBase Management Systems
- Software Engineering
- Theory of Computation
- Operational Research
- System Programming
- Data Mining
- Computer Graphics
- Data Science

- Compiled Books: https://www.tutorialsduniya.com/compiled-books
- Programs: https://www.tutorialsduniya.com/programs
- Question Papers: https://www.tutorialsduniya.com/question-papers
- Python Notes: https://www.tutorialsduniya.com/python
- Java Notes: https://www.tutorialsduniya.com/java
- JavaScript Notes: https://www.tutorialsduniya.com/javascript
- JSP Notes: https://www.tutorialsduniya.com/jsp
- Microprocessor Notes: https://www.tutorialsduniya.com/microprocessor
- OR Notes: https://www.tutorialsduniya.com/operational-research

I/O device Management :-

Based on I/O devices

1. Advanced I/O devices
2. Variety (or) new I/O devices

I/O Hardware :-

Port :- port is a connection point used to connect I/O device.

Bus :- A set of drives called bus. It is used to transfer data.

Controller :- A set of Electronics (transistors, diodes etc) used to controls the operation of I/O devices.

It receive commands from processor and Execute those commands.

System follow 3 types of I/O communications

1. Polling (or) programmed I/O :-

In this the processor want to perform on I/O operation on a device, it must wait And check again and again status of I/O device until the device is free.

Counters having 4 registers

1. Data-in :- To read data by host into I/O device
2. Data-out :- Used by the host to write data on to I/O devices.
3. Status :- It is used to specify the status of I/O device i.e device is free/busy. It also describes busy status of the currently executed command. The command is executed completly/not.
4. Command :- It is used to specify command/devices data i.e going to be executed I/O devices.

In polling the processor (handshake handshakes) with these 4 registers.

1. In polling 1st checks if busy bit is 0 if 0 then device is free after that
2. Write command into command register & data is write into data-out and set the Command Ready bit - 1

CPU | Controller

(1) busy - 0

(2) Command Ready - 1

3) busy - 1
Executes the command

4) It status is - 0 Executed successfully

5) If any Errors are occur then checks & detects

Drawback :

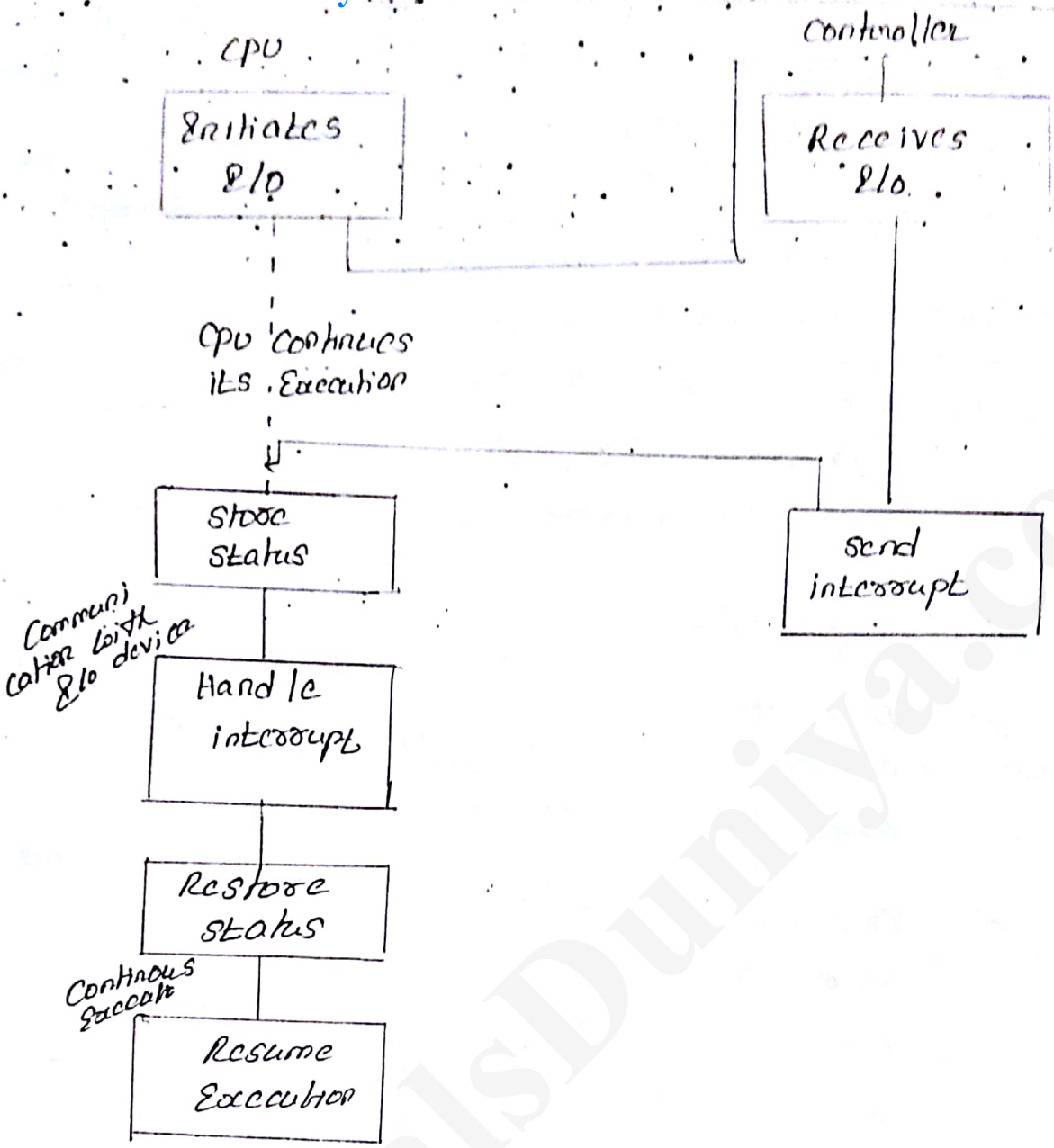
1. CPU wait / polling continously loss of processor performance.

To avoid this draw back use interrupted driven I/O.

To make the controller to send interrupt to CPU when the controller is free.

2. CPU also checks if interrupt I/O is occured or not for that add one more I/O device to CPU i.e interrupt request line. It is used to store an interrupt send by any I/O device controller.

3. The processor check the interrupt request line after executing every one instruction. The controller send an interrupt to the processor when it is free. once the processor knows that the device is free i.e read an interrupt then it handles the interrupt. After interrupt handling the processor returns to previous task and resumes its Executes as shown below.



Drawbacks:-

(1) It not check priority

(2) Which device is cpu need not find particular handler

To avoid those drawbacks we use maskable and non-maskable interrupts.

Maskable Interrupts :- These interrupts are not allowed in critical section Execution.

Non-Maskable :- These interrupts are allowed at any time.

DMA :- To provide direct communication blw main memory & main memory device.

To intimate that cpu can give control or bus to DMA unit.

& DMA send interrupt to cpu then the work is completed cpu gains the control of bus.

Applications of I/O :-

It provides set of techniques / interfaces to access I/O devices from application program

The I/O device are classified as follows:

1. Blocked stream & character stream I/O devices :-

In this device are classified based on mode of data transmission. In blocked stream data can be transmitted block by block.

Ex:- file or harddisk.

on blocked device we use read / write functions to access data.

character stream device :- In character stream device data transmitted character by character. Ex: monitor

N/w I/O :- These devices provided n/w transmission facilities Ex:- modem

Blocked I/O & non-blocked I/O :-

In non-blocked I/O the cpu need not to wait for the availability of I/O device in general all I/O devices are non-blocked nature.

In blocked I/O the processor need to wait until the device is available Ex:- keyboard.

Shared & Dedicated I/O :- The device shared by multiple computers Ex :- printers, scanners

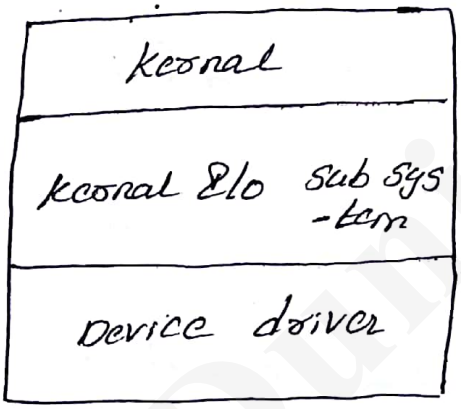
The dedicated I/O dedicated to only one host Ex: Hom

Synchronous & Asynchronous I/O :-

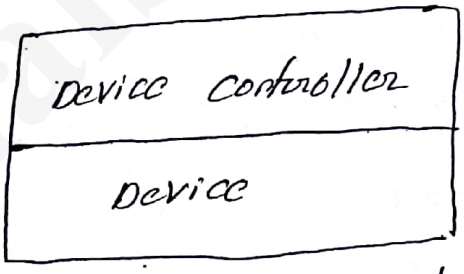
Synchronous I/O device must follows a schedule to transfer data

Asynchronous I/O device can't follows a schedule to transfer data.

Service provided by kernel I/O :-



The device driver sends commands to the device controller.



Kernal I/O subsystem provide the following set of services to provide communication b/w user process and hardware device.

(1) I/O scheduling :- In this kernal put I/O request in a order such that the order can improves the system efficiency and decreases the waiting time for a process

(2) Buffering :- It is storage area used to store data : Kernal provides buffering from I/O device for the following 3 Reasons.

(1) To coping speed mismatches b/w different I/O devices.

Ex :- Modem is slower then compared to hard disk.

(2) To adopt any two devices with two different data transfer sizes is used in networking applications.

(3) To support copy semantics while Executing read/write operations

(3) Caching :- Caching is a small memory with high speed. Kernal I/O maintain Cache to store data i.e recently accessed from I/O. device.

(4) Spooling :- It is also a buffer provided to output devices such as printer, it can't intercept the data send by multiple processes.

(5) Error handling :- Kernal automatically handles the simple Errors generated by I/O device.

The kernal returns an integer value to describe type of the error. i.e device is busy / failure If device is busy, kernal can re execute the I/O request.

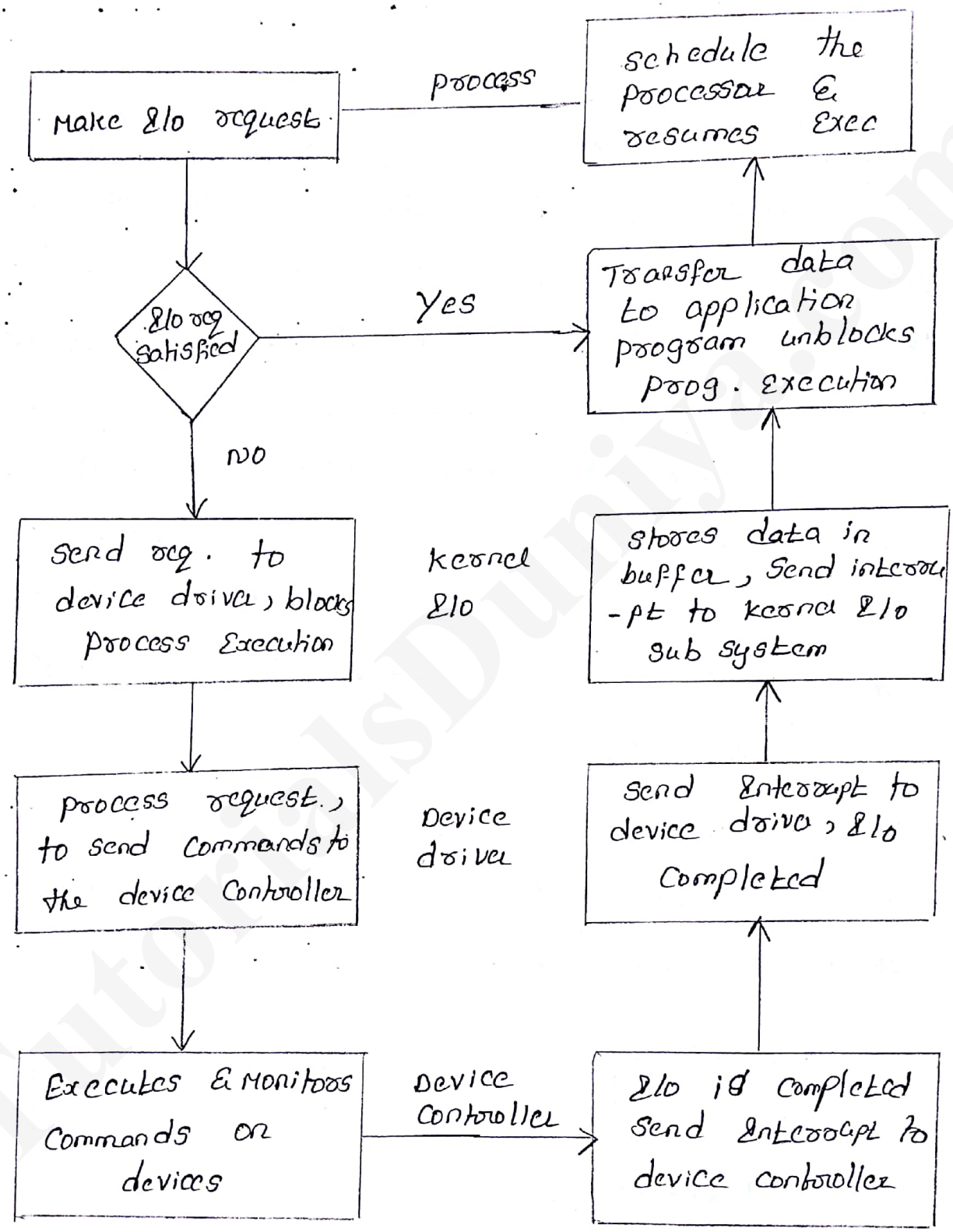
(6) protection :- kernal I/O provides protection while accessing I/O devices. In this kernal I/O can avoids the destruction of operating system due to the illigal I/O operations executed by user process. The kernal I/O defines all the I/O instructions as provilage so the I/O instructions are executed through kernal. I/O. now kernal I/O verifies every I/O operation before it's execution to know if it is ligal / illigal.

(7) Data structures :- kernal I/O maintains following data structures to perform I/O operations in an efficient manner

(1) file
(2) open
(3) status label
(4) n/w connection table etc

Lifecycle of I/O request :-

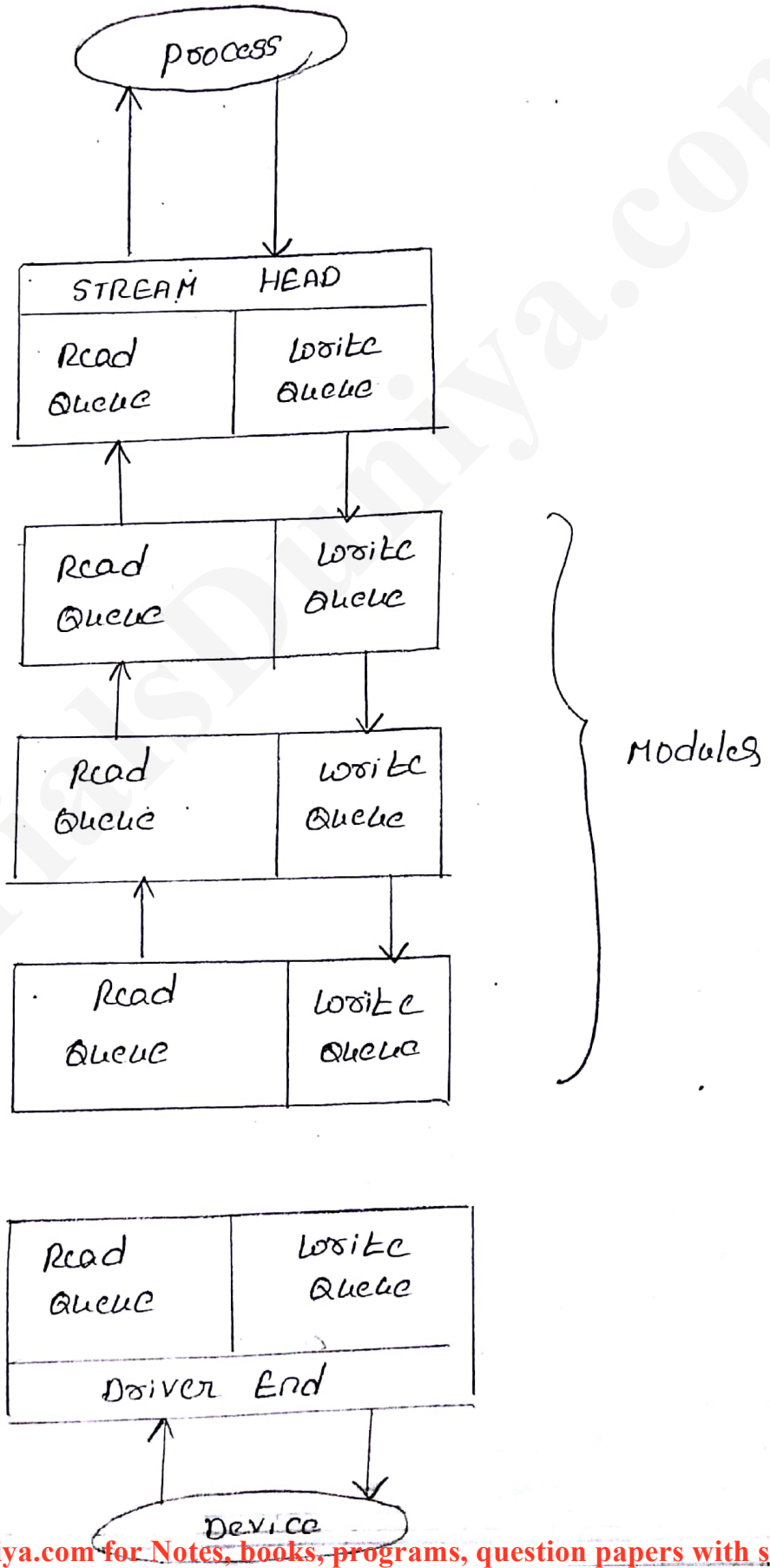
Transformation of I/O request to H/w :-



STREAMS :-

Stream can be defined as full-duplex connection b/w application process and h/w device. Streams are generally provided by UNIX operating system

Streams are used to transfer data blw application and device in the form of messages. A stream have given structure:



As shown in the above diagram Stream consists of following.

↳ Stream head : It is used to provide interface b/w process & stream modules

↳ Driver end : It is used to control the operation of H/w device.

MODULES :-

A stream consists of a set of modules to implement its functionality [transfer messages]

Each module contains two sets of queues

↳ Read Queue - used to store msg's read from device

↳ Write Queue - used to store msg's going to be write on device.

The module must apply flow control. that is a module queue either receive or send a when sufficient space is available.

Streams are asynchronous in nature. Stream can block the process execution until msg is transferred completely.

— End —

TutorialsDuniya.com

Get FREE Compiled Books, Notes, Programs, Books, Question Papers with Solution* etc of following subjects from https://www.tutorialsduniya.com.

- C and C++
- Programming in Java
- Data Structures
- Computer Networks
- Android Programming
- PHP Programming
- JavaScript
- Java Server Pages
- Python
- Microprocessor
- Artificial Intelligence
- Machine Learning
- Computer System Architecture
- Discrete Structures
- Operating Systems
- Algorithms
- DataBase Management Systems
- Software Engineering
- Theory of Computation
- Operational Research
- System Programming
- Data Mining
- Computer Graphics
- Data Science

- Compiled Books: https://www.tutorialsduniya.com/compiled-books
- Programs: https://www.tutorialsduniya.com/programs
- Question Papers: https://www.tutorialsduniya.com/question-papers
- Python Notes: https://www.tutorialsduniya.com/python
- Java Notes: https://www.tutorialsduniya.com/java
- JavaScript Notes: https://www.tutorialsduniya.com/javascript
- JSP Notes: https://www.tutorialsduniya.com/jsp
- Microprocessor Notes: https://www.tutorialsduniya.com/microprocessor
- OR Notes: https://www.tutorialsduniya.com/operational-research