Anil

link

HTML

Anil

BLACK BOOK

Date
Page

Unit - I

(Introduction)

OS Defination ⇒ An operating system is a system program that act as an interface (link) between the user of the computer and the computer Hardware. it gives the insviernment in which a user of computer can exicute program.

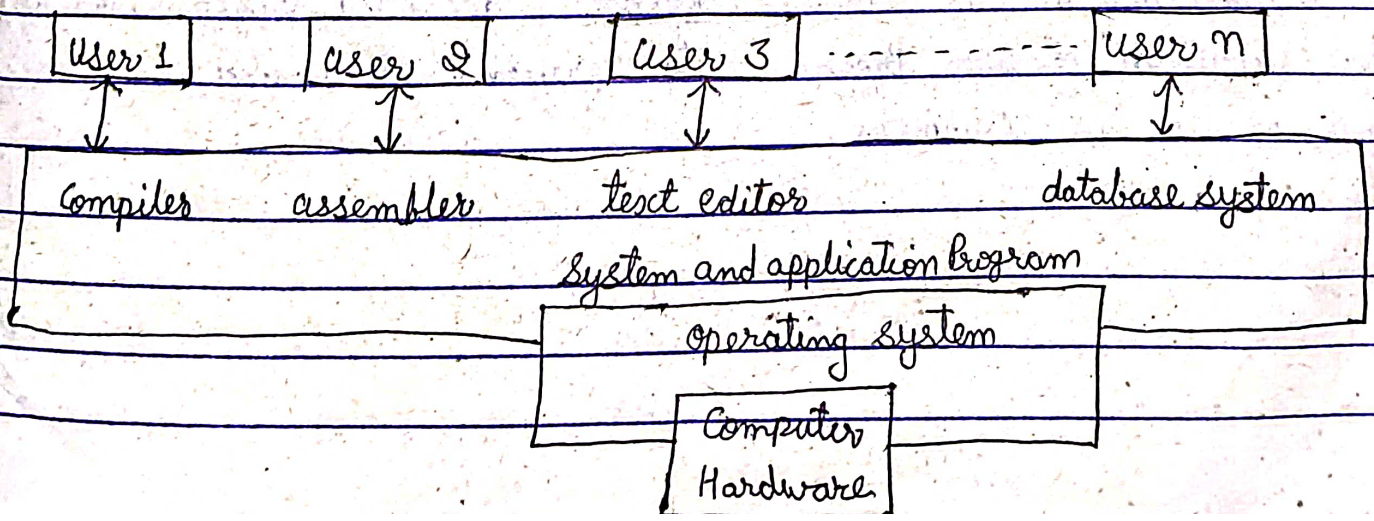
A Computer system can be divided ruffly into four components ⇒

1.) The Hardware ⇒ The CPU, The memory and input output divices Provide basic computing resources for the System.

2.) The application program such as word processor, spreadsheet, web browser define the way in which these resource are used to solve user computing ~~program~~ problem.

3.) The OS controls & codenate the use of hardware among various application for the various user.

4.) The User



I/O ⇒ MODEM, NIC, Touch Screen

The OS can be explored from the user point of view and system point of view.

1.) User view ⇒ Most users sit in front of PC and uses resources of the PC in this case OS is design for easy to use. Some attention is given to the performance of the system but no attention is given to resource utilization.

Some user share resource and exchange information in such case operating system is design to maximize resource utilization.

Some computer have little or no user view for example embedded computer in home devices have numeric keypad and indicate light to on & off and ⌀ to show status.

2.) System view ⇒ From system point of view we can view OS as resource allocator the OS manages resource and allocate to specific program and user as necessary for there task.

It can also be view as control program. A control program manages the execution of user program to prevent ether and in proper use of computer.

Part B

★ Function of OS

1.) starting of computer ⇒ when computer is turn ON it automatically begin to execute the boot program this boot program copy's the OS execution program into the main memory.

2.) Input output management ⇒ At any time a computer has to handle many different input and output processes in input device may send data to the computer at the same time CPU is sending data to an output device. the OS is responsible for managing this input and output processes.

3.) Memory management and CPU management ⇒ Some memory is used to store operating system (Kernel), application program instruction and data waiting for processing other area of memory is used for temporary calculation storing and storing intermediat result. In this it is OS job to allocate, asign memory to them. data that come from input divice and ready to be send to output divice are store in memory called buffer. The OS also deside for what data are to be send to the CPU for calculation.

4.) File management ⇒ Information disk is organized inform of files. A file is a collection of bites information about file such as name, size, date & time is stored on disk. This is the task of O.S

③ Types of OS ?

to manage all theise things

5.] Process management → A program in execution is called a process. A process is a unit of work the operating system can create or delete process, suspend or resume process, it provide machenisum for process senchronization, Communication and deadlock Handling.

6.] Data security ⇒ OS provide data security and makes the integrity. So that there is no conflict between various program and data.

7.] Communication ⇒ OS stablish the communication between computer system and user. it also genrate Various error massage as and when required.

8.] Re-source manager ⇒ OS distribute hardware between various user, recovery from falt, stablish user interface and management of input output.

★ Types of system ⇒

1. Main frame System ⇒ They are very large and fast but smaller and slower then supercomputer. They are used in centralized location where many terminals are connected with on CPU. and thuse allow different user to share

Super Computer ⇒ 4GH

single CPU. They have high memory and support thousands of user. they are genrally use in reserch organization, large industries, business and Government organization, bank and airline & resevation. when large size data is required. the processing speed of main frame computer is 100 to 900 MIPS [million instructions per second].

Example of main frame are IBM 4300 series, IBM 308X series and IBM 3090 series.

2.] Desktop System ⇒ Personal computer which are smaller and less expensive then main fram system are desktop system. In desktop system CPU utilization is not prime concern. It is a personal computer of form intended regular user at a single location

3.] Multiprocessor system ⇒ It is also known as parallel sysem or tightly coupled system. It has two or more processor in close communication, sharing the computer bus and some times the clock, memory and paripheral divices.

Multiprocessor system has three main advantage.

(i) Increased through put ⇒ By increasing the number of processor we accept to get more work done in less time. when multiple processor coprate a fask surten ammount of overhead occur in

Keeping all the parts working correctly.

(ii) Economy of scale ⇒ Multiprocessor system can cost less then equivalent multiprocessor single processor system. because they can share peripherals mass storage and power supplies. If several program operate on the same set of data it is cheaper to store those data.

(iii) Increased reliability ⇒ If functions can be distributed among several processor then failier of one processor will not halt the system, only slow it down.

Ex ⇒ If we have ten (10) processor and one face then each of 9 processor can share the work of fold processor and those the system runs 10% slower.

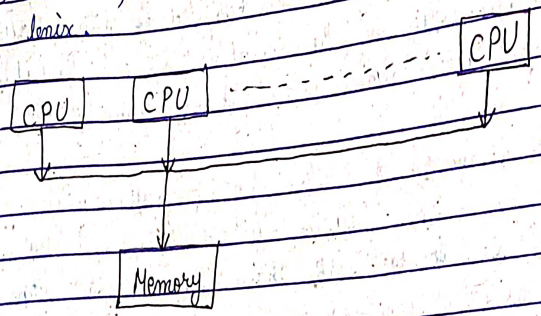
★ There are two types of multiprocessor system ⇒

1. Asymetric multiprocessor system ⇒ In this each processor is assigned a specific task a master processor control the system and other processor either look to master for instruction or have predefine task this scem defines a master slave relationship.

The master processor allocate work to slave processor.

2. Symetric multiprocessor system ⇒ In this each processor the operating system clear no master slave relationship perform all task within exist between processor.

Ex ⇒ Solaris, windows, windows XP, Mac OS and Unix.



4.] Distributed system ⇒ A distributed system is a collection of autonomous computer system or heterogeneous system. That are networked and are capable of communicating and coperating via there hardware & software by using distributed system we can increase computation speed, functionality, data availability and reliability.

Distributed system depend on networking for there functionality TCP/IP is the most common network protocal used so far.

Network are based on distance between there nodes a LAN connect computer within room or building. A WAN link building within a city.

A distributed operating system is one that looks to its user like an ordinary centralized operating system but run on multiple independent CPU.

In distributed system user are not aware of where there program are being run or where there files are residing. They should be handled automatically and efficiently by the operating system. Distributed system are more relible there uniprocessor system

⑤ clustered system ⇒ another type of multiple CPU system is the clustered system. It also have multiple CPU to do work they different from multiprocessor system as they composed of two or more individual system coupled together. They are linked via LAN.

clustering provide high availability service that is service will continue even if one or more system in the cluster fail. clustering can be structure asymmetrically or symmetrically.

In asymmetric clustering one machine is in hot stand by mode while other is running the application. The hot stand by host machine does nothing but monitor the active server.

In symmetric mode two or more host are running application and are monitoring each other.

⑥ Hand held system ⇒ It include personal digital assistant (PDA) such as palm and pocket PC, cellular Phone which has enbeded operating system due to limited size they are difficult to develop a PDA is about 5 inches in hight and 3 inch in width ..... So, it has small memory, slow processor and small display screen for fast processor more power is required so a larger battery is required which in turn require more space and need to rechange more frequently there is also lack of physical space which limits the input metual small keyboard and small output screen.

Hand held divices uses wiree technology such as bluetooth and wifi. Generally, limitation in PDA are balanced by there convenient and potability.

⑦ Real time system (RTS) ⇒

It is used in environment where large number of events mostly external to computer system must be accepted and processing is done in a short time or within deadlines.

Primary goal of RTS is to provide Quick response time, resource utilization and user convenience in this type of system processors are asigned priority and the processor are allocated to highest priority process.

Rocket Launching, telephone switching equipment and flight control are the application of real time system.

Real time system are classified into two types ⇒

1) Soft RTS ⇒ Video Conferencing is one of the application of such system if certain deadlines are missed nothing terible will occur only performance is degroded.

2) Hard RTS ⇒ In such system on processing must be done within the certain period of time if deadline is missed then system will fail.

In Read time operating system there is little swapping of program between primary and secondry memory time critical divice management, Interrupt management and input, output buffering are characterstics of RTS. Real time operating system is used when rigid time requirement have been placed on the operation of processor. Sensor bring data to computer then data is analysid. Real time System has well defined fixed time constraints.

Part - II

Operating System structure

★ Operating System Components :-

1) Process management ⇒ The OS manges many kind of activities ranging from user program to system program a process is a program in excicution the activities of operating system in regard To process management are -

(i) Creation and deletion of user and system process.
(ii) Suspention and resumption of process.
(iii) A mechanism for process synchronisation.
(iv) A mechanism for deadlock handling.

2) Main memory management ⇒ main memory provide storage and can be accessed directly by CPU the activities of operating system in regard to main memory management are -

(i) Keep track of which part of memory is currently being used and by whom.
(ii) Decide which process are loaded into memory when memory space become available.
(iii) Allocate and deallocate memory space as needed.

3) File management ⇒ A file is a collection of related information defined by its creator the activities of an OS in regard to file management are ⇒

① Creation and deletion of file and directories.
② The support for manipulating file and directories.
③ The mapping of file into secondary memory.
④ The backup of file on storage media.

4] I/O system management ⇒ The I/O system hides the peculiarities of specific hardware device from the user.

5] Secondary storage management ⇒ Secondary storage consist of tapes, disk and other media design to hold information that will eventually be accessed in primary storage. the major activities of OS in regard to secondary storage mangement are →
① Managing the free space available on secondry storage device.
② allocation of storage space when dus file have to be return.
③ scheduling the request for memory access.

⑥ Networking ⇒ The processes communicate with one another through communication line or network. The communication network design must consider routing and connection strethergies and problem of security.

⑦ Protection system ⇒ If computer system has multiple user and allows the concurant execution of multiple process then the various process must be protected from one another activities.

★ operating system services ⇒ OS provide and ensurement following are the services for user.

① programe exicution ⇒ Computer system allow user to system provide ensurement where user can run program. running a program involve the allocation and de-allocation of memory.

② Input output operation ⇒ The OS hide the details of runs the user program for effiuence and protection hardware for the I/O and user can not control I/O.

③ File system manipulation ⇒ The output of program may need to be return into new file or we need to use input from files. the user does not have to worry about secondry storage management user gives a command for reading or writting to a file and operating system makes it possible.

④ Communication ⇒ There are instenses where processor need to communicate with each other to exchange information it may be between processors running on same computer on running on different computer by providing thise service. the OS releves the users from the worry of

passing messages between processors.

⑤ error detection ⇒ OS constently monitor the system for detecting error. this free the user from the worry of error's as error can cost malfunctioning of the computer system.

⑥ Resource allocation ⇒ when there are ma multiple user or multiple job running at same time. Resource must be allocated to each of them. OS manages many types of resources such as CPU cycle, main memory, file storage, etc.

⑦ Accounting ⇒ we want to keep track of an record which user use how much and what kind of computer resources for accounting, billing of usage stastic.

⑧ Protection ⇒ The owner of information stored in multiple user computer system may want to control use of that information. Protection involve insureing that all access to system resource is controlled.

System call ⇒ System call provide an interface to the services mode available by an OS. System calls are available in assembly language example ~~sh~~ showing how system calls are used —

Written a simple program to read data from one file and copy them to another file.

Source file → Destination file

Example of system call sequence
acquire input file name
- write prompt to screen
- Accept input ~~acquire out~~
- Acquire output file name
- write prompt to screen
- Accept input
- open the input file
- if file does not exist abath
- Create output file
- If file exicet abath
- loop
- Read from input file
- write to output file
- Until read fails
- close the output file
- write completion massege to screen
- terminate normally.

* Types of system call ⇒

① Process control → ⊙ end, abort
⊙ load, execute
⊙ Create Process, terminate

- get process attribute, set
- wait for time
- wait event, signal event
- allocate, free memory

A running program needs to be abele to hold its execution either normally (end) or abnormally (abort). if a system call its made to terminate program a O.S. must transfer control to the invoker. Its command. a process exicuting one program may want to load and excute a another program Some time we want to terminate a process and to start a next process for K ( ). System call is use. the program is loaded in to memory by a exec ( ) and when the process is done it exicute exit ( ) system call to terminate.

⑤ File management ⇒

① create file, delete file
② open, close
③ read, write, reposition
④ get file attribute, set file attribute

we first need to be able to create and delete files one's the file is created we need to open it and use it we can also read, write or reposition the file finally we can close the file indicating that we are no longer using it file attribute includes file name, file type, protection code accounting information and so on.

⑥ Device Management ⇒

① Request device, relase device
② read write and reposition
③ Get device attribute, set device attribute.
④ logically attach or detach device

A process may need several resource to execute main memory, disk, drives, access to files and so or if the resourse are available they can be granted otherwise the processor has to wait t'S the resource are used they must be release.

⑦ Information maintanence ⇒

① Get time or date, set time or date.
② Get system data, set system data.
③ Get process, file or device attributes, set process, file or device attribute

Many system calls exist simplly for the purpose of transfaring information between the user program & operating system. System call may return information about system such as no. of current user, the version no. of OS, the amount of free memory or disk space and so on.

5] Communication ⇒

(1.) Create, delete communication connection.
(2) Send, recieve message.
(3) Transfer status information.
(4) Attach or dettach remote devices.

There are two common models of interprocess communication (IPC) the massege passing model and share memory model. massage can be exchange either directly or indirectly throw a common mail box. firstly connection must be open. in shared memory model processor's view share memory to create and gain access to region of memory owen by other processor.

☆ System Program ⇒

System Program Provide a convinient enviorment of program development and execution. they can be provided into following categries.

1. File management ⇒ This program create, delete, copy, rename, print, list and manipulate files and directories.

2. Status information ⇒ Some program simply ask system for date, time, amount of available memory, number of user or status information. othe

are complex providing detail performance, logging and debugging information.

(3) File modification ⇒ Several text editor are available to create and modify the content of files store on disk or other storage devices.

(4) Programing language support ⇒

Compiler, assembler, debugger and interpreter for common programing language are provided to user with the operating system.

(5) Program loading and exicution ⇒

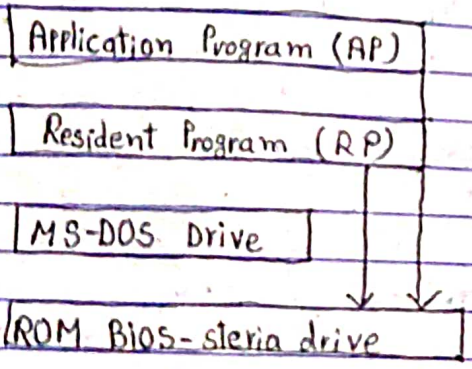
Once the program is compiled or assembled it must be loaded in to memory to be exicuted the system may provide absolute loader relocatabel loader and linkage editor.

(6) Communication ⇒

These program provide the machenisum for creating virtual processor among processes, users and computer system. they allow user to send masseges to one another screen, to browse web pages, to send electronic mail massege, to transfer file from one machin to another machin

★ System Structure ⇒ As modern operating system are large and complex carefull engineering is required when we want to design operating system. There are four different structure.

① Simple structure ⇒ This type of operating system are simple small and has limited system. MS-DOS is an example of such a system it was written to provide most functionality in the list of space. So it was not divided into modules.



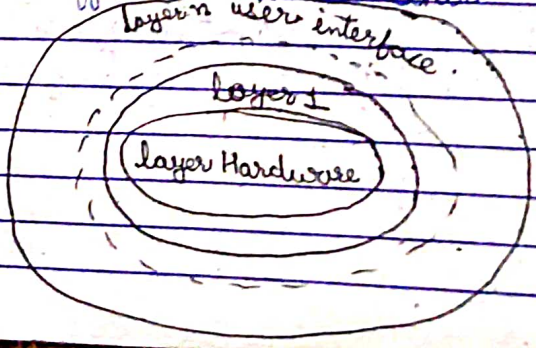
BIOS ⇒ Basic input output system.

In MS-DOS the interface and levels of functionality are not well seperated. Application program are addi to access the basic input output routines to write directly to the display and disk drive. MS-DOS is also limited by Hardware. Another example is limited structuring is the unix - operating system it has two parts the kernal and the system program. The kernal is further seperated into the series of interfaces and device drivers.

② Layered structure ⇒ with proper hardware support OS can be broken into pecies that are smaller and more appropiate than those allow by the original MS-DOS or UNIX system. Here topdown approach is used in layered approach OS is broken into a number of layer the bottom layer into a number of layer in the bottom layer. The bottom layer layer 0 is the hardware. the highest layer layer n is the user interface.

The main advantage of layer approch is simplicity of construction and debugging the layer are selected so that each user function and services of only lower level layer the layer does not need to know how these operation are implemented it need to know what there operation do each layer heights data structure operation and hardware from high level layer.

i) Defining the layers, as layers can use only lower level layers, carefull planning is necessary.
ii) It is less efficient than other type.



(3) Micro Kernal ⇒ This method structures the operating system by removing all non-essential component from Kernal and implementing them as system and user level program the main function of the micro Kernal is to provide communication facility between client program and various services running in user space communication is provided by massage passing one benifit of micro kernal approch is ese ease of extending the operating system as the services are added to user space and no modification of Kernel it also provide most security and reliability.

Ex ⇒ True 64 unix
QNX. windows DNT

(4) Module ⇒ The best method for operating system design involve using OOPS (object oriented programing) technique to create module Kernel now Kernel has a set of core component and dynamic links in additional service either during boot time or during run time.

Ex ⇒ Solaries, linux, mac osx.

The solaries operating system structures is organised around a core kernel with seven type of loadable Kernel module.

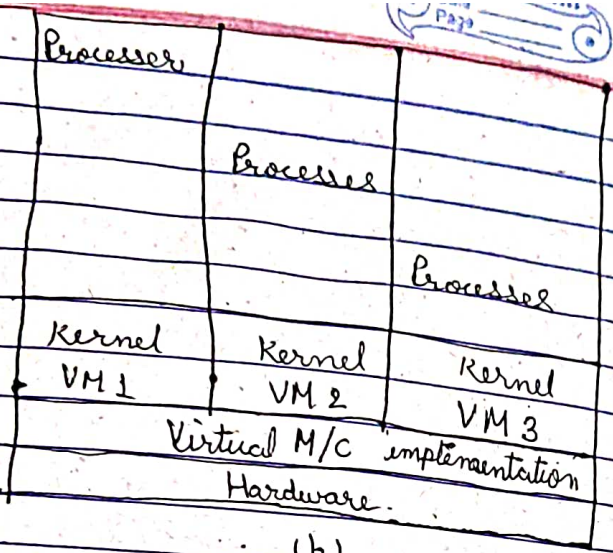
It is more flexiable then the layered approch here the Kernel provide core services yet also allows certain features to be implimented dynamically.

For example ⇒ Device and bus drivers for specific hardware can be added to the Kernel and support for different file system can be added as loadable module.

★ Virtual machine ⇒ The Idea behind virtual machine is to abstract the hardware of a single computer (CPU memory) into several different execution enviorment there by creating the illusion that each seperate execution ~~env~~ environment is running its own ~~illusion~~ private computer the operating system create illusion that a process has it own processor with its own memory.

The reason for creating virtual machine is to share the same hardware yet run several different ese execution environment concurrently.

(a)



(b)

System models ⇒ (a) Non virtual m/c
(b) virtual Machine.

Important Question

1. what is system call? explain types of system call provided by an OS.
2. Describe the main advantage of an OS designer and for a user in using virtual machine architecture.
3. what is OS Discuss function of OS in detail? (2014)
4. Explain multiprocessor system?
5. what r different types of an OS?
6. what are goal of an OS describle about various services provides by an OS.
7. what are advantage and disadvantage of multiprocessor system? Described distributed and clustered system.
8. write a short note on (a) System call (b) virtual call (c) operating system services.
9. Describe difference between symmetric and asymmetric multiprocessor and write advantage and disadvantages. (2011)
10. Describe distributed and clustered system. (2011)
11. what are major activities of OS in regard to file mgt. (2011)
12. write a purpose of system calls and system program. (2011)

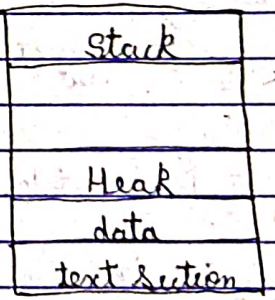
Unit - II

(Process)

A Process is a Program in execution. It is unit of work in a mor moden time sharing system. A system is a collection of Processor's.

A Process is more then Program Code which is some time known as text system section. It also include current activities as represented by the value of Program counter (PC) and the content of Processor's register. A Process include Process stack which contain's temporary data such as function Parameter return address and local variables.

It also has a data section which contain's globle variable. A Heap which is memory that is dynamic allocated during process runtime a process is a active entity with a program counter specifing the next instruction to execute and a set of accosiated resources.



Process in memory.

☆ Process state ⇒ As a process execute it change state. A The state of the process is define in parts by the current activity of that process. Each process may be in one of the following state :

① NEW ⇒ The Process is being created.
② RUNNING ⇒ Instruction are being executed.
③ WAITING ⇒ The process is waiting for some event to occure.
④ READY ⇒ The process is waiting to be assign to a processor.
⑤ TERMINATED ⇒ The process has finished execution.

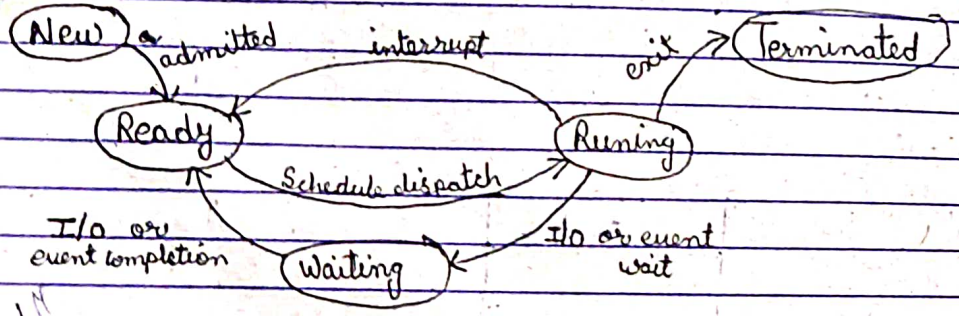


Diagram of Process state

NOTE ⇒ Only one process can be running on any processor at any instant, Many processes may be ~~are~~ ready and waiting however.

Process control Block ⇒

Each Process is represented in OS by a process control block (PCB). also called task control block. It contain information associated with specific process such as.

① Process state ⇒ The state may be new, ready, running, waiting and so on.

② Program counter ⇒ The counter indicate the address of the next instruction to be executed for this process.

③ CPU Register ⇒ There are many registor such as accumulators index register, stack pointer, general purpose registers etc. Thise are used to store information when interrupt occurs.

④ CPU scheduling Information ⇒ This information include a process priority, pointers to scheduling queues and other ~~oth~~ scheduling parameters.

⑤ Memory management information ⇒ This information on include value of base and limit register, the page table, Sigment tables.

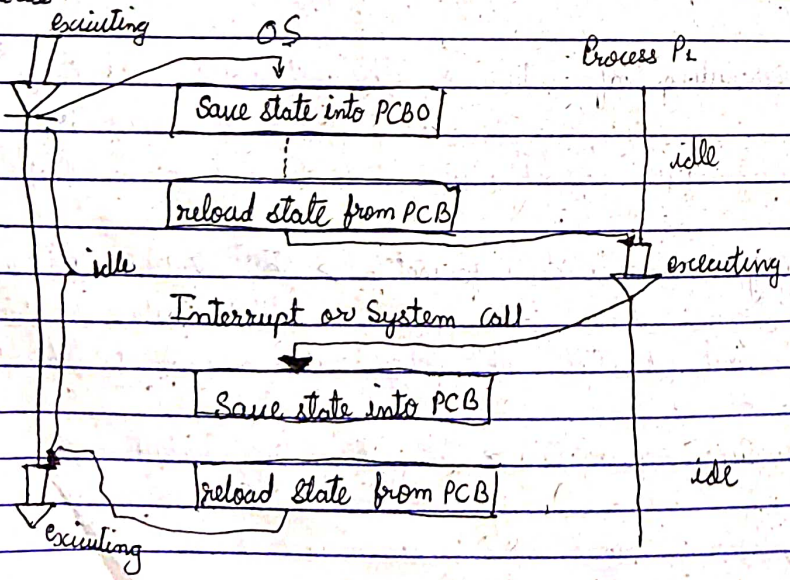
⑥ Accounting information ⇒ This information include the amount of CPU and real time used, time limits, job numbers and so on.

⑦ I/O status information ⇒ This information include the list of I/O device allocated to the process, a list of open files and so on.

Process state
Process Number
Program counter
registers
Memory limits
list of open files
...

Process Control Block.

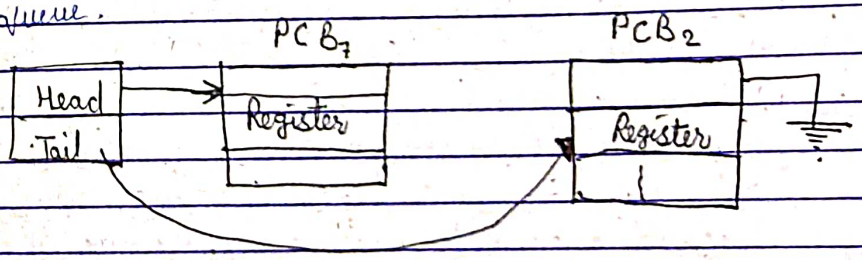
Fig 2 → Diagram showing CPU switch from process to be



★ Process scheduling ⇒ The objective of multi programing is to have some processes running at all time to maximize CPU utilization. the process scheduler select and available process for program execution on the CPU.

Sh Scheduling queue ⇒ As the process enter the system they are put into Job queue which consist of all processes in the system. the processes that are residing in main memory and are ready and waiting to execute are caped on a list called the ready queue.

The ready queue header contents pointer to first and final PCB in the list. each PCB include pointer that points to the next PCB in the ready queue.

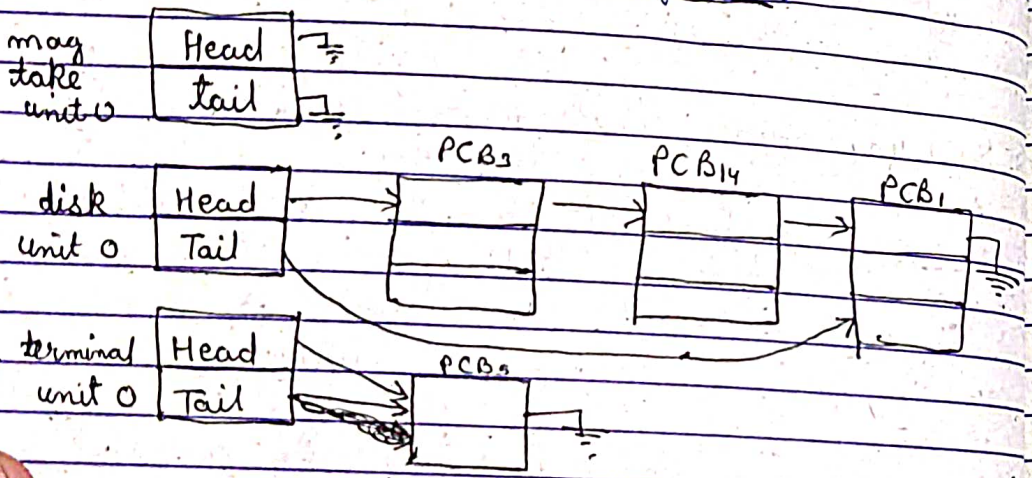


The ready queue.

The system also include other queue. when the process get CPU it executes for a while and eventually quite, is interrupted or wait for occurrence of a particular event such as completion of I/O request

The list of processes waiting for a particular I/O device is called device queue.

Each device has its own device queue.



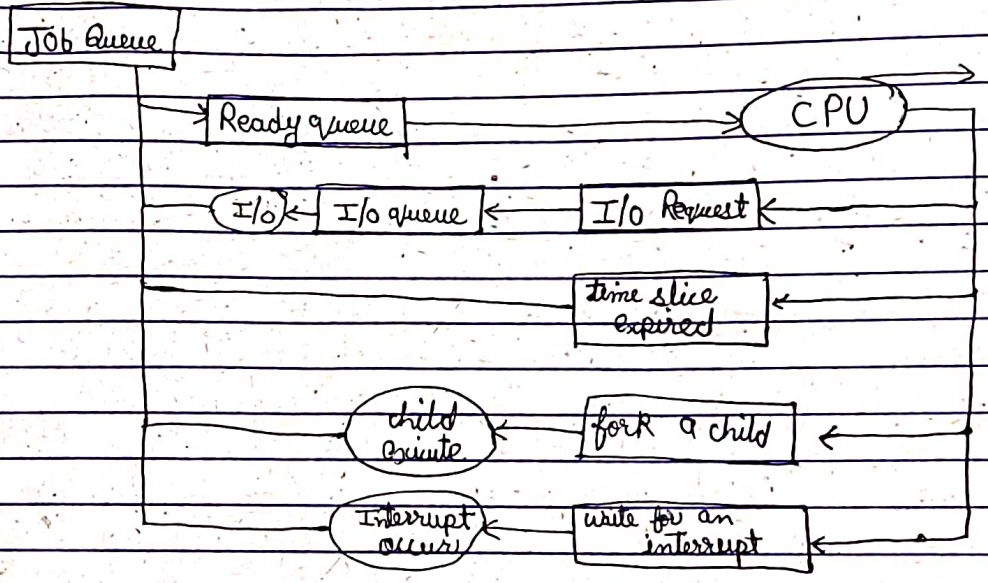
I/O Device Queues.

A common representation for Process scheduling is Queuing Diagram Each rectangl represent a queue, circle represent the resourse that serve the queue and the arrow indicate the flow of processes in the system. A new process is caped in ready queue then CPU is allocated to it and it start executing. One of several event could occure.

(1) The process could issue and I/O request and then we placed in I/O queue.

(2) The process could create new sub process and wait for its termination

(3) The process could be remove forcable from CPU as a result of interrupt and be put back in the ready queue.



Queuing diagram representation of process scheduling.

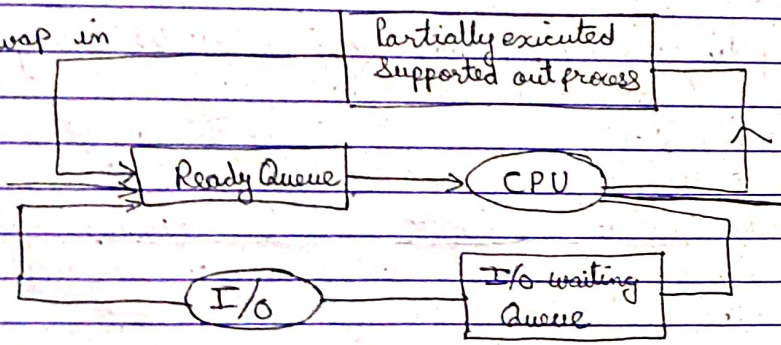
★ Schedulers ⇒ The OS must select process from queue. the selection process is carried out by schedulers. the long term scheduler or job scheduler select process from pool and loades them into memory for execution.

The short term scheduler or CPU scheduler select from among the processes that are ready to execute and allocates the CPU to one of them.

The short term scheduler must select for CPU frequently the short term scheduler execute atleast once every 100 milli sec. the long term scheduler execute much less frequently it control the degree of multiprograming it need to be invote only when a process leaves the system.

Some OS such as time sharing system may introduce and additional, intermediate level of scheduling this is midium term scheduler.

The idea behing medium term scheduler is that some times it can be advantages to remove process from memory and thus reduce the degree of multiprograming. the process is swapp swaped out and swaped in by the midium term scheduler.



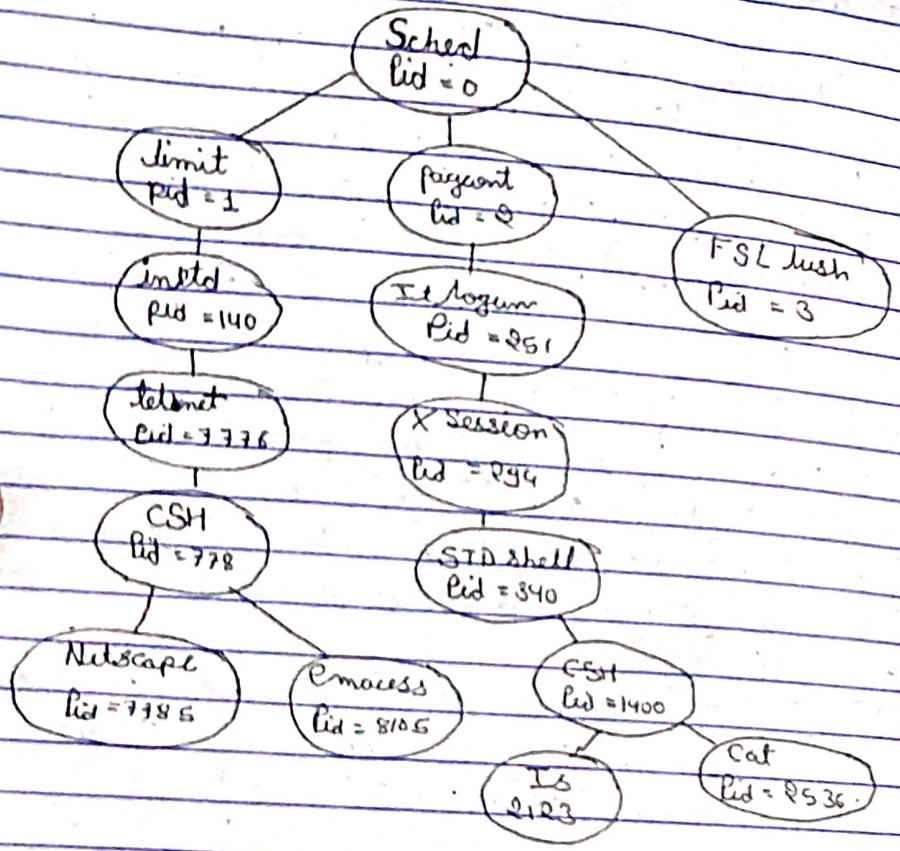
addition of midium term scheduling to the queuing diagram

★ Context switch ⇒ when and interrup occure the system need to save the current context of the process currently running on the CPU. So that it can restore that context when its processing is done spacially suspen asentially suspending the process and then resuming it. switch the CPU to another process requires performing a state save of the current process and state restore of different process this task as known as context switch.

★ Operation on Process

① Process creation ⇒ A process may create several new process via a create process system call. the creating Process is called parent process and the new process is called children of that process.
Each children process inturn create other processes forming a tree of a process process are identified by unique process identifier PId. which is integer no.

Fig. Shows process tree for the solaris OS. In this the process at the top of the tree is sched process with id 0. sched process create several children process including page out, fsflush and init init process serve. serve as grouped parent process for all user process. There are 2 children of init - inetd and dtlogin. inetd is responsible for networking services such as telnet and FTP. dtlogin is the process representing a user login screen



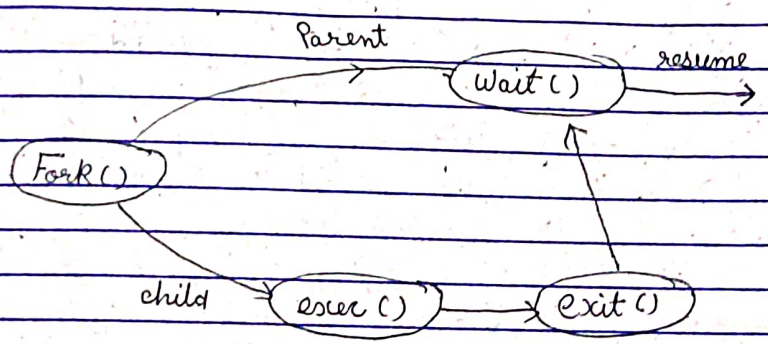
When a process create a new process to possibility exist in terms of execution.

(1) The Parent continue to execute concurrently with its children
(2) The Parent wait until some or all its children have terminated

There are two possibility in term of address space of new process.

(1) The child process is duplicate of parent process.
(2) The child process has new program loaded into it.

A New process is created by fork system call parent and child process communicate easily the return code for the fork is 0 for child process where as PID of child is return to the parent parent Exec system call is used after a fork system call by one of the two processes to replace the process memory space with a new program. the Exec system call loads a binary file and starts its execution a parent can issue a wait system call to move it self of the ready queue until the termination of the child.



(2) Process Termination ⇒ A process Terminate when it finishes executing its final statement and ask the operating system to delete it by using exit system call. all the resources of the process are de-allocated by operating system.

A parent can terminate the execution of one or fix children

for of variety of reasons such as -

① The child has exceeded its usage of some of resources that it has been allocated
② The task asign to the child is no longer required
③ the parent is exiting and operating system does not allow child to continue if its parent terminate.

In some system if the process terminat then all children must also be terminated this phinomena is called Cascading termination.

★ Interprocess communication (IPC) ⇒

A process is independent if it cannot effect or effectid by other process executing in the system any process that does not share data with other process is independent.
A process is cooperating if it is effected or it can effect other process executing in the system.

Reason for process co-operating ⇒

1. Information sharing ⇒ several users may be interested in same piece of information so current access should be allowed.

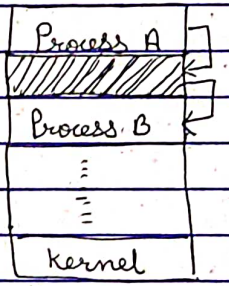
2. Computation speed up ⇒ If we want task to run faster we must break it into sub task and all can be executed parellely.

3. Modularity ⇒ we may want to construct the system in a modular fashion dividing the system function into seperate process or threads.

4. Convinience ⇒ even an individual users may work on many task at the same time for eg:- A user may be editing compilling & printing in parellel.

→ Co-operating process require an interprocess communication (IPC) mechanism that will allow them to exchange data and information. there are two models of IPC.
(i) Shared memory.
(ii) Message passing.

(i) Shared memory ⇒



In shared memory a region of memory that is shared by co-operating process is established process exchange information by reading and writing data to the shared region Shared memory allows maximum speed and convinience of communication. It is faster than message passing.

Producer Consumer Problem ⇒ A producer process produce information that is consumed by a consumer process. eg. A compiler may produce assembly code which is consumed by assembler.
One solution to the producer consumer problem uses shared memory a buffer is available which is filled by producer and emptied by consumer these buffer reside in shared region. the producer and consumer must be synchronised so that consumer does not try to consume an item that has not yet be produced there are 2 types of buffer.

(i) Unbounded Buffer ⇒ In these there is no limit on the size of the buffer consumer may have to wait but producer can always produce new item.

(ii) Bounded buffer ⇒ In their there is fixed sized buffer here Consumer has to wait if buffer is empty, and producer has to wait if buffer is full.

```
# define Buffer size 10
type of struct {
    ---
    ---
} item;
item buffer[Buffer size];
int in = 0; // pointer
int out = 0;
```

It is implemented as circular array with two pointers in and out.
in :- It points to next free location in the buffer.

Out :- It points to first full position in the buffer.

Buffer is empty when in == out
Buffer is full when ((in + 1) % of Buffer size) == out.

★ Code for Producer Process.

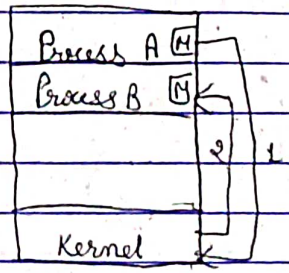
The producer process has local variable next-produced in which the new item to be produced is store.
The consumer process has local variable next-consume in which item to be consumed is stored.
Code for producer process.

```
item next product;
while (true)
{
    /* Produce an item in nextproduced */ while ((in+1)
    % of Buffering size) == out) // full; // DO nothing
    Buffer[in] = next produce;
    in = (in + 1) % buffersizing;
}
```

```
consumeer process
item next consumed;
while (true)
```

}
while (in == out) // empty
; // do nothing
next consumed = buffer [out];

(ii) Message passing ⇒



Message Passing model communication take place by means of message exchange between the co-operating processes. Message Passing provide a mechanism to allow processes to communicate and to synchronise their action without there sharing the same address space and is particularly useful in distributive environment.

If process P & Q want to communicate they must send message to and recieve message from each other, a communication link must exist between them.

Several med methods for implimenting link are ⇒

1. Direct or indirect communication
2. Synchronus or Asynchronus Communication
3. automatic and explicit Buffering

1. Direct or indirect Communication ⇒

Under direct communication each process that want to communicate must explicitly name the recipient or senders of the communication.

Send (P, Msg) ⇒ Send a msg to process P.
recieve (Q, msg) ⇒ recieve a msg from Process Q.

When there exist exactly one link b/w each Pair of process then this is symentric addressing that is both sender and recievier process must name the other to communicate.
There can also be asymetri in addressing where only Sender Sender name the recipient in this send and recieve are defined as —

Send (P, Msg) ⇒ Send a msg to Process P
recieve (id, Msg) ⇒ recieve msg from any process the Variable Id is set to the name of process with with Communication has taken place.

With indirect communication the message are send to and recieve to mail boxes or Ports. Process places msg into mail box which has unique

Identification

Send (A, msg) ⇒ Send a msg to mail box A.
recieve (A, msg) ⇒ recieve a msg from mail box A.

There should be a shared mail box between a pair of processes.

2. Synchronization ⇒

Communication between processes take place through calls to send and recieve premitive message passing may be either blocking or non blocking also known as synchronous and asynchronous.

Blocking Send ⇒ The sending process is block untill the message is recieve by the recieving process or by the mail box.

Non blocking send ⇒ The sending process send the msg and resume operation here the contept of send and forget is used

Blocking recieve ⇒ The recievers blocks untill a msg is available.

Non blocking recieve ⇒ The recievers retrives either a vallid msg or a null.

③ Buffering ⇒ whether communication is direct or indirect message exchanged by communication process reside in temporily queue. queue can be of three ways ⇒

1.) zero capacity ⇒ The queue has maximum length of 0. thus the link cannot hasse any message waiting in it. the sender must block until the recipient recieve the message it is a system of no buffering.

2.) Bounded capacity ⇒ The queue has finite length "n" if the queue is full the Senders must block until the space is available in a queue.

3.) Unbounded capacity ⇒ The queue length is infinite thus any number of msg can wait in it the senders never blocks.

PART - II

CPU Scheduling

In single processor system only one process can run at a time other process has to wait. The objective of multiprogramming is to have some process running at all time to maximise CPU utilization in this several process are kept in memory at a time when one process has to wait the operating system takes the CPU away from that process and gives CPU to another process. scheduling of this time is a fundamental O.S function.

CPU, I/O, Burst Cycle ⇒ Process execution consist of a cycle of CPU execution and Input, output wait process alternate between these two state process execution process begins with CPU burst and then I/O burst.

load store add store } CPU burst
read from file
wait for I/O } I/O burst
store increment } CPU burst
write to file
wait for I/O } I/O burst
load store add } CPU burst
store read from file
wait for I/O } I/O burst

Alternative CPU & and I/O burst

* CPU scheduler ⇒ when ever the CPU become ideal the operating system must select one of the process from ready queue to be exicuted the selection process is candid out by short term scheduler. there are two types of scheduling.

1. Preemptive scheduling ⇒ The scheduling is preemptive if onece a process has been given the CPU can taken away the CPU scheduling disicion can take place under four se transes t

(I.) when process switches from runing state to the writing state.

(2.) when process switches from runing state to ready state.

(3.) when process switches from waiting state to ready state.

(4.) when process terminates.

For (1) & (4) there is no choice in terms of scheduling a new process must be selected for excution there is a choice for (2) & (3). when the scheduling take place only under (1) & (4) we say scheduling is non preemptive or co-operative otherwise it is preemptive.

Anil
Date
Page

② NON preemptive scheduling ⇒ Once the CPU has been allocated to the process the process keeps the CPU untill it releases the CPU either by terminating of by switching to waiting state.

★ Dispatcher ⇒ It is a module that gives control of the CPU to the process selected by the short term schedulers. the function of dispatcher are ⇒

1 Switching context
2. Switching to user mode
3. Jumping to proper location in the user program to re-start that program.

Imp

★ Scheduling Criteria ⇒ The criteria include the following.

1 CPU utilization
2 Throughput
3 Turn around time
4 Waiting time
5 Response time

1) CPU utilization ⇒ We want to keep as busy as possible it can range from 0 to 100% in real system it should range from 40% to 90%.

2) Throughput ⇒ If CPU is busy then work is being done one

measure of work is number of process that are completed per unit time is called throughput.

3. Turn around time ⇒ The interval from time of submission of a process to the time of completion is the turn around time it is sum of the periods spend waiting to get into memory, waiting in ready queue, executing on the CPU and during I/O.

4. Waiting time ⇒ It is the sum of period waiting in ready queue.

5. Response time ⇒ Time from submission of request until the first response is produce this measure is called response time. It is desirable to maximise CPU utilization and throught put and to minimise turn around time, waiting time and response time.

★ Scheduling Algorithm ⇒

1. First come first served (FCFS) scheduling ⇒

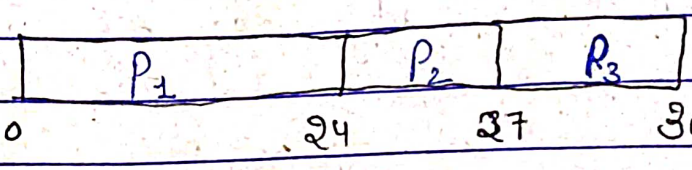
The process that request the CPU first is allocated the CPU first. the FCFS policy is managed by FIFO (first in first out) queue. when the CPU is free it is allocated to the process at the head of queue the average waiting time under the FCFS policy is quite long. the FCFS scheduling algorithm is non-preemptive once the CPU has been allocated to a process, that process keeps the CPU untill it relegious the CPU either by terminating or by requesting input output.

Anil
Date
Page

Ex ⇒

Process	Burst time
P_1	24
P_2	3
P_3	3

Grantt chart



Waiting time of $P_1 = 0$

$P_2 = 24$

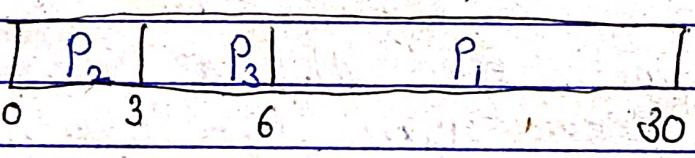
$P_3 = 27$

The average waiting time $= \frac{0 + 24 + 27}{3} = \frac{51}{3} = 17$ ms

Ex ⇒

Process	Burst Time
P_2	3
P_3	3
P_1	24

Grantt chart



Waiting time of $P_2 = 0$

$P_3 = 3$

$P_1 = 6$

The average waiting time $= \frac{0 + 3 + 6}{3} = \frac{9}{3} = 3$ ms

Anil
Date
Page

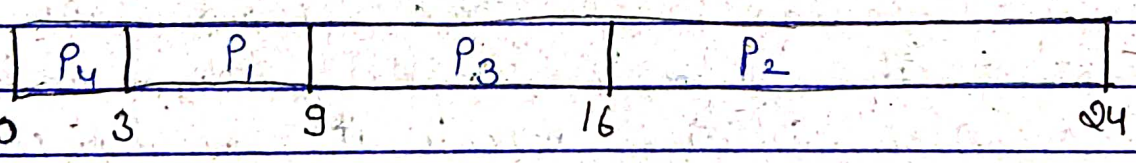
2. Shortest Job first algorithm (SJF) ⇒

Here the processes next CPU burst is seen when the CPU is available it is asigned to the process that has the smallest next CPU burst if next CPU burst is same FCFS scheduling is used. it is also called shortest next CPU burst algorithm because scheduling become on the length of the next CPU burst of a process reather then its total length the SJF algorithm is optimal (best) in that it gives the minimum average waiting time for a given set of processes

Ex ⇒

Process	Burst time
P_1	6
P_2	8
P_3	7
P_4	9.

Gantt chart.



Waiting time $P_1 = 3$

$P_2 = 16$

$P_3 = 9$

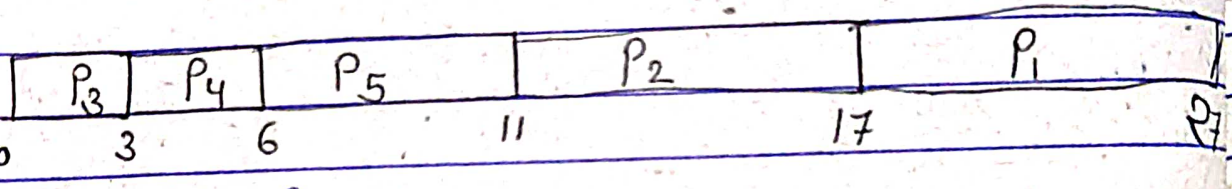
$P_4 = 0.$

The average waiting time $= \frac{3+16+9+0}{4} = \frac{28}{4} = 7$ ms

Ex ⇒

Process	Burst time
P_1	10
P_2	6
P_3	3
P_4	3
P_5	5

Gantt chart



Waiting time ⇒ $P_1 = 17$

$P_2 = 11$

$P_3 = 0$

$P_4 = 3$

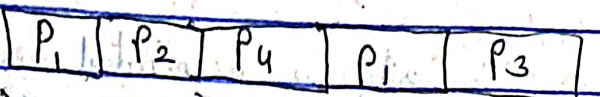
$P_5 = 6$

The average waiting time ⇒ $\frac{17+11+0+3+6}{5} = \frac{37}{5} = 7.4$ ms

The SJF algorithm can be either preemptive or non preemptive the preemptive SJF algorithm will print the currently executing process it is also called shortest remaining time first scheduling.

Ex ⇒

Process	arrival time	Burst time
P_1	0	8
P_2	1	4
P_3	2	9
P_4	3	5



WT of $P_1 = 10 - 1 - 0 = 9$

$P_2 = 1 - 1 = 0$

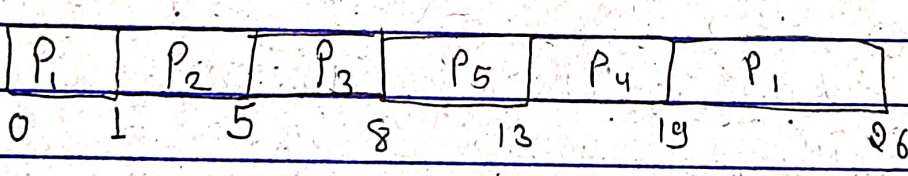
$P_3 = 17 - 2 = 15$

$P_4 = 5 - 3 = 2$

$$\text{aWT} = \frac{26}{4} = 6.5 \text{ ms}$$

Ex 2 ⇒

Process	AT	BT
P_1	0	8
P_2	1	4
P_3	2	3
P_4	3	6
P_5	4	5



WT of $P_1 = 19 - 1 - 0 = 18$

$P_2 = 1 - 1 = 0$

$P_3 = 5 - 2 = 3$

$P_4 = 13 - 3 = 10.$

$P_5 = 8 - 4 = 4$

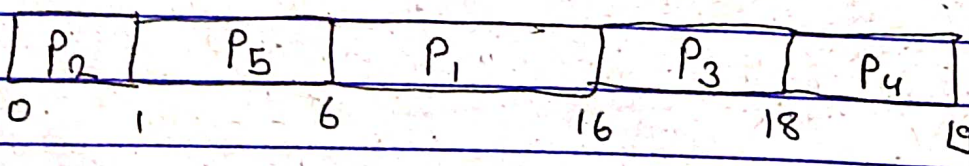
$$\frac{18 + 0 + 3 + 10 + 4}{5} =$$

Anil
Date
Page

★ Priority scheduling ⇒ A priority is associated with each process and the CPU is allocated to the process with highest priority. Equal priority processes are schedule in FCFS order. the larger the CPU Burst the lower the priority and vise verse priority indicated by a range of numbers such as 0 to 7 or 0 to 4095. the assume that low number represent highest priority. priority can be defined either internally or externally priority scheduling can be either preemptive or non preemptive. A preemptive priority scheduling algorithm will print the CPU if the priority of newly alive process is higher then the priority of currently runing process. A non-preemptive priorty scheduling algorithm will simply put the new process at the head of ready queue.

Ex ⇒

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



WT of $P_1 = 6$

$P_2 = 0$

$P_3 = 16$

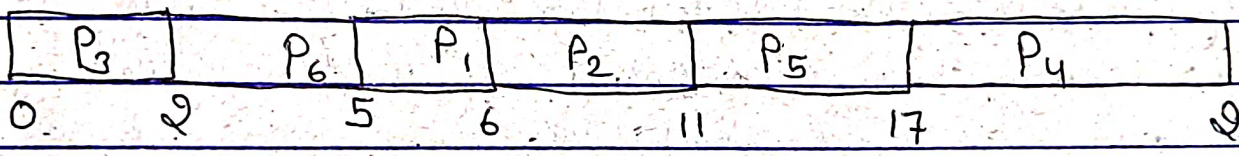
$P_4 = 18$

$P_5 = 1$

$$\frac{6+0+16+18+1}{5} = \frac{41}{5} = 8.2$$

Ex2 ⇒

Process	BT	Priority
P_1	1	3
P_2	5	4
P_3	2	1
P_4	7	6
P_5	6	5
P_6	3	2



WT of $P_1 = 5$

$P_2 = 6$

$P_3 = 0$

$P_4 = 17$

$P_5 = 11$

$P_6 = 2$

$$\frac{5+6+0+17+11+2}{6} = \frac{41}{6}$$

⇒ 6.8 Ans

A major problem with priority scheduling algorithm is indefinite blocking or starvation.

A priority scheduling algorithm can leave some low priority process waiting indefinitly.

A solution to the problem of indefinite blocking of low priority process is CPU time

Process की priority को बढ़ाता है।

priority process is aging it is a technice of gradually increasing the priority of processes that wait in the system for long time.
For example increasing the priority of process by one in every 15 mint.

★ Round robin scheduling algorithm ⇒ The round robin scheduling algorithm is designed for time sharing system, it is similler to FCFS but preemption is added to switch between process a small unit of time called a time quantum is used the average waiting time under this policy is long. the performance of round robin scheduling algorithm depends hawyly on size of time quantum. if time quantum is large the round robin scheduling policy is same as FCFS if time quantum is extreamly small the round robin approch is called processor sharing.

Ex →

Process	BT
P_1	24 ~~20 16 10~~ 8 6 0 = 4 ms.
P_2	3
P_3	3

P_1	P_2	P_3	P_1	P_1	P_1	P_1	P_1
0	4	7	10	14	18	22	26	30

WT of P_1 = 26 − 5(4) = 26 − 20 = 6

P_2 = 4

P_3 = 7

aWT ⇒ $\frac{6+4+7}{3} = \frac{17}{3} = 5.6$

Anil
Date
Page

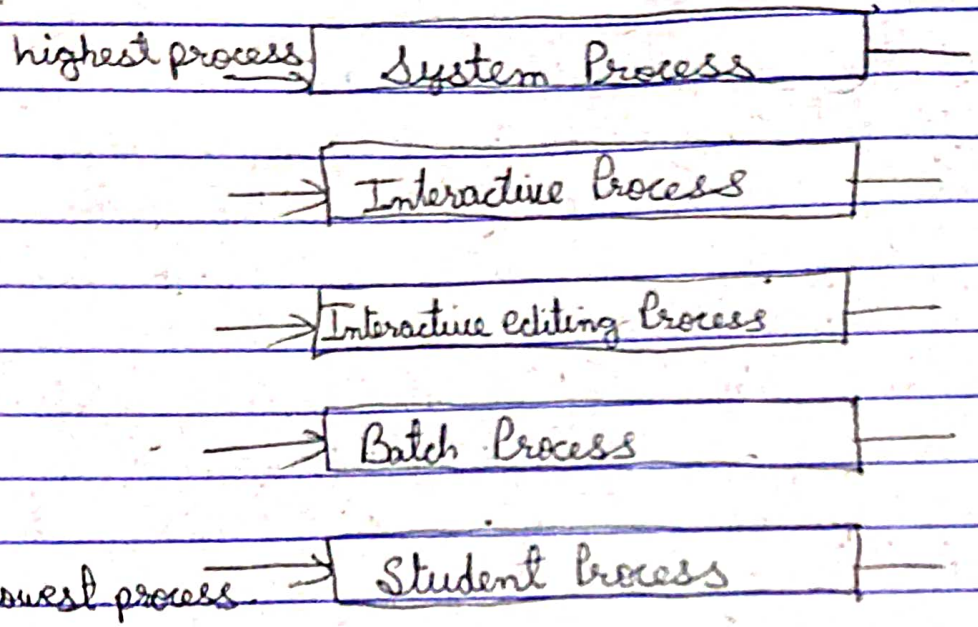
* Multi level queue scheduling ⇒ It is used for process which are classified into groups such as forground (interactive) process and background (batch) process. This two process have different response time requirement. also the priorty of process in forground is more.

In this algorithm readyqueue is partation into sapret queue and processes are assign to this queue on the bases of some property such as memory size, process priority ETC each queue has there own scheduling algorithm. For Ex ⇒ the forground queue is is scheduling by round robin while background queue is scheduled by FCFS algorithm there is also scheduling among queue which is based on priority scheduling and example of multi level queue scheduling with five queue with order of priority are!

1. System process
2. Interactive process
3. Interactive editing process
4. Batch process
5. Student process.

No process in batch queue could run unless the queue for system process, interactive process and interactive editing process were all empty if the process enter in interactive editing queue while a batch processing is runing, the batch process would be

premptive



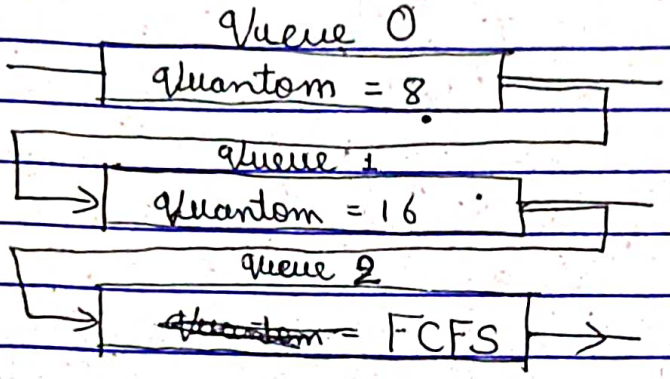
~~Ex ⇒ Process~~

★ Multilevel Feedback queue scheduling ⇒

In ML q/S process donot move on one queue to another where as in this algorithm process move b/w queues if a process uses to much of CPU time it is moved to lower priority queue. In addition - a process that waits to long in lower priority queue may be move to higher priority queue this form of aging prevent starvation

Ex ⇒ ther are 3 queue no 0 to 2, the scheduler must execute process in queue 0, when queue 0 is empty it will execute process in queue 1 and then in queue 2. A process that arrive for queue 1 will prempt process in queue 2. A process entring ready queue is put in queue 0. A process in queue 0 has

time quantom 8 m/s. if it doesn't finish with in time it is move to tale of queue 1 and so on this scheduling algorithm gives highest priority to any process with the CPU burst of 8 ms or less



* Algorithm evaluation ⇒ two Criteria used for selecting and algorithm are:

1. maximising CPU utilization
2. " ... through put such that turn around time is linnerly proportion to total execution time
3.

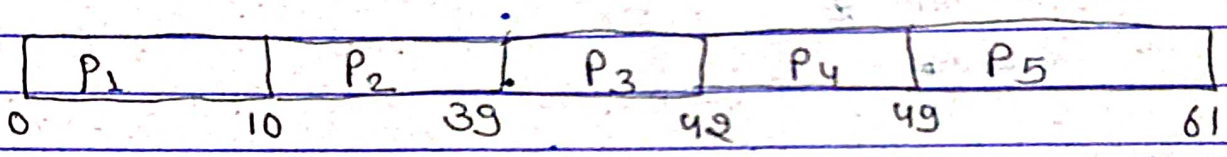
* deterministic modeling ⇒ One major class of evalution method is analytic evaluation. it uses the given algorithm and the system work load to produce the formula that evaluate the performance of algorithm One ~~time~~ type of analytic evalution is deterministic modeling this method takes a particular predetermine work load and defines the performance of each algorithm for that work-load.

Ex ⇒

Process	BT					
P_1	10	6	2	X		
P_2	29	25	21	17	13	9
P_3	3	X	X	X		
P_4	7	3	X	X		
P_5	12	8	4	X		

FCFS
5 SJF
RR

FCFS ⇒

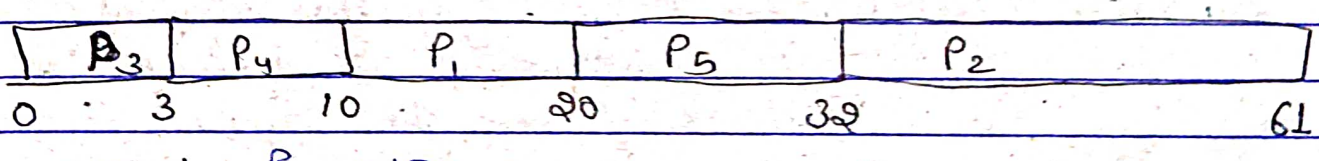


WT of $P_1 = 0$
$P_2 = 10$
$P_3 = 39$
$P_4 = 42$
$P_5 = 49$

$\frac{0+10+39+42+49}{5} \Rightarrow \frac{140}{5}$

$\Rightarrow 28$ ms

SJF ⇒

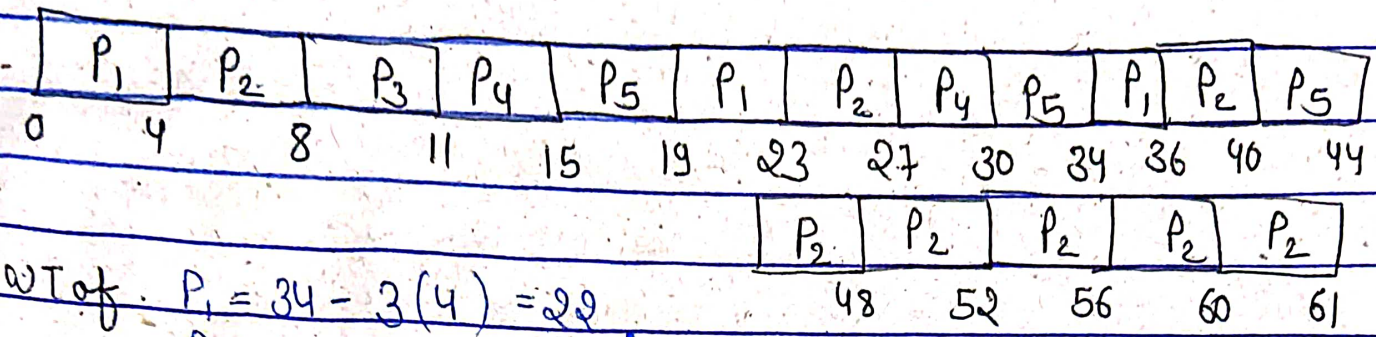


WT of $P_1 = 10$
$P_2 = 32$
$P_3 = 0$
$P_4 = 3$
$P_5 = 20$

$\frac{10+32+0+3+20}{5} = \frac{65}{5}$

$\Rightarrow 13$ ms

Round Robin ⇒



WT of. $P_1 = 34 - 3(4) = 22$

$P_2 = 60 - 7(4) = 32$

$P_3 = 8$

$P_4 = 27 - 1(4) = 23$

$P_5 = 40 - 2(4) = 32$

$= \frac{22+32+8+23+32}{5} = \frac{117}{5}$

$= 23.4$ ms

* In SJF the average waiting time is less then half that obtain in FCFS and round-robin algorithm gives us and intermiddeate value deterministic modeling is simple and fast it give us exact no. to compair the algorithm.

Deadlock ⇒ जब कोई Process किसी resource को request करता है और यदि उस time resource available नहीं है तो Process resource के लिए wait करता है, और जब तक resource नहीं मिलता तब तक Process अपना काम नहीं कर पाता है।

Unit III (Part I)

(Deadlock)

A process request resource and if resource is not available at that time the process enters waiting state some times a waiting process never again able to change the state because the resource it has requested are healed by other waiting process this situation is called a Deadlock.

★ System model ⇒ जितने Process हैं उतने ही resource दिए जाते हैं।

resources (CPU cycle, files, I/o divices, memory Space etc). must be distributed among no. of processes if a system has two CPU then resource type CPU has two instances A system with 5 Printer has 5 instances. A system have 2 Printer, this printers may be defined to be in same resource class if no one cares which printer may be printing output. However if 1 Printer is on 9th floor and other in baisement then saperate resource classes may need to be define for each printer.

A process must request a resource before using it and must release after using it. under normal mode of operation A process utilize a resource in following sequence. Kisi resource को use करने से पहले request करता है और use करने के बाद release करता है। system mode में following sequence में चला करता है।

1.) Request
2.) Use
3.) release

The request and release of resource are system call

To elastrat a deadlock state consider a system with 3 CD RW drives, if 3 processes hold one of this CD drive and how each process request another drive then 3 process will be in deadlock.

Deadlock may also envolve different resource type.
Ex → concider a system with 1 Printer and 1 DVD drive suppose that process PI is holding the DVD and process PJ is holding printer. if PI request the printer and PJ request the DVD drive then deadlock occure.

★ Deadlock characterization or Necessary Condition Of Deadlock ⇒ इसमे resource non sharable mode की help लेते है, अर्थात एक time पर केवल 1 process ही resource का use कर सकता है। उस time यदि दूसरे Process की request आती है तो resource 1 Process का काम पूरा होगा फिर दूसरे Process की तरफ जाएगा।

1.) Mutual exclusion ⇒ Atleast one resource must be help in nonsharable mode that is only one process at a time can use the resource if another process request that resource then it must be delay untill the resource has been released.

2.) Hold and wait ⇒ A process must be holding atleast one resource and waiting to get additional resource that are currently being help by other process.
इसमे Process 1 resource को hold कर सकती है या उस resource का wait कर सकती है।

Anil

Date ______
Page ______

resource ~~no~~ Preempted है क्योंकि resource तभी release होगा जब Process अपना task complete करेगा।)

3) No Preemption ⇒ resource can not be preempted that is a resource can be release only by the process after completing its task.

4) Circular wait ⇒ A set {$P_0, P_1, P_2 \dots P_{n-1}, P_n$} of waiting processes must exist such that P_0 is waiting for resource held by P_1, P_1 is waiting for resource held by P_2 so on P_{n-1} is waiting for ~~a~~ resource held by P_n and P_n is waiting for resource held by P_0.

All four condition ~~must~~ must hold for deadlock to occure.

* Resource allocation ~~graf~~ graph ⇒ deadlock ~~a~~ can be describe in terms of ~~dea~~ directed graph call system resource allocation graph.

This graph consist of set of vertices (V) and set of edges (E)

V is partition into two different types of nodes.

$$P = \{P_1, P_2 \dots P_n\}$$

that is active process.

$$R = \{R_1, R_2 \dots R_n\}$$

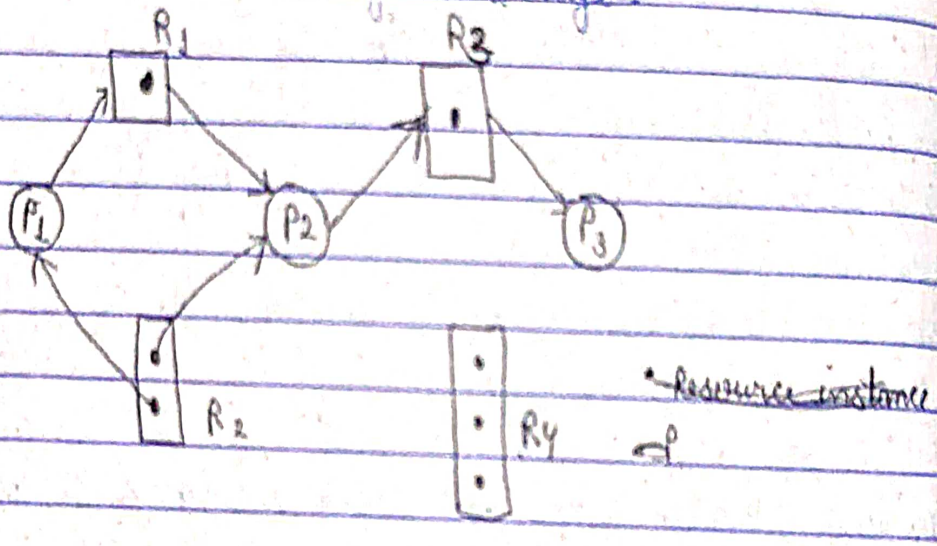
set of resources

A directed edge from process PI to resource type Rj denoted as $PI \rightarrow Rj$ says that Pi has requested and instance of resource type Rj and it is waiting for that resource.

An edge from R_j to P_i that is $R_j \rightarrow P_i$ indicates that an instance of resource type R_j has been allocated to the process P_i

$P_i \rightarrow R_j$ is called request edge.

$R_j \rightarrow P_i$ is called assignment edge.



RAG

Process is represented by circle and each resource is represented by rectangle. (•) inside rectangle represent instance of that resource.

The resource allocation graph shows →

⊙ The set P, R and E

$P \Rightarrow \{P_1, P_2, P_3\}$

$R \Rightarrow \{R_1, R_2, R_3, R_4\}$

$E \Rightarrow \{P_1 \rightarrow R_1, R_1 \rightarrow P_2, R_2 \rightarrow R_3, R_3 \rightarrow P_3, R_2 \rightarrow P_1, R_2 \rightarrow P_3\}$

is $R_j \rightarrow P_i$ it says that
pe R_j has been
P_i

edge
edge

P_3

• Resource instance

each resource is
rectange represent

P_3, $R_2 \rightarrow P_1$, $R_2 \rightarrow$

⊙ Resource instance ⇒ R_1 has one instance
R_2 has two instance
R_3 has one instance
R_4

⊙ Process instance ⇒ ① P_1 is holding an instance of R_2 and waiting for instance R_1.

② P_2 is holding an instance of R_2 and waiting for instance R_3.

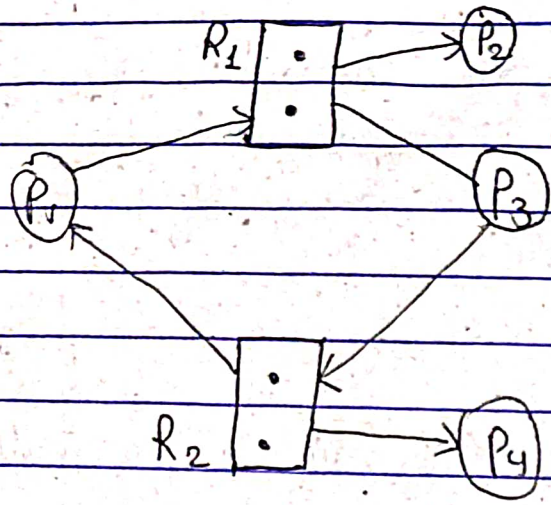
③ P_3 is holding an instance of R_3.

if a graph has no cycle then no process in the system is in deadlock.
if the graph has cycle then deadlock may exist.

if each resource type has exactly one instance then cycle implies that deadlock has occure.

if each resource type have several instance then a cycle does not neccesarylly imply that deadlock has occure.

In the earlier figure if Process P_3 request of instance of resource type R_2 since no resource instance is currently unavailable a request edge $P_3 \rightarrow R_2$ is added to the graph at this point two cycle exist in the system and the process P_1 P_2 & P_3 are on deadlock.



Resource allocation graph with a cycle but no deadlock

★ Method for Handling Deadlock ⇒

We deal with deadlock in one of 3 ways.

1.) We can use protocol to prevent or avoid deadlock.

2.) We can allow system to enter the deadlock state, detected it and recover it

3.) We can ignore the problem and pretend that deadlock never occur in the system.

★ Deadlock prevention ⇒ It provide a set of method for ensuring that atleast one of necessary condition cannot hold

★ Deadlock Avoidance ⇒ It requires that O.S. be given advanced additional information concerning which resource a process will request and used during its lifetime.

* Deadlock Prevention ⇒ If atleast one of the necessary condition cannot hold we can prevent deadlock.

1. Mutual exclusion ⇒ this condition must hold for non shareable resource.
Ex⇒ A printer cannot shared by many process sharable resource don't require mutual exclusion and thus are not involved in deadlock read only files are a good example of shareable resources. A process never need to wait for a shareable resource.

2. Hold and wait ⇒ To ensure that this condition never occurs in the system we must guarantee that whenever a process request a resource it doesn't hold any other resource One protocal required that each process request and get allocated all its resource before it begin execution another protocal allow a process to request resource only when it has null.
To show the difference between these two protocal we consider a process that copies data from DVD drive to a file, sort the file & print the result if all resources are requested at the starting then it will hold printer for its entire execution even though printer is needed at the end.
By the second protocal the process request initially only DVD drive and disk file it copies and then release DVD drive. Now the process request printer and after printing it releases both file & printer.
Both these protocal have two disadvantage.

1. resource utilization is low
2. starvation is possible.

③ No Premption ⇒ To ensure that this condition does not hold we can use the following

Anil
Date
Page

Protocol

① if a process is holding some resource and request some resource they are available if they are available we allocate them if not we check weather they are allocated to some other process that is waiting for additional resource if so we premp the desired resource from the waiting process and allocate them to the requesting process this protocal is applied to resource whose state can be easily saved and restore such as CPU register and memory space it can not be applied to printer and tape drive.

④ Circular wait ⇒ to insure that this condition never hold is to impose a total ordering of all resource type and to require that each process request resource in encreasing order of enumeration.

Let $R = \{R_1, R_2, R_3, \ldots, R_m\}$

Be a set of resource type every resource is asign a unique integer. there is one to one function $F : R \to N$ where N is a natural number.

Ex ⇒ F(take Drive) = 1
F(Disk drive) = 5
F(Printer) = 12

we can use the following protocal to prevent protocal

1.) Each process can request resource only in increasing order of enumeration that is a process can initialy request any number of resource type say Ri after that the process can request instance of resource Rj if $F(R_j) > F(R_i)$

2) If several instance of the same resource type are needed then a single request for all of them must be issue.

Ex ⇒ A process that want to use tape drive and printer at the same time it must first request the tape drive and then request the printer alternatively we can require them when ever a process request an instance of resource type Rj it has release any resource Ri such that $F(R_i) \geq F(R_j)$ if this two protocal are used then circular wait condition can not hold.

Dead lock avoidance ⇒ if editional information about how resources are to requested is known then deadlock can be avoided.

Ex ⇒ A system will one tape drive and one printer and the process P will request first tape drive and then printer befor releasing both resources where as process queue will first request printer and then drive, with this knowledge of system the system can decide for each request weather or not process should wait to

avoid deadlock.

The simplest and most useful model require that each process declare maximum numbers of resource of each type that it may need with this information it is easy to construct and algorithm such a algorithem defines deadlock avoidance approach.

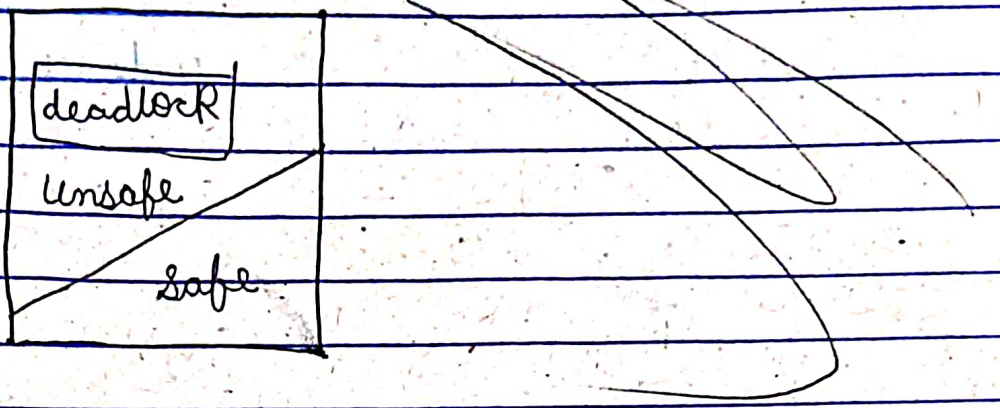
Three deadlock avoidance algorithm are :-

1 safe state
2 Resource allocation graph algorithm
3 Banker's algorithm.

(1) safe state ⇒ A state is safe if the system can allocate resource to each process in some order and avoide deadlock. A system is in safe state only if there is safe sequence.

A The sequence of the process $< P_1, P_2, P_3 \ldots P_n >$ is a safe sequence for current allocation state if each Pi the resource request that Pi can still make can be satisfyed by the currently available resource + the resource held by all Pj with $j < i$. If the resources that Pi needs are not available then Pi can wait until all Pj have finished. if no such sequence exist then the system state is set to be unsafe.

A safe state is not a deadlock state a deadlock state is unsafe state not all unsafe state are deadlock.



safe, unsafe and deadlock state spaces.

Example ⇒ Consider a system with maximum needs and current needs of magnetic tap drive for 3 process P_0, P_1, P_2.

	max need	current need
P_0	10	5
P_1	4	2
P_2	9	2

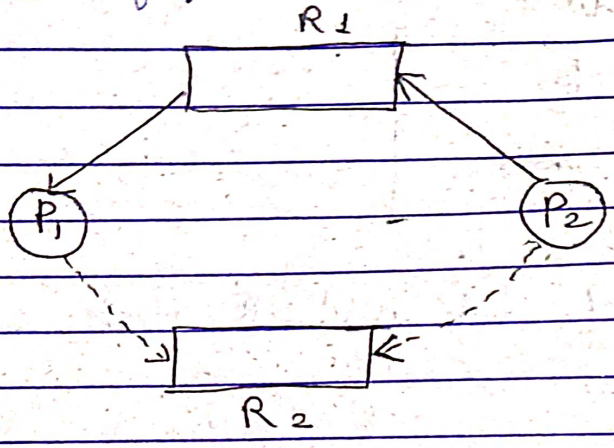
Total magnetic tape drive are ⇒ 12

At time t_0 to process P_0 is holding 5 tape drive, t_1 holding 2 and t_2 is holding 2 at this time system is in safe state. at time t_1 process P_2 request and is allocated one more tape drive then the system is no longer in the safe state the request should be granted only if the allocation leaves the system in the safe state.

(2) Resource allocation graph algorithm ⇒

If we have resource allocation system with only one instance of each resource then a varient of RAG

Can be used for deadlock avoidance :-



In addition to request and assignment edge a new type of edge call claim edge is used. A claim edge $P_i \rightarrow R_j$ indicate that process P_i may request resource R_j at some time in future. it is represented by dashed line & when process P_i request resource R_j the claim edge P_i to R_j is converted to request edge. Similarly when R_j is released by P_i the assignement edge is reconverted to claim edge. Suppose P_i request resource R_j the request can be granted only if converting request edge to assignment edge does not result in formation of a cycle in RAG. if no cycle exist then the allocation of the resources will leave the system in safe state if a cycle is found then allocation will put the system in unsafe state.

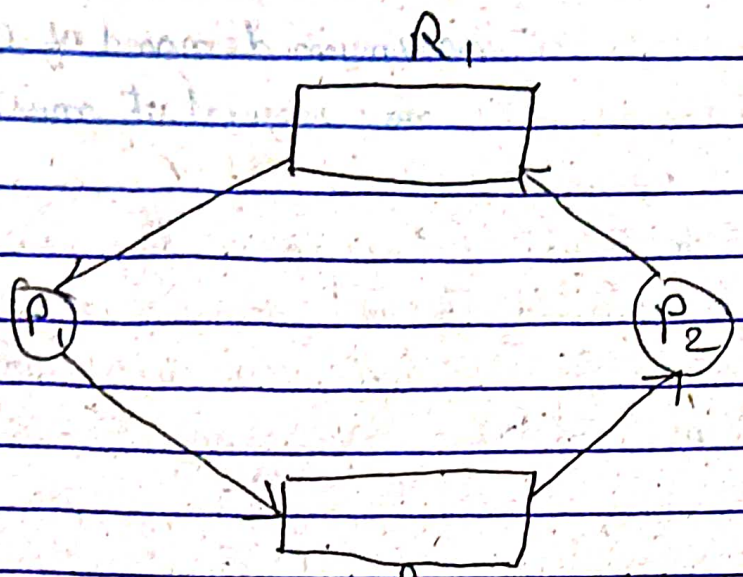


fig a unsafe state in RAG

Ex ⇒ Suppose P_2 request R_2, also R_2 is currently free we can not allocate it to P_2 since this action will create a cycle in the graph. A cycle indicate unsafe state

③ Banker's algorithm ⇒ Banker's algorithm applicable for the system who have multiple instances of each resource type it is called Banker's algorithm because the algorithm could be used in Banking system to ensure that bank never allocate its available cash in such a way that it could not longer satisfy the needs of all its customers.

When a new process enters a system it must declare maximum no. of instances of each resource type that it may need. number should not exist total number of resources several data structure are maintained to implement Banker's algorithm.

Let n = no of process

m = no of Resource type.

1) Available ⇒ A vector of length m indicates the no. of available resource of each type if available [j] = K where K are instances of resource type R_j available.

② MAX ⇒ A n×m matrix defines the maximum demand of each proc
If max. ij = K then process Pi may request at most K inst
of resource type Rj.

③ Allocation ⇒ A n×m matrix defines the number of resource of ea
type currently allocated to each process If allocation ij
then process Pi currently allocated with K instance of resource type Rj

④ Need ⇒ A n×m matrix indicates the remaining ~~resource~~
resource need of each process if need[i][j]
= K then process ~~= K then~~ Pi may need K more instan
of resource type Rj to complete its task.
note that need[i][j] = max[i][j] - allocation[i][j]

There are 2 types of Banker's algorithm

1 Safty algorithm
2 resource request algorithm

1. Safty algorithm ⇒ it is an alg. to find out
wether or not a system is in
safe state.

Step 1 ⇒ let work and finish be vector of length
m and n.
initialize work = available and finish[i] = False
for i = 0, 1, 2 ... n-1

Step 2 ⇒ find an i such that both
(a) finish[i] == false
(b) Need ≤ work
if no such i exist go to step 4

step 3 ⇒ Work = work + allocation
finish [i] = true
go to step 2.

step 4 ⇒ If finish [i] == true for all i, then system is in a safe state.

(2) Resource request algorithm ⇒ it is an algorithm which determine if request can be safely granted.

Let request i be request vector for process Pi. if request i [j] == K then process Pi wants K instance of resource type Rj when a request for resource is made by process Pi the following actions are taken.

1. If request ≤ need goto step 2 otherwise raise error condition since process has exceeded its maximum claim.

2. If request ≤ available goto step 3. otherwise Pi must wait since resource are not available.

3. Have the system pretend to have allocated the requested resource to process Pi by modifing the state as follows.

Available = available − request

allocation = available + request

Need = need - request

Ex ⇒ Consider a system with 5 process P_0 - P_4 and 3 resource type A, B, C. Resource type A has A=10 instence, Resource type B has B=5 instence, Resource type C has C=7 instence.

	allocation A B C	max. A B C	Available A B C	Need (max - allocation) A B C
P_0	0 1 0	7 5 3	3 3 2	7 4 3
P_1	2 0 0	3 2 2		1 2 2
P_2	3 0 2	9 0 2		6 0 0
P_3	2 1 1	1 2 2		1 1 1
P_4	0 0 2	4 3 3		4 3 1

we claim that system is currently in safe state. The sequence P_1, P_3, P_4, P_2, P_0 is safe sequence.

Suppose new process P_1 request 1 instence of resource type A and 2 instence of resource type C. So request = (1, 0, 2) to decide that request can be granted we check that request is less than equal to R ≤ Available, P_1 req (1, 0, 2) ≤ (3, 3, 2) which istence now the new state is:

	allocation	Max	Avai.	Need (Max all.)
P_0	0 1 0	7 5 3	2 3 0	7 4 3
P_1				
P_2				
P_3				
P_4				

Now we must determine whether this new state is safe by safety algorithm we find the sequence P_1, P_3, P_4, P_0, P_2 satisfy the safety requirement hence we can grant the request of process P_1

✱ D/L detection ⇒

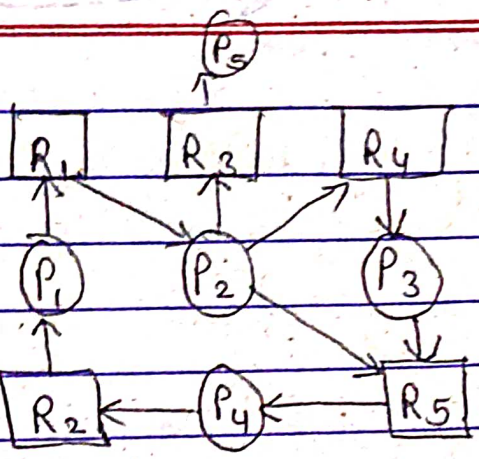
If a system does not use d/c prevention or a d/L avoidable on a by that the d/c situation may occur then system must provide.

1. An algorithm that examine the state of the system to determine if d/L has occurs.
2. An algorithm to recovers from d/L.

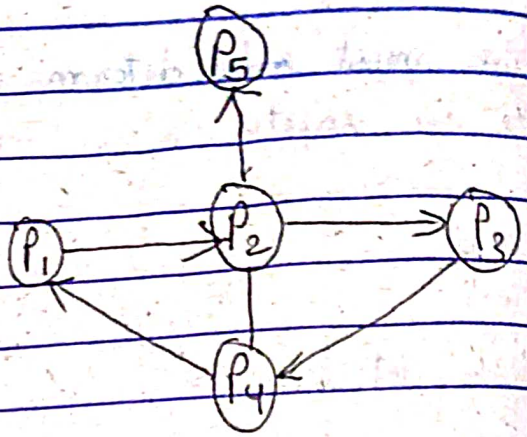
→ Single instance of each resource type ⇒

Here a Varient of resource allocation graph is made called wait for graph. It is attained from resource allocation graph by remaing resource nodes and call apring the edge. An edge P_i to P_j in a wait for graph implize that process P_i is waiting for process P_j to release resource that P_i need P_i to P_j exist only if there in two edge. P_i to R_q, $P_i \rightarrow R_q$ and $R_q \rightarrow P_j$ for some resource R_q.

A d/L exist in the system. if an only if the wait for graph contains cycle.



(RAG)



wait for graph

An algorithm to detect a cycle in a graph require an order of n^2 operation, where n is the no. of various in the group

→ Several instance of a resource type

wait for graph is applicable for single instance of known. we see an algorithm that in applicable for multiple instance the algorithm has many time varying data structure as used in Bankars algorithm.

Available ⇒ A vector of length m indicate the no of available resource of each type.

Allocation ⇒ An n X m matrix defines the no of resource of each type currently allocated to each process.

Request ⇒ An n x m matrix indicates the current request of each process. If Req[i][j] = K then process Pi is requesting K more instance of resource type Rj

① Network and finish be vectors of length m and n initialize work = Available for i = 0, 1, ... n-1 if allocation, then finish [i] = false, otherwise finish [i] = true.

② Find an india i such that both
a finish [i] == false
b request ≤ work

③ work = work + allocation
finish [i] = true
go to step 2.

④ If finish [i] = false for some i, 0 ≤ i < m then this system is in a deadlocked state. more over if finish [i] == false then process Pi is d/L.

→ Detection algorithm useage.

when should we call detection algorithm depends on 2 factors
① How often is d/L likely to occur.
② How many processor will be affered by d/L when it happens.
If d/L occurs frequently, then the detection algorithm be called frequently resource allocated to deadlock process will be idle outel d/L can be broken.
d/L occurs only when one process makes a requested that can not be granted immediately then in wake d/L detection algorithm for the request, which can't be granted

immediatly.

if the d/L detection algorithm is invoked for every resource request this will cause overhand.

is to invoke the algorithm at less frequent intervals.

Ex ⇒ Once perhours or whenever the CPU utilization looks below 40%.

★ Recovery from Deadlock.

When a detection algorithm determines that d/L exist many methode are available.

① Inform the operator that a d/L has occured and let him deal it with manually.

② let the system recover form d/L automatically there are 2 option for breaking d/L.

→ Simply apart one or more processes to break the circular wait.

→ Preempt some resource from 1 or more of the d/L process.

★ Process termination ⇒ To eliminate d/L by aborting a process we use one of the 2 method.

(a) Abort all d/L process
→ this method break d/L cycle but at great expense. the d/L process may have computed too long an now we have to recompute at latters.

(b) Abort 1 process at a time until the d/L cycle is eliminated.
→ This method add overhead as after each process is aparted, a d/L detection algorithm must be invoked to determine any process r in d/L.

Abortinges process is not easy because if the process is in middle of file updating termination will keep the file in incorrect state. OS must select the process which should be terminated some factors which effect the process to be closure ie :-

① what the priority of the process
② How long the process there computed
③ How many and what type of resource the process has used
④ How many more resource the process need in order to compeletive

Anil
Date
Page

✱ Resource preemption

By using this method are preempt some resource from processes and give this resource to other process until the d/L cycle is broken. If preemption is required to deal with d/L - than 3 issue need to be addressed.

① Selecting a victim

Which resource and which process are to be prempted. process in such a way to minimize cost. cost include parameters as no. of resources a d/L process is holding and amount of time the process has executed.

② Roll back

If we prempt resource from a process then it can not continue with its normal execution so its should be roll back to some safes state as its difficult to find safe state, simplest thing is total roll back that is restart again.

③ starvation

Starvation should not occur not all the time Some process is prompted with its resources.

Jina

Anil

Date
Page

UNIT - III (Part I)

Process synchronization

It is a machanism to ensure systematic sharing of resource among concurrent process. A Co-operation P is I that can effect or get effected by other process excieting in the system. Here we will discuss various machanism to ensure the orderly execution of Co-operating process so that data consistency is mentain.

(जब Process excute होता है तो critical section में होता है।)

★ Critical section Problem

Consider a system having and Process $\{P_0, P_1, P_2, \cdots P_{n-1}\}$ each process has a segment code called critical section in which process may be changing common variable updating a table writing a file and so on no two process can excieute in there critical section at the same time the critical section problem is to design a protocol that process use to co-operate. each process request permission to enter its critical section the section of code implimenting this request is the entry section the critical section is followed by an exit section. the remaining code is in remainder section the general structure of a tipical process is

```
{
    entry section
    Critical section
    exit section
    remainder section
while (true)
}
```

The solution to critical section problem must satisfy the following three requirement.

① Mutual exclusion ⇒ If process P_i is executing in its critical section, then no other process can be executing in there section.

② Progress ⇒ If no process in its critical section and some process wish to enter critical section then only those process that are not executing in there reminder section can participant on which will enter its critical section next.

③ Bounded waiting ⇒ there exist a bound or limit on the no. of times that other processes are allowed to enter there critical section after a process has made a request to enter its critical Section and before that request is granted.

* Semaphores ⇒ it is a synchronization tool a semaphore S is an integer variable that apart from initialization is exist only throw two standard atomic operation wait and signal.

The defination of wait is as follow -

```
wait (S)
{
while S<=0
    ; // no -operation
```

```
    S--
}
```

The defination of signal is as follow

```
signal (S)
{
    S++;
}
```

When one process modify a semaphore value, no other process can semultaniously modify that same semaphore value.

Usage ⇒ There are two types of semaphore ⇒ 1) Counting semaphore
2) Binary semaphore

The value of counting semaphores can range over Unrecepted domain. The value of binary semaphore can range only between 0 and 1. The binary semaphore is also called muter lock as they are locks that provide mutual exclusion. Binary - semaphore can dup with critical section problem for multiple processes It is initialised to 1 each process Pi is organised as

```
do
{
    Waiting (Muten);
        // CS
    Signal (Muten);
        // Remainder section
while (true)
}
```

Counting semaphore can be used to control access to a given resource consisting of a finite no. of instances the semaphore is initilize to the no. of resource available each process that wishes to use a resource perform wait operation there by decrimenting the count when a process releases a resource it perform a signal operation there by incrementing the count when count semaphore is 0 then all the resources are being used.

★ Implementation ⇒

The main disadvantage of semaphore is that it requires Busy waiting. when a process is in a critical section any other process that tryies to enter its critical section must loop continously in the entry code, this looping is a problem where single CPU is share among many processes busy waiting west CPU cycles. this type of semaphore is also called spinlock because the process spins while waiting for lock.

To over come need of busy waiting we modify wait and signal operation when wait is excuted and find that semaphore value is -ve it wait or process block itself that is put into waiting queue and state is change to waiting state the process that is block restart when other process exicute a signal operation. the process is restarted

by WAKE UP ( ) a operation which change the process from waiting state to ready state that is process is cape in ready queue.

Now the semaphore defined as –

```
type def struct {
        int value;
        struct process * list;
    } semaphore;
```

When a process wait on semaphore it is added to list of processes a signal operation removes one process from the list of waiting process and awakens that process.

the wait semaphore operation can now be define as -

```
wait (semaphore * S)
{
  S -> value --
  if (S -> value < 0)
  {
    add this process to S -> list;
    block ();
  }
}
Signal (Semaphore * S)
{
  S -> value ++
  if (S -> value <= 0)
```

Anil
Date
Page

```
{
    remove a process P from S->list;
    WAKE UP(P);
  }
}
```

1. Give examples of hand held systems ? Symbian, Palm, Linux, Pocket PC, windows, and
2. what is a system call ?
3. Write five state of the processes. New, Ready, Running, waiting, terminate
4. what is CPU scheduling ?
5. What is Critical section ?
6. what is safe state ?
7. what is segmentation ?
8. what is demand paying ?
9. what is the need of an OS ?
10. What do you understand by virtual machines ?
11. Defination of Process ? A program under execution is called Process.
12. what is a deadlock ?
13. what the meaning of swapping ?
14. what is Thrasing ?
15. what is an OS ?
16. write the types of OS ? Batch, time sharing, Distributed, Network, Real time
17. Define process life cycle ?
18. what do you mean by process control block ?
19. what is priority scheduling ?
20. what is semaphore ?
21. ~~what is deadlock~~ ? Define scheduling criteria ?
22. what is demand Paging ?
23. what do you mean by memory allocation ?

(24) Define multiprocessor system?
(25) what do you mean by scheduling?
(26) Define distributed system?
(27) write about protection system?
(28) write about process state?
(29) Define preemptive scheduling?

~~system call~~ ⇒

Critical section ⇒ Critical section allows one process to enter and modify the shared variable.

Safe state ⇒ Safe state is when there is no chance of deadlock occuring and unsafe state doesn't mean a deadlock has occurred yet, but means that a deadlock could happen.

Segmentation ⇒ the memory is divided into the variable size parts. each part is known as a segment.

demand paying ⇒ The OS copies a disk page into physical memory only if an attempt is made to access it and that page is not already in memory

Virtual machine ⇒ It is a computer resource that uses software instead of a physical computer to run programs and deploy apps.

Deadlock ⇒