Operating System os notes CS 2nd year

B.tech (Dr. A.P.J. Abdul Kalam Technical University)

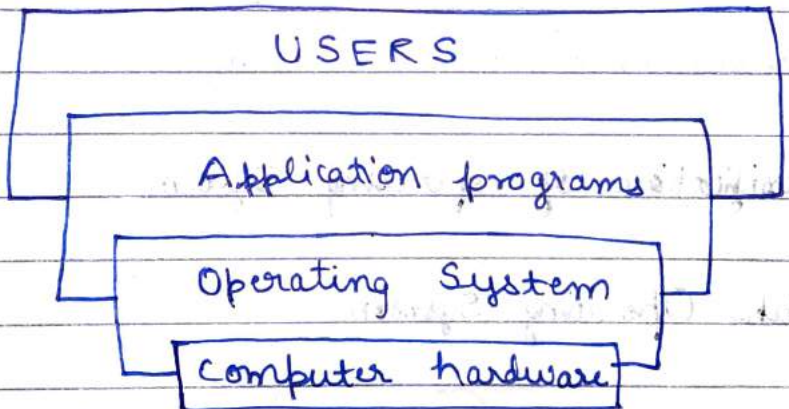
Operating System

An operating system is a program that acts as an intermediary between a user of a computer and the computer hardware. An operating System is software that manages the computers hardware.

Purpose: The purpose of an operating system is to provide an environment in which a user can execute programs in a convenient and efficient manner.

Goals of Operating System

1) Execute user programs and making solving user problems easier.
2) Make the computer system convenient to use.
3) Use the computer hardware in an efficient manner.



Logical computer hierarchy

Function of an Operating System:

1. Booting
   - Copies BIOS programs from ROM chips to main memory.
   - Loads operating system into computer's main memory.

2. formatting

formats diskettes so they can store data and programs.

3. Managing computer resources

- Keeps track of locations in main memory where programs and data are stored.
- Moves data and programs back & forth between main memory and secondary storage via partitioning.

4. Managing files:

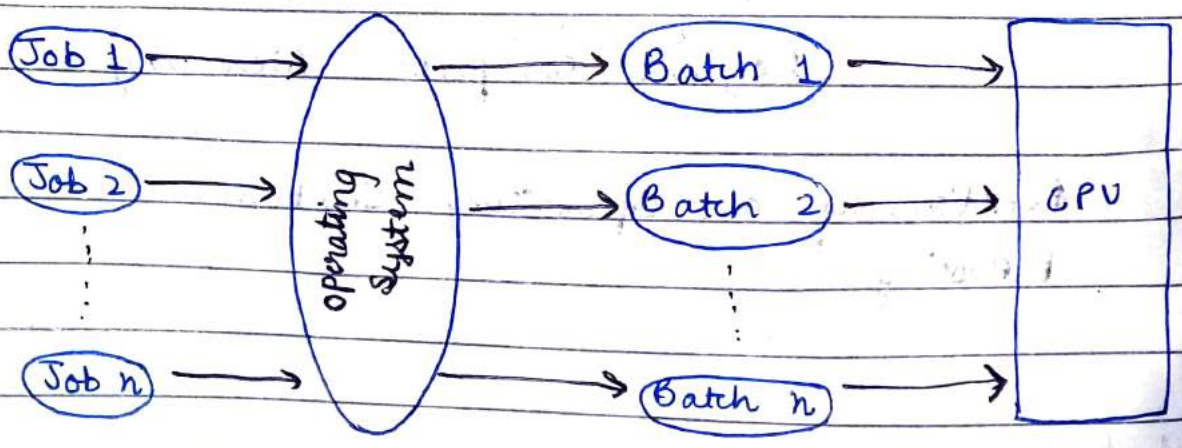
- Copies programs from one disk to another.
- Backup programs.
- Erases programs
- Rename files.

5. Managing tasks

May be able to perform multi-tasking, multi-programming time-sharing or multi-processing.

Classification of operating System

1) Batch Operating System

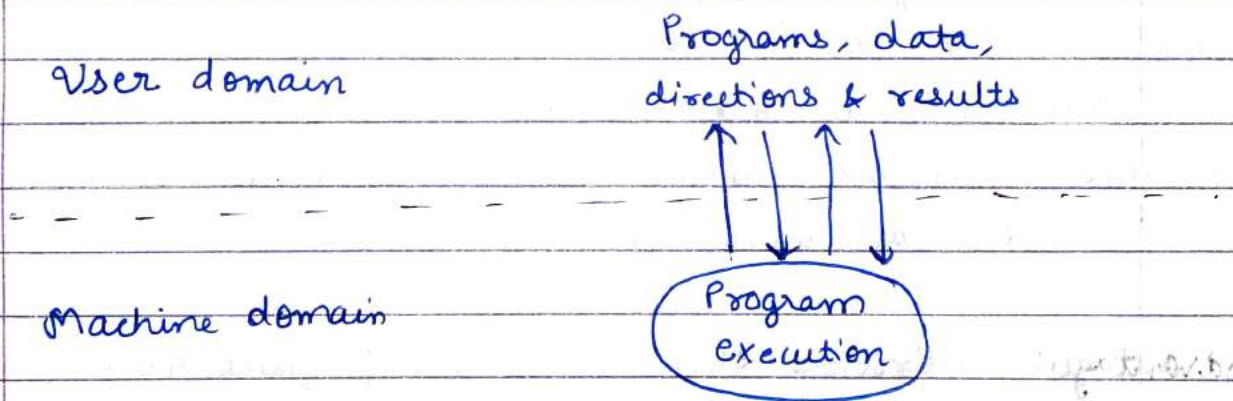


The users of a batch operating system do not interact with the computer directly. Each user prepares his job on an offline device like punch cards and submit it to the computer operator. To speed up processing, jobs with similar needs are batched together and run as a group.

Problems with Batch Systems:

- Lack of interaction between user and the job.
- CPU is often idle.
- Difficult to provide the desired priority.

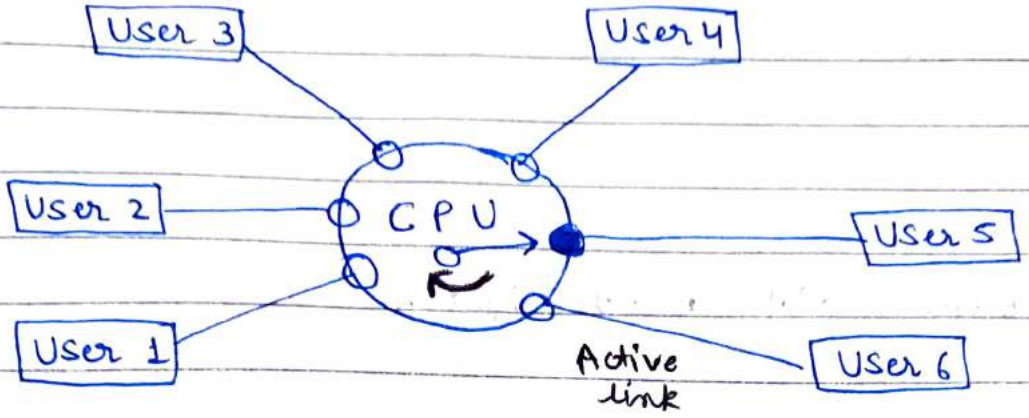
2) Interactive operating System



An OS that allows users to run interactive programs. Accepting input from a human. Interactive computer systems are programs that allow users to enter data or commands. Most popular programs, such as word processor & spreadsheet are interactive. A non interactive program is one that, when started continues without requiring human contact. Pretty much all operating systems that are on PCs are interactive OS's.

Date

3) Time-sharing operating System



Time-sharing is a technique which enables many people, located at various terminals, to use a particular computer system at the same time. Processor's time which is shared among multiple user simultaneously is termed as time-sharing.

Multiple jobs are executed by the CPU by switching between them, but the switches occur so frequently. Thus, the user can receive an immediate response.

Advantages: 1) Provide the advantage of quick response
2) Avoid duplication of software
3) Reduces CPU idle time.

Disadvantages: 1) Problem of reliability
2) Problem of data communication
3) Question of security and integrity of user programs and data.

4) Real-time Operating System

Real time system means that the system is subjected to real time, i.e. response should be guaranteed within a specified timing constraint or system should meet the specified deadline. Ex - flight control system.

Types of real time system based on timing constraints:

a) Hard real time system:

This type of system can never miss its deadline. Missing the deadline may have disastrous consequences. The usefulness of result produced by a hard real time system decreases abruptly and may become negative if tardiness increases. Ex - Flight controller System.

✓ Tardiness means how late a real time system completes its task with respect to its deadline.

b) Soft real time System:

✓ This type of system can miss its deadline occassionally with some acceptably low probability. Missing the deadline have no disastrous consequences. The usefulness of result produced by a soft real time system decreases gradually with increases in tardiness.

Ex - Telephone switches.

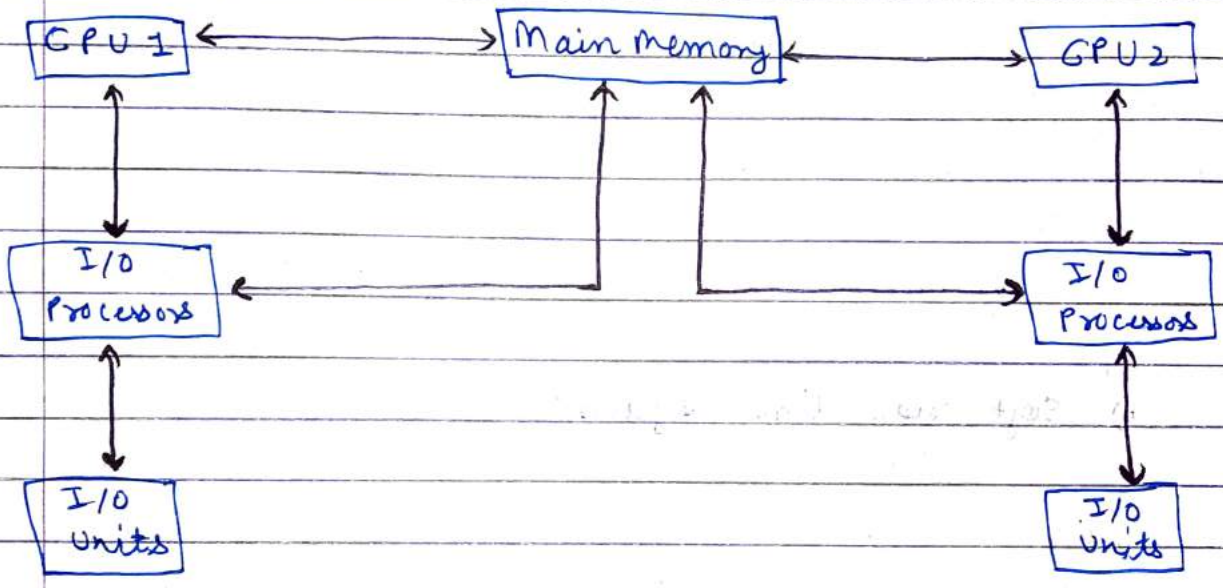
Imp: The main difference between multiprogrammed Batch Systems and time-sharing systems is that in case of multiprogrammed batch systems, the objective is to maximize processor use, whereas in time-sharing system, the objective is to minimize the response time.

5) Multiprocessor operating System

Multiprocessor systems have more than one processor in close communication. They share the computer bus, memory and other peripheral devices. These systems are referred as tightly coupled systems.

These types of systems are used when very high speed is required to process a large volume of data.

Ex - Satellite control, weather forecasting.



Multiprocessor Systems are of two types:

1) Symmetric multi-processing

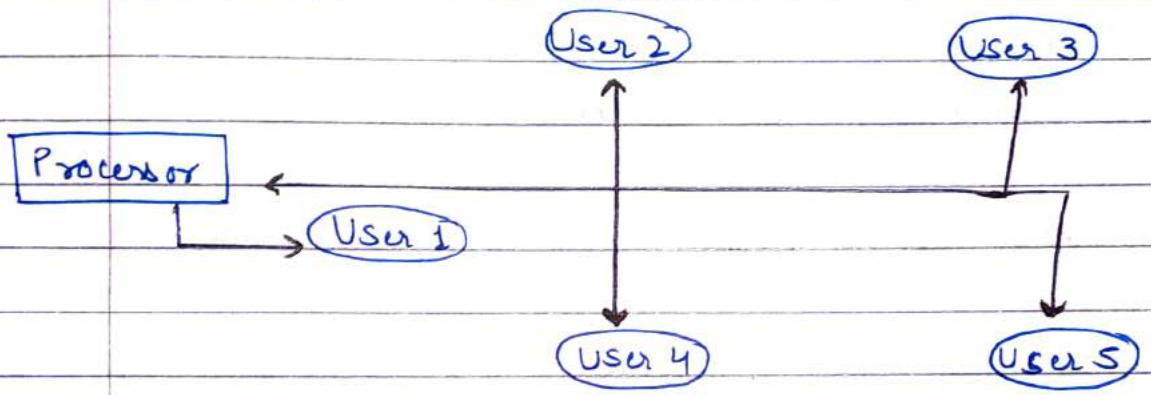
In this, each processor runs identical copy of the operating system and they communicate with one another as needed. All the CPU shared the common memory.

2) Asymmetric multi-processing

In this, each processor is assigned a specific task. It uses master-slave relationship. Processors need not to communicate as they are controlled by the master processors.

6) Multi-User Operating System

A multi-user OS is a computer operating system which allows multiple users to access the single system with one operating system on it. It is generally used on large mainframe computers. In this OS, different users connected at different terminals and we can access these users through network.



Using multi-user OS, we can perform multiple tasks at a time & we can share different peripherals like printers, hard drives or we can share a file or data.

Three types of multi-user operating system

1. Distributed Systems:

In this, different computers are managed in such a way so that they can appear as a single computer. So, a sort of network is formed through which they can communicate with each other.

2. Time-Sliced Systems:

In this, a short period is assigned to each task. As we know these time slices are tiny, so it appears to the users that they all are using the mainframe computer at the same time.

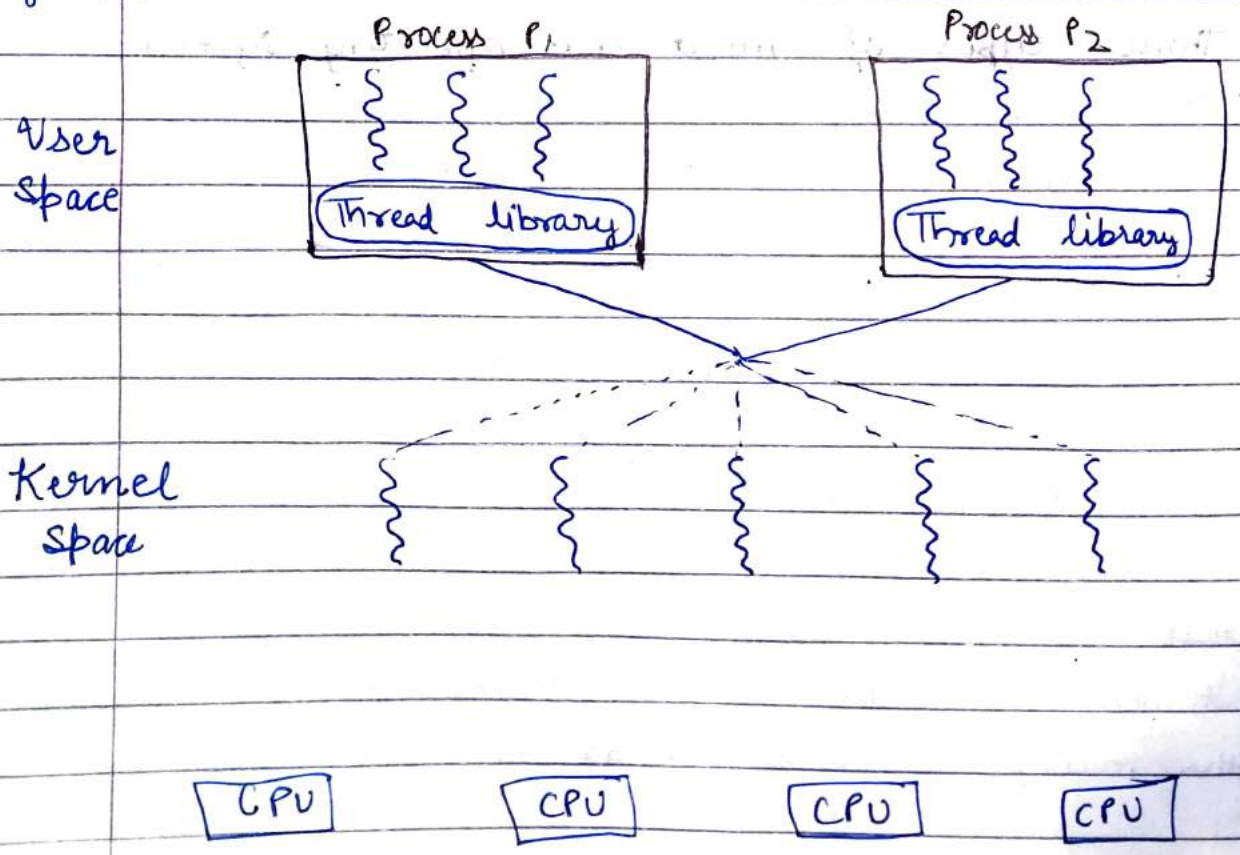
3- Multiprocessor Systems:

In this, the OS utilises more than one processor. Ex- Linux, UNIX, windowsxp.

7) Multithreading Operating System

Multi-threading extends the idea of multi-tasking into applications, so we can sub-divide specific operations within a single application into individual threads. Each of threads can run in parallel. The OS divides processing time not only among different applications but also among each thread within an application.

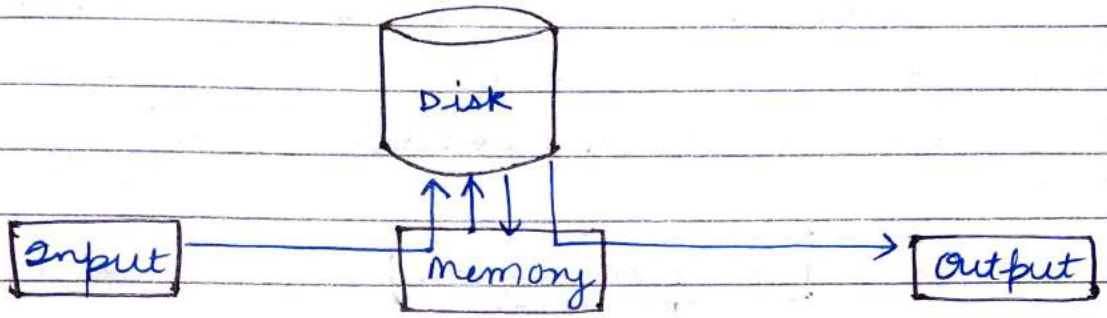
In a multi-threaded program, an example application might be divided into four threads: a user interface thread, a data acquisition thread, network communication and a logging thread. We can prioritize each of these, so that they operate independently. Thus, in multi-threaded applications, multiple tasks can progress in parallel with other applications that are running on the system.



Page No.
Date

Spooling

is acronym for Simultaneous Peripheral Operations Online. Spooling refers to putting jobs in a buffer, a special area in memory or on a disk where a device can access them when it is ready.



Spooling is useful because device access data at different rates. The buffer provides a waiting station where data can rest while the slower device catches up. The most common spooling application is print spooling.

In print spooling, documents are loaded into a buffer and then the printer pulls them off the buffer at its own rate. Spooling is also used for processing data at remote sites. The CPU sends the data via communications path to a remote printer.

Advantages:

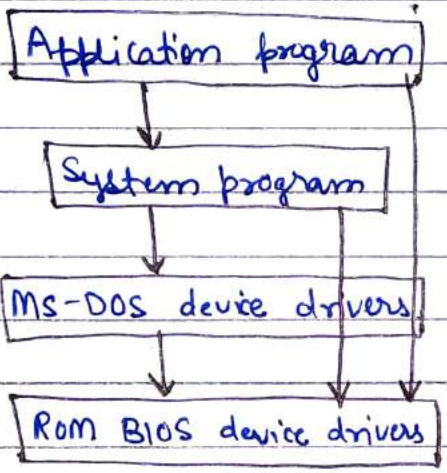
1) The spooling operation uses a disk as a very large buffer
2) Spooling is capable of overlapping I/O operation for one job with processor operations for another job.

Page No.
Date

Structure of Operating System

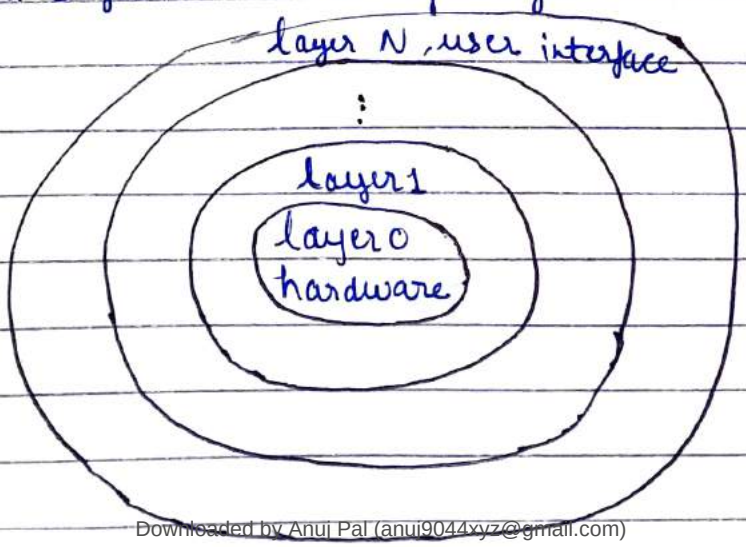
1. Simple Structure:

Operating systems such as MS-DOS and the original UNIX did not have well-defined structures. There was no CPU execution mode and SO errors in applications could cause the whole system to crash. Although MS-DOS has some structure, its interfaces and level of functionality are not well separated.



2. Layered approach

One way to achieve modularity in the operating system is the layered approach. In this, bottom layer is the hardware and the topmost layer is the user interface. All the layers hide some structures, operations etc from their upper layers. In this, each layer needs to be carefully defined. Because the upper layers can only use the functionalities of layers below them.



Page No.
Date

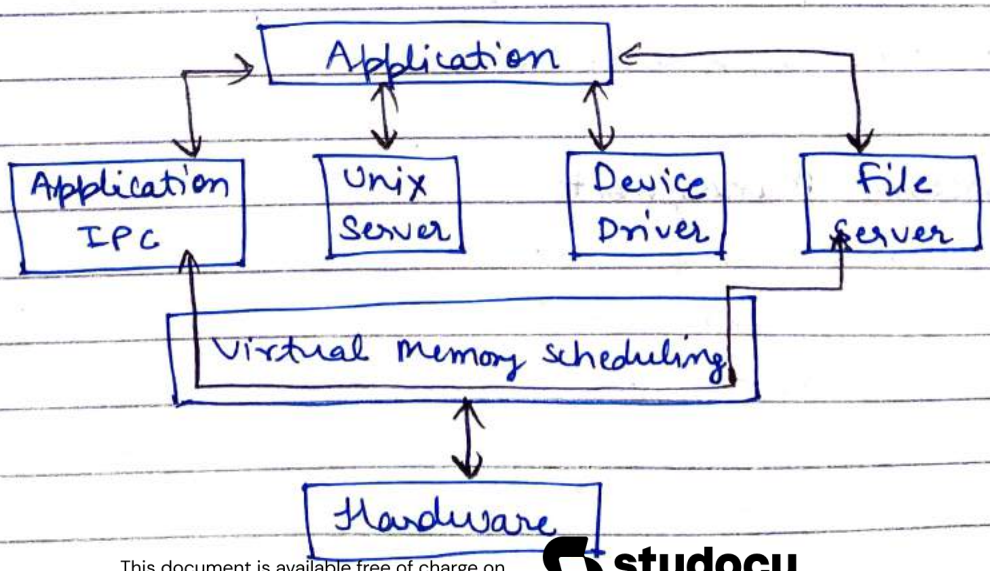
3. Micro-Kernels:

It is the classification of kernel. In this, the user services and kernel services are implemented in different address space. The user services are kept in user address space, and kernel services are kept under kernel address space, thus also reduces the size of kernel and as well as size of operating system.

It provides minimal services of process and memory management. The communication between client application and services running in user address space is established through message passing, reducing the speed of execution microkernel. The operating system remains unaffected as user services and kernel services are isolated so if any user service fails it does not affect kernel service.

And the microkernel is solely responsible for the most important services of operating system they are named as follows:

- Inter process - Communication
- Memory management
- CPU - scheduling



4. Modules:

A module is a separate unit of software & hardware which provide modularity. Modular components include portability which allows them to be used in a variety of systems and interoperability which allows them to function with the components of other systems.

Modular programming is the concept that similar functions should be contained within the same unit of programming code and that separate functions should be developed as separate units of code so that the code can easily be maintained and reused by different programs.

Components of Operating System

1. Memory Management

a) Maintain bookkeeping information: what part of memory are in use by whom, what part are not in use.
b) Map processes to memory locations: It decides which process will get memory when and how much.
c) Allocates the memory when a process requests it to do so.
d) De-allocates the memory when a process no longer needs it or has been terminated.

2. Processor Management

a) Allocates the processor to a process.
b) Deallocates processor when a process is no longer required.
c) It manages process scheduling, process synchronization, process communication, deadlock handling.
d) It keeps tract of processor and status of process.

3. Device Management

a) Allocate / Deallocates devices in the efficient way.

b) Decides which process gets the device when and for how much time.

c) Keeps track of all devices. Program responsible for this task is known as I/O controller

4. File Management

a) file creation or deletion.

b) Keeps track of information, location, uses, status etc. The collective facilities are often known as file system.

c) Support for hierarchical file systems.

Services provided by Operating System

1. Program Execution

The purpose of computer system is to allow the users to execute programs in an efficient manner. The OS provides an environment where the user can conveniently run these programs. To run a program, the program is required to be loaded into into the RAM first and then to assign CPU time for its execution.

2. I/O Operations

Each program requires an input and after processing it, produces output. This involves the use of I/O devices. The I/O service cannot be provided by user-level programs and is provided by the operating System

3. file system manipulation:

While working on the computer, generally a user is required to manipulate various types of like like opening, Saving & deleting a file from the storage disk.

4. Communication:

OS performs the communication among various types of processes in the form of shared memory. Such as message passing in the form of packets of information which is in predefined formats are moved between processes by the Operating System.

5. Error detection:

The main function of OS is to detect the errors like memory overflow and error related to I/O devices. After detecting the errors, Operating system takes an appropriate action for consistent computing.

6. Resource Allocation

In the multitasking environment, when multiple jobs are running at a time, it is the responsibility of an OS to allocate the required resources to each process for its better utilization.

7. Protection and Security

Protection involves ensuring that all access to system resources is controlled. Such security starts with requiring each user to authenticate him or her to the system, usually by means of a password, to gain access to system resources.

Shell

When a user logs in, the login programs check the username and password and then starts another program called the shell. A shell is a software that provides an interface for an operating system's user to provide access to the kernel's services.

The shell is just an environment where applications can run in protected memory space so that resources can be shared among multiple active shells; with the Kernel managing the resources requests for I/O.

Kernel

Kernel is the hub of the operating system, it allocates time and memory to programs and handles the file storage and communications in response to system calls. It is the most fundamental part of an operating system. It can be thought of as the program which controls all the other programs on the computer. The Kernel is the part of operating system that interacts with the hardware. Kernel gives the hardware interaction with user.

There are two types of kernel:

1) Monolithic Kernel
2) Micro-kernel

1) Monolithic Kernel

All the parts of a kernel like the scheduler, file system, memory management, networking stacks, device drivers etc., are maintained in one unit within the kernel in Monolithic Kernel

2) Micro-kernel

Only the importants parts like IPC, basic scheduler, basic memory handling are put into the kernel. Communication happen via message passing. others are maintained as Server processes in User space.

Page No. _____
Date | | | |

Difference between Monolithic Kernel & Micro-Kernel

Monolithic Kernel	Micro Kernel
1) Kernel size is large.	1) Kernel size is small.
2) OS is complex to design.	2) OS is easy to design.
3) All the OS services are includd in the kernel	3) Kernel provides only IPC and low level device management services
4) Request may be serviced faster.	4) Request may be serviced slower.
5) No message passing and no context switching are required while the kernel is performing the job.	5) Micro-Kernel requires message passing and context switching.

System Call

A system call is the programmatic way in which a computer program requests a service from the kernel of the OS it is executed on. A system call is a way for programs to interact with the operating system. A computer program makes a system call when it makes a request to the operating system's kernel.

System call provides the services of operating system to the user programs via Application Program Interface (API). It provides an interface between a process and operating system to allow user-level processes to request services of the operating system. System calls are the only entry points into the kernel system.

Page No.
Date

Services provided by System calls:

1. Process creation and management
2. Main memory management
3. File access, Directory and file system management
4. Device handling (I/O)
5. Protection
6. Networking

Types of System Calls:

There are 5 different categories of system calls-

1. Process control:

CreateProcess()	fork()
ExitProcess()	exit()
WaitforSingleObject()	wait()

2. File manipulation:

Createfile()	open()
Readfile()	read()
Writefile()	write()
CloseHandle()	close()

3. Device Manipulation:

SetConsoleMode()	ioctl()
ReadConsole()	read()
WriteConsole()	write()

4. Information Maintenance

GetCurrentProcessID()	getpid()
SetTimer()	alarm()
Sleep()	sleep()

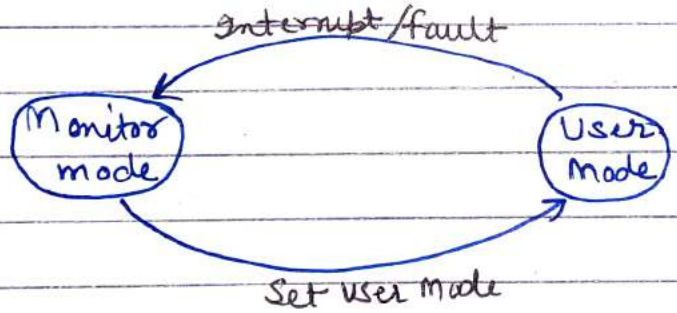
5. Communications:

CreatePipe()	pipe()
CreateFileMapping()	Shmget()
MapViewOfFile()	mmap()

Dual Mode operation of processors

In dual mode operation, there are two separate modes: monitor mode and user mode. In monitor mode the CPU can use all instructions and access all areas of memory. In user mode, the CPU is restricted to unprivileged instructions and a specified area of memory.

For indicating mode of the system, mode bit is used in the computer hardware. The mode bit is 0 for monitor and 1 for user.



Difference between Process Switch & Mode Switch

1) Process switching occurs when the processor switches from one process to another.
2) When process switching occurs, the kernel saves the context of old process in its PCB and loads the saved context of new process scheduled to run.
3) If an interrupt occurs, the system needs to save the current context of running process on CPU, so that it can restore that saved context when processing of interrupt is over.

Page No.
Date

1) Mode switching is the switching of a process between Kernel mode and user mode. A process can execute in either of these two modes.
2) Mode switching can be done using system call This is a special instruction that sets the system's state to Kernel mode.
3. If a user process needs to access things controlled by kernel, it is necessary to perform mode switching.

Printed Pages: 02 Sub Code: RCA301

Paper ID: 1 4 1 0 Roll No.

M. C. A.
(SEM III) THEORY EXAMINATION 2017-18
OPERATING SYSTEM

Time: 3Hours *Max. Marks: 70*

Note: Attempt all Sections. Assume missing data, if any.

SECTION A

1. Attempt *all* questions in brief. 2 x 7 = 14

a. What are advantages of spooling over buffering.
b. What do you mean by thrashing?
c. What is a lazy swapper and how it is used?
d. Differentiate between global and local page replacement.
e. What is the difference between hard real time and soft real time operating system?
f. What do you mean by Virtual Memory? Discuss in short.
g. Describe the difference between short-term scheduling and long-term scheduling.

SECTION B

2. Attempt any *three* of the following: 7 x 3 = 21

a. What do you mean by Kernel? Describe various operations performed by Kernel.
b. What do you mean by critical section problem? Write Peterson's Solution to solve this problem.
c. What do you mean by Inter-Process Communication? Explain fundamental models of IPC in details.
d. Find the page fault for following reference string (for 3 frames and 4 frames) using FIFO, LRU, MFU and Optimal page replacement algorithms:
1, 2, 3, 4, 1, 2, 5, 6, 3, 1, 3, 2, 4, 5
e. What is the concept of deadlock? Discuss the necessary conditions for deadlock with examples.

SECTION C

3. Attempt any *one* part of the following: 7 x 1 = 7

(a) Explain the structure of an operating system. Also discuss various components of an operating system.
(b) What do you mean by monolithic and micro lithic kernels?

4. **Attempt any *one* part of the following:** 7 x 1 = 7

(a) Describe Banker's algorithm for safe allocation.

(b) What is the average turnaround time and waiting time of following with the FCFS and SRTF scheduling algorithms?

Process	Arrival time	Burst time
P1	0.0	8
P2	4.4	4
P3	1.0	1

5. **Attempt any *one* part of the following:** 7 x 1 = 7

(a) Describe the Producer-consumer problem. Provide a solution to this problem using semaphores.

(b) What do you mean by critical section problem? Write Peterson's Solution to solve this problem.

6. **Attempt any *one* part of the following:** 7 x 1 = 7

(a) Find the page fault for following reference string (for 3 frames and 4 frames) using FIFO and LRU page replacement algorithms:

1, 2, 3, 4, 1, 2, 5, 6, 3, 1, 3, 2, 4, 5

Discuss whether Belady's Anomaly exists here or not.

(b) Explain the concept of segmentation with proper diagram.

7. **Attempt any *one* part of the following:** 7 x 1 = 7

(a) Describe the memory management and file system for LINUX operating system.

(b) Discuss the following with respect to file system:

(i) Consistency Checking

(ii) Common file attributes

Unit - 2

Concurrent Processes

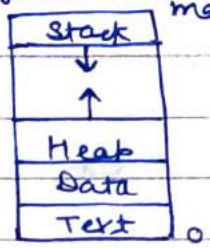
Page No. _____
Date

Process Model

A process is a sequential program in execution. A program defines the fundamental unit of computation of the computer. Process is an active entity.

When a program is doubled-clicked in windows then the program starts loading in memory and become a live entity. Process contains four sections:

a. ~~Data~~ Stack
b. Data
c. Heap
d. Stack.



1) Stack Section

This section contains local variable, function and return address. As stack and heap grow in opposite direction which is obvious if both grow in same direction they may overlap so it is good if they grow in opposite direction.

2) Heap Section

This section is used to provide dynamic memory whenever memory is required by the program during run-time. It is provided from heap section.

3) Data Section

This section contains the global variables and static local variables.

4) Text Section

This section contain the executable instructions, constants and macros, it is read-only location and is sharable so that can be used by another process also.

Page No.
Date

Difference between Process and Program

Process	Program
1) Process is an operation which takes the given instruction and performs the manipulation as per the code called execution.	1) Program is a set of instructions that perform a designated task.
2) A process is entirely dependent on program.	2) Program are independent.
3) Process is a module that executes concurrently. They are separable and loadable modules.	3) Program performs a task directly relating to an operation.
4) A process includes program counter, a stack, a data section and a heap	4) A program is just a set of instructions stored on disk.
5) It is an active entity.	5) It is a passive entity.

Difference between Busy wait and Blocking wait

Busy Wait	Blocking Wait
1) Busy wait is a loop that reads that status register over and over until the busy bit becomes clear.	1) Blocking is a situation where processes wait indefinitely within a semaphores
2) It occurs when scheduling overhead is larger than excepted wait time.	2) It occurs when process resources are needed for another task.
3) It is schedule based.	3) In this, schedule based algorithm is inappropriate.

Page No.
Date

Principle of Concurrency

Concurrency is the interleaving of processes in time to give the appearance of simultaneous execution. The fundamental problem in concurrency is processes interfering with each other while accessing a shared global resource.

```
chin = getchar();
chout = chin
putchar(chout);
```

Concurrency can be implemented and is used a lot on single processing units, it may benefit from multiple processing units with respect to speed. If an operating system is called a multi-tasking operating system, this a synonym for supporting concurrency.

If we can load multiple documents simultaneously in the tabs of our browser and we can still open menus and perform more actions, this is concurrency.

Producer Consumer Problem

Producer process produce data item that consumer process consumes later. Buffer is used between producer and consumer. Buffer size may be fixed or variable. The producer portion of the application generates data and stores it a buffer and the consumer reads data from the buffer. The producer cannot deposit its data if the buffer is full. Similarly, a consumer cannot retrieve any data if the buffer is empty. If the buffer is not full, a producer can deposit its data. The consumer should be allowed to retrieve a data item if buffer contains.

```
Producer {
    while( True)
    /* produce an item and put in nextProduced */
      while (count == BUFFER_SIZE);   // do nothing
       buffer[in] = nextProduced;
        in = (in+1) %. BUFFER_SIZE;
        Count ++;
  }

Consumer {
    while (True)
     while (count == 0)              // do nothing
     nextConsumed = buffer[out];
      out = (out+1)% BUFFER_SIZE;
      count --;
    /* consume the item in nextconsumed */
}
```

Critical Section Problem

A critical section is a code segment where the shared variables can be accessed. An atomic action is required in a critical section i.e. only one process can execute in its critical section at a time. All the other processes have to wait to execute in their critical sections.

```
do {
      Entry Section
          (critical section)
      Exit Section
          (remainder Section)
  } while (TRUE);
```

The entry section handles the entry into the critical section. It acquires the resources needed for execution by the process. The exit section handles the exit from the critical section. It releases the resources and also informs the other processes that the critical section is free.

Solution to the Critical Section Problem

1. Mutual Exclusion:

It implies that only one process can be inside the critical section at any time. If any other processes require the critical section, they must wait until it is free.

2. Progress:

Progress means that if a process is not using the critical section, then it should not stop any other processes from accessing it. In other words, any process can enter a critical section if it is free.

3. Bounded waiting:

Bounded waiting means that each process must have a limited waiting time. It should not wait endlessly to access the critical section.

Mutual Exclusion

When a process is accessing shared variable is known as in critical section. When no two processes can be in critical section at the same time, this state is known as mutual Exclusion. It is property of concurrency control which is used to prevent race conditions.

Page No. _____
Date

Conditions for mutual Exclusion:

1) NO two processes may at the same moment be inside their critical sections.
2) NO assumptions are made about relative speed of processor or number of CPUS.
3) NO process outside the critical section should block other processes.
4) NO process should wait arbitrary long to enter its critical section.

Bakery Algorithm

Bakery algorithm is used in multiple process solution. It solves the problem of critical section for n processes. Before entering its critical section, process receives a ticket number. Holder of the smallest ticket number enters the critical section. The Baker algorithm cannot guarantee that two processes do not receive the same number. If processes P_i and P_j receives the same number, if $i<j$, then P_i is served first else P_j is served first.

```
do{
    choosing[i]=true;
    number[i] = max(number[0], number[1], ..., number[n-1]) + 1;
    for(j=0; j<n; j++){
        while(choosing[j]);
        while((number[j]!=0) && ((number[j],j)<(number[i],i)));
    }
    critical section
    number[i]=0;
    remainder section
} while(1);
```

Page No. _____
Date

Dekker's Algorithm

It is the first known algorithm that solves the mutual exclusion problem in concurrent programming. Dekker's algorithm is used in process queuing and allows two different threads to share the same single-use resources without conflict by using shared memory for communication.

Dekker's algorithm will allow only a single process to use a resource if two processes are trying to use it at the same time. It succeeds in preventing the conflict by enforcing mutual exclusion, meaning that one process may use the resource at a time and will wait if another process is using it. This is achieved with the use of two "flags" and a "token".

The flags indicate whether a process wants to enter the critical section or not, a value of 1 means TRUE that the process wants to enter the CS, while 0 or FALSE means the opposite. The token, which can have a value of 1 or 0, indicates priority when both processes have their flags set to TRUE.

Busy waiting

```
do
{
        wait(mutex);
            // critical section
        signal(mutex);
            // remainder section
}
while(TRUE);
```

Busy waiting wastes CPU cycles that some other process might be able to use productivity.

Page No.
Date

Peterson's solution

Peterson's algorithm is used for mutual exclusion and allows two processes to share a single-use resource without conflict. It uses only shared memory for communication. Peterson's formula orginally worked only with two processes, but has since been generalized for more than two.

Assume that the LOAD and STORE instructions are atomic i.e. cannot be interrupted.

Two processes share two variables:

→ int turn;

→ Boolean flag[2]

The variable turn indicates whose turn it is to enter the critical section. The flag array is used to indicate if a process is ready to enter the critical section. flag[i] = true implies that process Pi is ready.

Algorithm:

```
do {
        flag[i] = TRUE;
        turn = j;
        while (flag[j] && turn == j);
                critical section
        flag[i] = FALSE;
                remainder section
    } while (1)
```

Two processes executing concurrently:

```
do{                                   do{
    flag 1 = TRUE;                        flag 2 = TRUE;
    turn = 2;                             turn = 1;
    while (flag 2 && turn == 2)           while (flag 1 && turn == 1)
    critical section                      critical section
    flag 1 = FALSE;                       flag 2 = FALSE;
    remainder section                     remainder section
} while(1)                            } while (1)
```

A system is said to be concurrent if it can support two or more actions in progress at the same time. Concurrency is the property of the program.

A system is said to be parallel if it can support two or more actions executing simultaneously. Parallel execution is the property of the machine.

Semaphores

Semaphores is simply a variable. This variable is used to solve the critical section problem and to achieve process synchronization in the multiprocessing environment. The two most common kinds of semaphores are counting semaphores and binary semaphores. Counting semaphores can take non-negative integer values and binary semaphore can take the value 0 & 1 only.

Semaphores are integer variables that are used to solve the critial section problem by using two atomic operations, wait and signal that are used for process synchronization

1. Wait

The wait operation decrements the value of its argument S, if it is positive. If S is negative or zero then no operation is performed.

```
wait (s)
{
    while (s<=0);
    s--;
}
```

2. Signal

The signal operation increments the value of its argument.

```
signal (s)
{
s++;
```

Page No.
Date

Advantages of Semaphores

1) Semaphores allow only one process into the critical section. They follow the mutual exclusion strictly.
2) There is no resource wastage because of busy waiting in Semaphores.
3) Semaphores are implemented in microkernel. So, they are machine independent.

Disadvantages of Semaphores:

1) Semaphores are complicated so the wait and signal operations must be implemented in the correct order to prevent deadlocks.
2) Semaphores may lead to priority inversion where low priority processes may access the critical section first and high priority processes later.

Reader - Writer Problem

The reader-writer problem relates to an object such as a file that is shared between multiple processes. Some of these processes are readers i.e they only want to read data from object and some of the processes are writers. i.e. they want to write into the object.

The reader-writer problem is used to manage synchronization so that no problems with the object data. for ex- If two readers access the object at the same time, there is no problem. However if two writers or a reader and writer may access object at the same time, there may be problems.

To solve this problem, a writer should get exclusive access to an object i.e when a writer is accessing the object, no reader or writer may access it. However, multiple readers can access the object at the same time.

Page No. _____
Date

Reader Process:

```
wait (mutex);
rc++;
if (rc == 1)
   wait (wrt);
signal (mutex);
   .
   .
   .
   //Read
   .
   .
   .
wait (mutex)
rc--;
if (rc == 0)
signal (wrt);
signal (mutex);
```

Writer process

```
wait (wrt);
   .
   .
   .
   // write
signal (wrt);
```

Classical Problem in Concurrency

Concurrency is the interleaving of processes in time to give the appearance of simultaneous execution. Thus it differs from parallelism, which offers genuine simultaneous execution. However the issues and difficulties raised by the two overlap to a large extent:

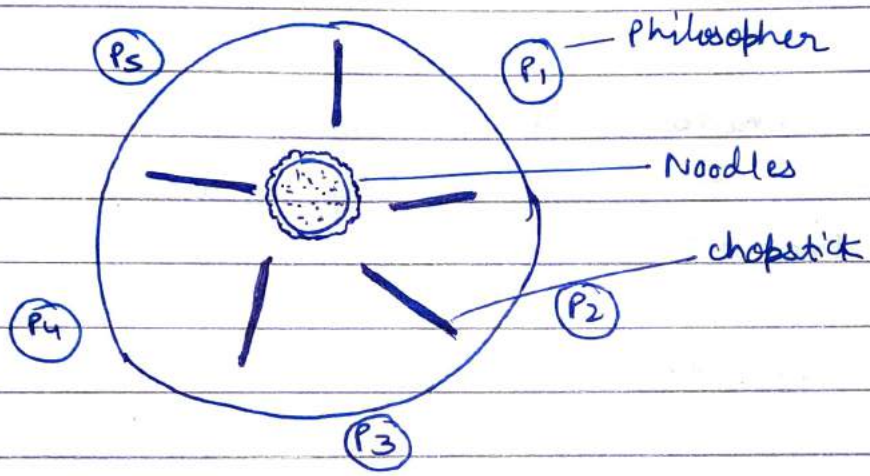
- Sharing global resources safely is difficult
- optimal allocation of resources is difficult.
- locating programming errors can be difficult.

Ex:

```
chin = getchar();
chout = chin;
putchar (chout);
```

The Dining philosopher problem:

It states that K philosophers seated around a circular table with one chopstick between each pair of philosophers. There is one chopstick between each philosopher. A philosopher may eat if he can can pickup the two chopsticks adjacent to him. One chopstick chopstick may be picked by any one of its adjacent followers but not both.



Semaphore Solution to Dining Philosopher:

```
Process P[i]
while true do
{   THINK;
    PICKUP(CHOPSTICK[i], CHOPSTICK[i+1 mod 5]);
    EAT;
    PUTDOWN(CHOPSTICK[i], CHOPSTICK[i+1 mod 5]);
}
```

There are three states of philosopher: THINKING, HUNGRY and EATING. Here, there are two semaphores: Mutex and a semaphore array for the philosophers. Mutex is used such that no two philosophers may access the pickup or putdown at the same time. The array is used to control the behaviour of each philosopher.

Page No.
Date

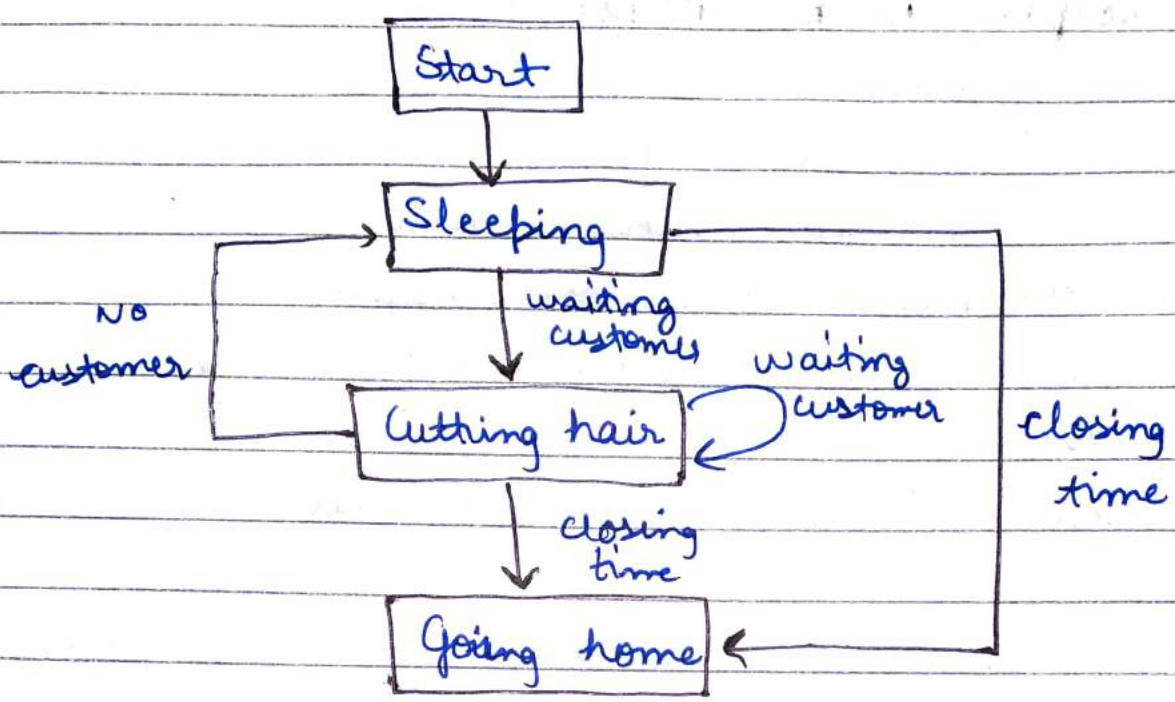
Sleeping Barber Problem:

The analogy is based upon a hypothetical barber shop with one barber. There is a barber shop which has one barber, one barber chair, and n chairs for waiting for customers if there are any to sit on the chair.

- If there is no customer, then the barber sleeps in his own chair.
- When a customer arrives, he has to wake up the barber.
- If there are many customers and the barber is cutting a customer's hair, then the remaining customers either wait if there are empty chairs in the waiting room or they leave if no chairs are empty.

Solution: The solution to this problem includes three semaphores. first is for the customer which counts the number of customers present in the waiting room. Second for barber 0 or 1 is used to tell whether the barber is idle or is working. and third mutex is used to provide the mutual exclusion which is required for the process to execute.

In the solution, the customer has the record of the number of customers waiting in the waiting room if the number of customers is equal to the number of chairs in the waiting room then the upcoming customer leaves the barbershop.



Unit 3:

CPU Scheduling

Page No.
Date

CPU scheduling is a process which allows one process to use the CPU while the execution of another process is on hold due to unavailability of any resources, thereby making full use of CPU. The aim of CPU scheduling is to make the system efficient, fast and fair.

Whenever the CPU becomes idle, the operating system must select one of the processes in the ready queue to be executed.

Performance Criteria for CPU scheduling:

1. CPU Utilization

CPU utilization is the average function of time, during which the processor is busy.

2. Throughput

It refers to the amount of work completed in a unit of time. The number of processes the system can execute in a period of time. The higher the number, the more work is done by the system.

3. Waiting time

The average period of time, a process spends in waiting.

4. Turnaround time

It is the time interval from the time of submission of a process to the time of the completion of the process.

5. Response Time

It is the difference between first execution time and Arrival time.

6. Priority

Give preferential treatment to processes with higher priorties.

7. Balanced Utilization

Utilization of memory, I/O devices and other system resources are also considered.

8. Fairness

Avoid the process from starvation. All the processes must be given equal opportunity to execute.

Objective of CPU Scheduling:

1) Efficiency must be very high.
2) Maximize throughput
3) Minimize response time
4) Minimize overhead
5) Maximize resource use
6) Avoid Indefinite postponement
7) Enforce priorities

Types of Scheduling

1) Pre-emptive Scheduling
2) Non Pre-emptive Scheduling

Pre-emptive Scheduling:

Processor can be pre-empted to execute a different process in the middle of execution of any current process.

Non pre-emptive Scheduling:

Once processor start to execute a process it must finish it before executing the other. It cannot be halted in middle.

Page No.
Date

Scheduler

Scheduler are special system software which handle process scheduling in various ways. Their main task is to select the jobs to be submitted into the system and to decide which process to run.

Schedulers are of three types –

- Long-term Scheduler
- Short-term scheduler
- Medium-term Scheduler

Long-term Scheduler

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

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

Short-term Scheduler

It is also called as CPU scheduler. Its main objective is to increase system performance in accordance with the chosen set of criteria. It is the change of ready state to running state of the processes. CPU scheduler selects a process among the processes that are ready to execute and allocates CPU to one of them.

It is also known as dispatchers, make the decision of which process to execute next. Short-term schedulers are faster than long-term schedulers.

Page No.
Date

Medium Term Scheduler

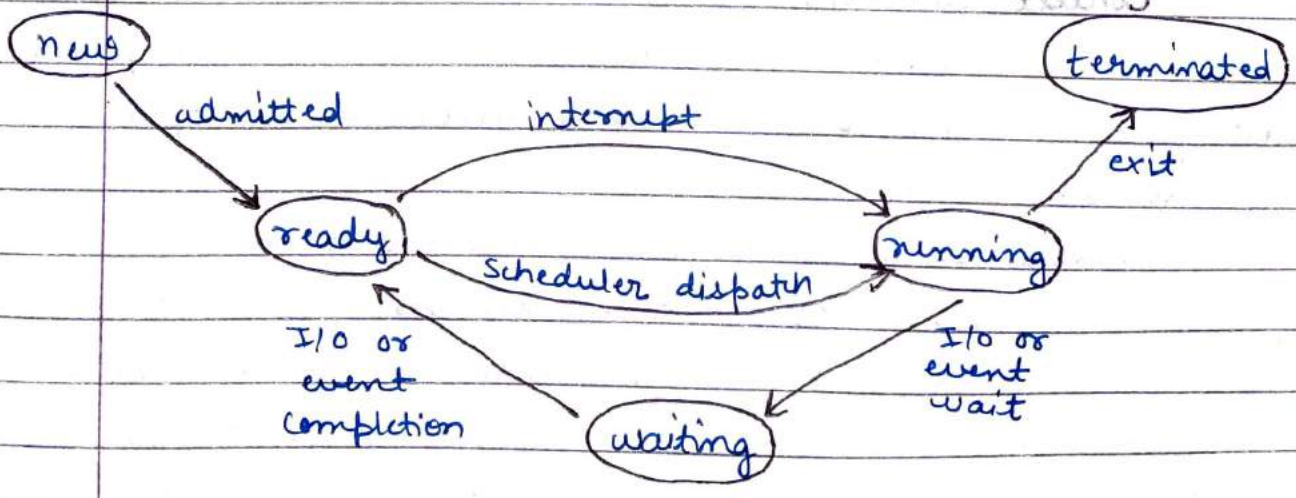
It is a part of scheduling. It removes the processes from the memory. It reduces the degree of multiprogramming.

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

In this condition, to remove the process from memory and make space for other processes, the suspended process is moved to the secondary storage. This process is called swapping and the process is said to be swapped out or rolled out.

Different Process States

- New- The process is being created.
- Ready- The process is waiting to be assigned to a processor.
- Running - Instructions are being executed.
- waiting- The process is waiting for some event to occur.
- Terminated- The process has finished execution.



Page No.
Date

Process Control Block

There is a Process Control Block for each process enclosing all the information about the process. It is a data structure, which contains the following:

1) Naming the process
2) State of the process
3) Resources allocated to the process
4) Memory allocated to the process
5) Scheduling information
6) I/o devices associated with process

Components of Process Control Block

1) Process ID
2) Process State
3) Program Counter
4) Register information
5) Scheduling information
6) Memory related information
7) Accounting information
8) Status information related to I/O.

Context Switching:

A context switch occurs when a computer's CPU switches from one process or thread to a different process or thread.

Context switching allows for one CPU to handle numerous processes or threads without the need for additional processors.

A context switch is the mechanism to store and restore the state or context of a CPU in process control block so that a process execution can be resumed from the same point at a later time.

Date

Process Address space

An address space is a range of valid addresses in memory that are available for a program or process. That is, it is the memory that a program or process can access. The memory can be either physical or virtual and is used for executing instructions and storing data.

Address space can be of two types:

a) Physical address space:

Physical address space is created in RAM.

b) Virtual address space:

Virtual address space is an address space that is created outside the main memory inside the virtual memory and it is created in hard disk.

Thread

A thread is a flow of execution through the process code, with its own program counter, system registers and stack. A thread is a path of execution within a process. A process can contain multiple threads.

A thread is known as light weight process. The idea is to achieve parallelism by dividing a process into multiple threads.

Advantages of Thread:

1) Thread minimize context switching time.
2) Use of thread provides concurrency within a process
3) Efficient communication.
4) It is more economical to create and context switch threads.

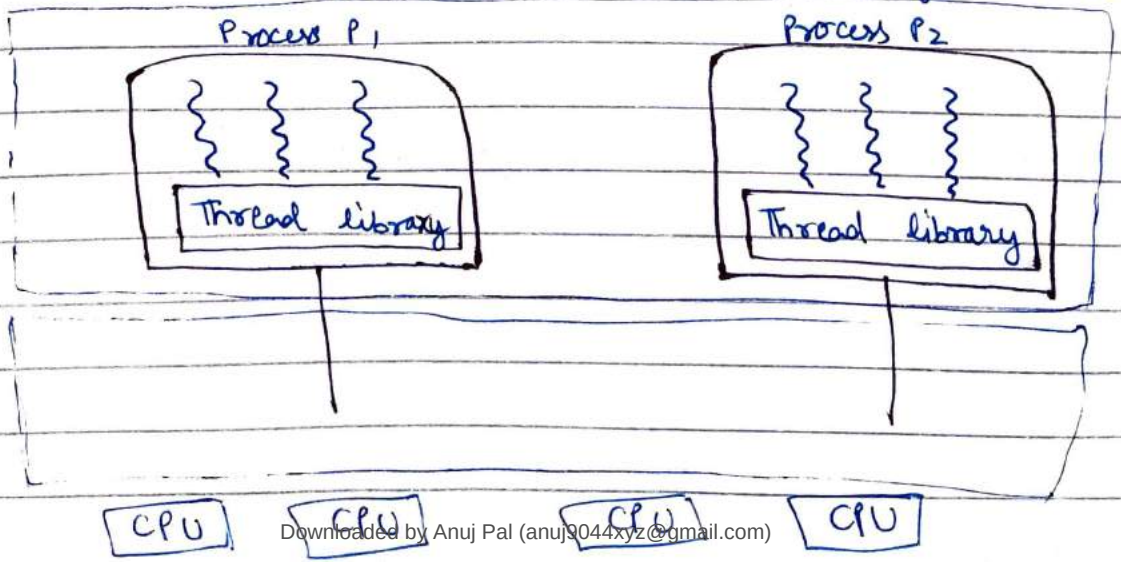
Process	Thread
1) Process is heavy weight.	1) Thread is light weight.
2) Process switching needs interaction with operating System	2) Thread switching does not need to interact with OS.
3) If one process is ~~execute~~ blocked, then no other process can execute.	3) while one thread is block, then second thread in the same task can run.
4) In multiple process implementation each process execute the same code but has its own memory and file resources	4) All threads can share same set of open files, child processes.
5) Each process operates independently of the other.	5) One thread can read, write or even completely wipe out another threads stack.

Types of Thread

1) User level threads
2) Kernel level threads

1) User level Threads

The thread library contains code for creating and destroying threads, for passing message and data between threads, for scheduling thread execution and for saving and restoring thread contexts. The application starts with a single thread.



Page No.
Date

Advantages:

1) Thread switching does not require Kernel mode privileges
2) User level thread can run on any OS.
3) User level thread are fast to create and manage.
4) Scheduling can be application specific.

Disadvantages:

1) In some OS, most system calls are blocking.
2) Multithreaded application cannot take advantage of multiprocessing.

2) Kernel Level Threads

Thread management is done by Kernel. Kernel threads are supported directly by the OS. Any application can be programmed to be multithreaded. All of the threads within an application are supported within a single process.

The kernel maintains context information for the process as a whole and for individuals threads within the process. Scheduling by the kernel is done on a thread basis. The Kernel performs thread creation, scheduling and management in kernel space. Kernel threads are generally slower to create and manage than the user threads.

Advantages:

1) If one thread in a process is blocked, the kernel can schedule another thread of the same process.
2) Kernel can simultaneously schedule multiple threads from the same process on multiple processes.
3) Kernel routines themselves can be multithreaded.

Disadvantages:

1) Transfer of control from one thread to another within the same process requires a mode switch to the Kernel.

2) Kernel threads are generally slower to create and manage than the user threads.

Thread Cancellation:

It is the task of terminating a thread before it has completed. for example, if multiple threads are concurrently searching through a database and one thread returns the result, the remaining threads might be cancelled.

First Come First Serve Algorithm

- Jobs are executed on first come, first serve basis.
- Easy to understand and implement
- Its implementation based on FIFO queue.
- Poor in performance as average wait time is high.

Advantages: 1) Better for long processes
2) No starvation and simple method

Disadvantages: 1) Even very small process should wait for its turn to come to utilize the CPU.
2) Throughput is not emphasized.

Completion time: Time taken for execution to complete, starting from arrival time.

Turn around time: Time taken to complete after arrival.
= Completion time - Arrival time

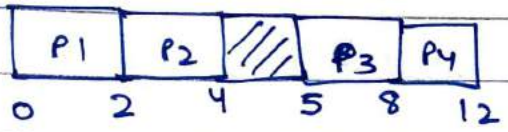
Waiting time: It is the difference between the Turn around time and burst time of the process.
= Turnaround time - Burst time

Page No.
Date

Q.

Process	Arrival time	Burst time	Completion time	Turn around	waiting time	Respone time
P_1	0	2	2	2	0	0
P_2	1	2	4	3	1	1
P_3	5	3	8	3	0	0
P_4	6	4	12	6	2	2

Gantt chart



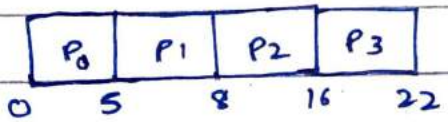
Average turn around time = $\frac{2+3+3+6}{4} = 3.5$

Average waiting time = $\frac{0+1+0+2}{4} = 0.75$

CPU utilization = $\frac{11}{12} \times 100 = 91.66\%$.

Q.

Process	Arrival time	Execution time	Completion time	Turn around	Waiting time
P_0	0	5	5	5	0
P_1	1	3	8	7	4
P_2	2	8	16	14	6
P_3	3	6	22	19	13



Gantt chart

Average turn around time = $\frac{5+7+14+19}{4} = 11.25$

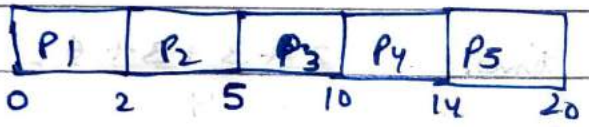
Average waiting time = $\frac{0+4+6+13}{4} = 5.75$

CPU Utilization = $\frac{22}{22} \times 100 = 100\%$.

Page No. _____
Date

Q.

Process	Arrival time	Burst time	Turnaround time	Waiting time	Completion time
P_1	0	2	2	0	2
P_2	1	3	4	1	5
P_3	2	5	8	3	10
P_4	3	4	11	7	14
P_5	4	6	16	10	20



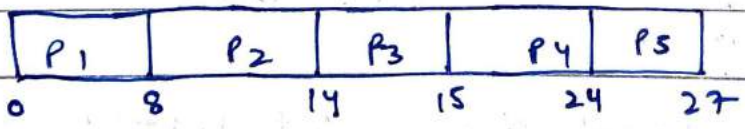
Average turn around time = $\frac{2+4+8+11+16}{5} = 8.2$

Average waiting time = $\frac{0+1+3+7+10}{5} = 4.2$

CPU utilization = $\frac{20}{20} \times 100 = 100\%$

Q.

	Burst	Priority
P_1	8	4
P_2	6	1
P_3	1	2
P_4	9	2
P_5	3	3



Average turn around time = $\frac{8+14+15+24+27}{5} = 17.6$

Average waiting time = $\frac{0+8+14+15+24}{5} = 12.2$

CPU utilization = 100%.

Page No.
Date

Shortest Job first (SJF) Scheduling Algorithm

Shortest job first or shortest job next (SJN) is a scheduling policy that selects the waiting process with the smallest execution time to execute next. SJN is a non-preemptive algorithm.

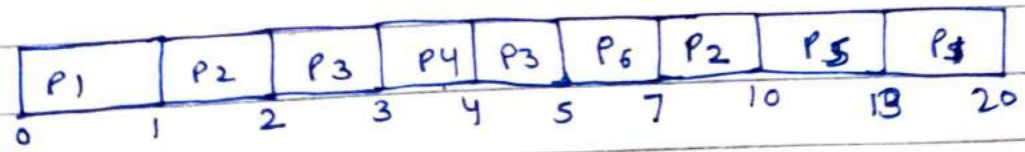
- SJF has the advantage of having minimum average waiting time among all scheduling algorithms.
- It is a Greedy Algorithm
- It may cause starvation if shorter processes keep coming.
- It is practically infeasible as OS may not know burst time and therefore may not sort them.

Advantages: Throughput is high.

Disadvantages: Starvation may be possible for longer processes

Q.

Process	Arrival time	Burst time	Completion time	Turn Around time	Waiting time
P1	0	7	20	20	12
P2	1	3	10	9	5
P3	2	1	5	3	1
P4	3	1	4	1	0
P5	4	3	13	9	6
P6	5	2	7	2	0



Average turn around time = $\frac{20+9+3+1+9+2}{6} = 7.333$

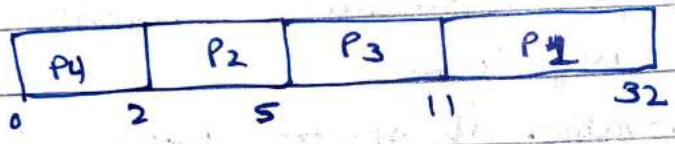
Average waiting time = $\frac{12+5+1+0+6+0}{6} = 4$

Date

Non-preemptive Shortest Job first

Q.

Process	Burst time
P_1	21
P_2	3
P_3	6
P_4	2

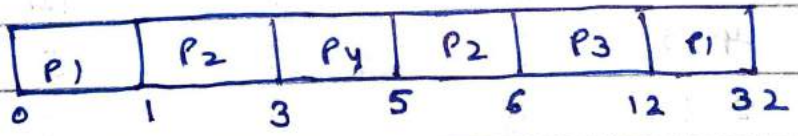


Average waiting time = $\frac{0+2+5+11}{4} = 4.5$

Pre-emptive shortest Job first

Q

Process	Burst time	Arrival time	completion time	Turn Around time	waiting time
P_1	21 20	0	32	32	11
P_2	3 1	1	6	5	2
P_3	6	2	12	10	4
P_4	2	3	5	2	0



Average turn around time = $\frac{32+5+10+2}{4} = 12.25.$

Average waiting time = $\frac{11+2+4+0}{4} = 4.25$

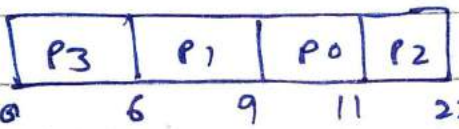
Date

Priority Scheduling Algorithm

- Priority scheduling is a non-preemptive algorithm
- Each process is assigned a priority. Process with highest priority is to executed first and so on.
- Processes with same priority are executed on first come first served basis.

Q.

Process	Arrival time	Execution time	Priority	Completion time	Turnaround time	waiting time
P_0	0	5	1	11	11	0
P_1	1	3	2	9	8	6
P_2	2	8	1	22	20	9
P_3	3	6	3	6	3	11



Average waiting time = $\frac{0+6+9+11}{4} = 6.5$

Average turn around time = $\frac{11+8+20+3}{4} = 10.5$

Advantages: 1) The priority of a process can be selected based on memory requirement, time requirement or user preference.

2) Priority scheduling provides a good mechanism where the relative importance of each process may be precisely defined.

Disadvantages:

If high priority processes use up a lot of CPU time, lower priority processes may starve and be postponed indefinitely.

Round Robin Scheduling Algorithm

Round Robin is a CPU scheduling algorithm where each process is assigned a fixed time slot in a cyclic way. It is simple, easy to implement and starvation free as all processes get fair share of CPU.

One of the most commonly used technique in CPU scheduling as a core. It is preemptive as processes are assigned CPU only for a fixed slice of time at most.

A fixed time is allocated for each process called quantum for execution. Once a process is executed for given time period that process is preempted and other process executes for given time period.

Advantages of RR scheduling:

1) fair treatment for all the processes.
2) Good response time.

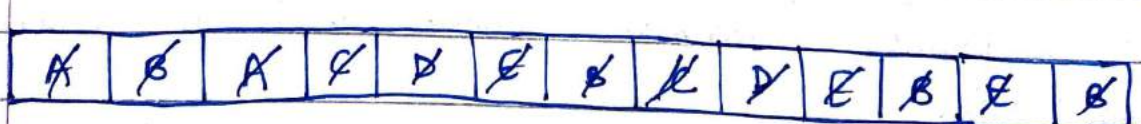
Disadvantages of RR scheduling:

1) Care must be taken in choosing quantum value.
2) Throughput is low if time quantum is too small.

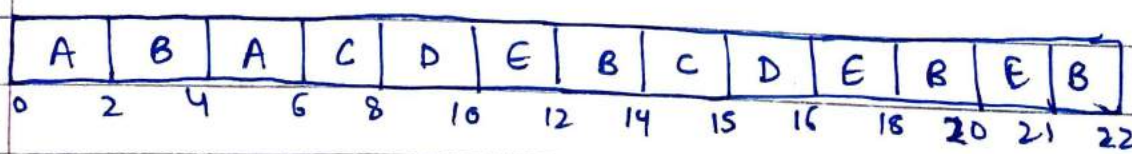
Q.

Process	Arrival time	Execution time	completion time	Turnaround time	Waiting time
A	0	4 2	6	6	2
B	2	7 8 3 1	22	20	13
C	3	3 7	15	12	9
D	3.5	3 7	16	12.5	9.5
E	4	5 3 1	21	17	12

Ready Queue



Running Queue



Page No.
Date

Average turnaround time = $\frac{6 + 20 + 12 + 12.5 + 17}{5} = 13.5$

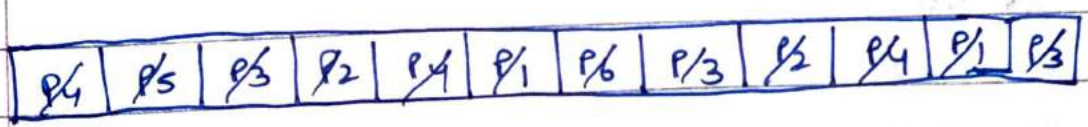
Average waiting time = $\frac{2 + 13 + 9 + 9.5 + 12}{5} = 9.1$

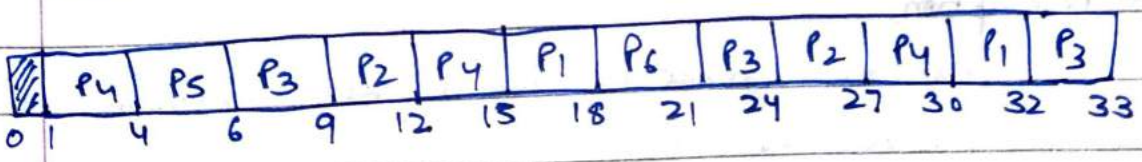
Q.

Process	Arrival time	Burst time
P_1	5	5
P_2	4	6
P_3	3	7
P_4	1	9
P_5	2	2
P_6	6	3

Sol.

Process	Arrival time	Burst time	Completion time	Turnaround time	waiting time
P_4	1	~~9 6 3~~	30	29	20
P_5	2	~~2~~	6	4	2
P_3	3	~~7 4 1~~	33	30	23
P_2	4	~~6 3~~	27	23	17
P_1	5	~~8 2~~	32	27	22
P_6	6	~~3~~	21	15	12



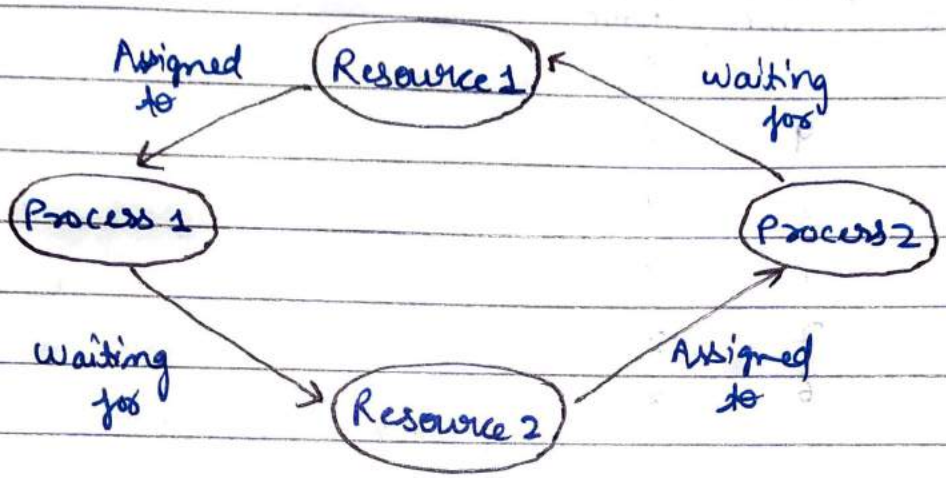


Average turn around time = $\frac{29 + 4 + 30 + 23 + 27 + 15}{6} = 21.33$

Average waiting time = $\frac{20 + 2 + 23 + 17 + 22 + 12}{6} = 16$

Deadlock

Deadlock is a situation where a set of processes are blocked because each process is holding a resource and waiting for another resource acquired by some other processes.



Deadlock can arise if following four conditions hold simultaneously:

1) Mutual Exclusion:
   One or more than one resources are non-sharable.

2) Hold and wait:
   A process is holding at least one resource and waiting for resources.

3) NO Preemption:
   A resource cannot be taken from a process unless the process releases the resource.

4) Circular wait:
   A set of process are waiting for each other in circular form.

Methods for handling deadlock

There are three ways to handle deadlock

1) Deadlock prevention or avoidance
2) Deadlock detection and recovery
3) Ignore the problem all together

Deadlock prevention:

1) Avoid mutual exclusion

If a process could have been used by more than one process at the same time then the process would have never been waiting for any resource.

2) Avoid hold and wait

A process must be assigned all the necessary resources before the execution starts. A process must not wait for any resource once the execution has been started.

3) Avoid no preemption

Deadlock arises due to the fact that a process can't be stopped once it starts. However, if we take the resource away from the process which is causing deadlock then we can prevent deadlock.

4) Avoid Circular wait

we can assign a priority number to each of the resource. A process can't request for a lesser priority resource. This ensures that not a single process can request a resource which is being utilized by some other process and no cycle will be formed.

Avoidance Algorithm:

- Single instance of a resource type - Resource-Allocation graph
- Multiple instance of a resource type - Banker's algorithm

Page No.
Date

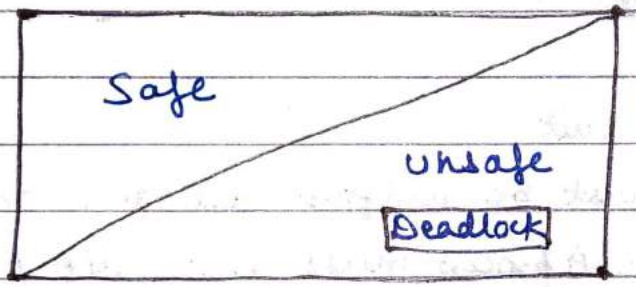
Deadlock Avoidance

Deadlock avoidance allows the three necessary conditions but makes judicious choice to assure that the deadlock is never reached.

Two approaches are used to avoid deadlock:

1. Do not start a process if its demands might lead to deadlock.
2. Do not grant an incremental resource request to a process if this allocation might lead to deadlock.

- System can be in one of the following states:



1. Safe state:

when the system can allocate resources to each process in some order and avoid a deadlock.

2. Unsafe state:

If the system did not follow the safe sequence of resource allocation from the beginning and it is now in a situation, which may lead to a deadlock.

3. Deadlock state:

If the system has some circular-wait condition existing for some processes, then it is in deadlock.

Unit - 4

Memory Management

Page No.
Date

Memory management is the process of controlling and coordinating computer memory, assigning portions called blocks to various running programs to optimize overall system performance.

Memory management is mainly concerned with the allocation of main memory to requesting process. No process can ever run, before a certain amount of memory is allocated to it.

Five requirements of memory management are:

1. Relocation:

Relocation is to find a way to map virtual address into physical address.

When a process is loaded in main memory, since there are several instructions insides the process so here address of these different instructions inside the process are relocatable address, which are convated into actual address by loader in software approach.

2. Sharing:

Any protection mechanism must have the flexibility to allow several processes to access the same portion of main memory.

3. Protection

Memory protection is to prevent a process from accessing memory that has not been allocated to it. Every process should protect against unwanted interference by other processes.

4. Logical Organization

Main memory is organized as linear or one-dimensional space that consists of sequence of bytes or words.

Page No.
Date

Secondary memory at its physical level is similarly organized. Most of the programs are organized into modules.

5. Physical Organization

The computer memory is organized as main memory and secondary memory. The main memory provides fast access. However the secondary memory is slower but can used for long term storage with large capacity.

Address binding

Address binding is the process of mapping the program's logical or virtual addresses to corresponding physical or main memory addresses.

Address binding is part of computer memory management and it is performed by the operating system on behalf of the applications that need access to memory.

Instructions and data to memory addresses can be done in following ways:

i) Compile time:

When it is known at compile time where the process will reside, compile time binding is used to generate the absolute code.

ii) Load time:

When it is not known at compile time where the process will reside in memory, then the compiler generates relocatable code.

iii) Execution time:

If the process can be moved during its execution from one memory segment to another, then binding must be delayed to be done at run time.

Page No.
Date

Dynamic loading

Dynamic loading refers to mapping an executable or library into a process's memory after is has started. In dynamic loading, a routine of program is not loaded until it is called by the program.

Dynamic loading makes better memory space utilization and unused routines are never loaded. It is useful when large amounts of code are needed to handle infrequently occuring cases.

Dynamic Linking

Linking is the process of collecting and combining various modules of code and data into a executable file that can be loaded into memory and executed. OS can link system level libraries to a program.

When it combines the libraries at load time, the linking is called static linking and when this linking is done at the time of execution, it is called as dynamic linking.

Overlays:

Overlays is used to keep in memory only those instructions and data that are needed at any given time. When other instructions are needed, they are loaded into space that was occupied previously by instructions that are no longer needed.

Swapping:

Swapping makes it possible for the total physical address space of all processes to exceed the real physical memory of the system, thus increasing the degree of multiprogramming in a system.

Logical V/S Physical Address Space

Logical address space is set of all logical address generated by program. Physical address space is a set of all physical addresses corresponding to these logical addresses.

Logical address is generated by CPU. Physical address is the address of main memory and it is loaded into the main memory address register.

Q. Consider a logical address space of eight pages of 1024 words, each mapped onto a physical memory of 32 frames then:

a) How many bits are in logical address?

b) How many bits are in physical address?

Sol. The logical address is split into two parts:
the page address and then offset.

Page address = $8 = 2^3$ = 3 address bits.

offset = $1024 = 2^{10}$ = 10 address bits.

Total = 13 address bits.

The physical address is split into two parts:
the frame address and then offset.

frame address = $32 = 2^5$ = 5 address bits

offset = $1024 = 2^{10}$ = 10 address bits

Total = 15 address bits.

Holes in Memory partitioning

The OS keeps a table indicating which parts of memory are available and which are occupied. Initially, all memory is available for user processes and is considered one large block of available memory known as hole. When a part of memory is occupied by a process and left rest of the memory also known as hole. A hole can be created when a process is completed and leaves the memory.

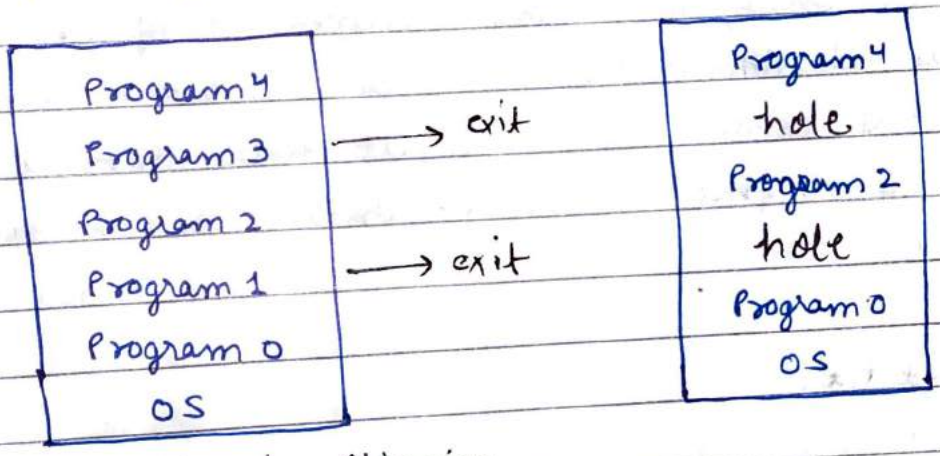
There are two Management techniques: Contiguous and Non-Contiguous.

In contiguous technique, executing process must be loaded entirely in main-memory. Contiguous Technique can be divided into:

1. fixed / Static partitioning
2. Variable/ dynamic partitioning

1) fixed / Static partitioning:

Partition main memory into a set of non-overlapping memory regions called partitions. fixed partitions can be of equal or unequal sizes.



2) Variable / Dynamic partitioning

Initially RAM is empty and partitions are made during the run-time according to process's need instead of partitioning during system configure. The size of partitions will be equal to incoming processes. The partition size varies according to the need of the process so that the internal fragmentation can be avoided to ensure efficient utilisation of RAM.

Page No.
Date

Dynamic Memory Allocations technique

first fit, best fit and worst fit are the the most common strategies used to select a free hole from the set of available holes.

1. first fit:

In the first fit approach is to allocate the first free partition or hole large enough which can accomodate the process. It finishes after finding the first suitable free partition.

2. Best fit:

The best fit deals with allocating the smallest free partition which meets the requirement of the requesting process. This algorithm first searches the entire list of free partitions and considers the smallest hole that is adequate. It then tries to find a hole which is close to actual process size needed.

3. Worst fit:

In worst fit approach is to locate largest available free portion so that the portion left will be big enough to be useful. It is the reverse of best fit.

Internal Fragmentation

It somehow relates to fixed size partitioning. The system allocates memory to various programs and processes by dividing them into small blocks as required by the program. However, more memory is allocated sometimes than is needed by the process, which eventually results in excess memory going waste or left unused.

external fragmentation

The main memory forms holes between portions of allocated memory that are too small to hold any process. It's downside of storage allocation algorithms, when contiguous blocks of unused space cannot serve a new request because the spaces are too small for large memory application needs.

In simple terms, the non-contiguous blocks create holes in the memory resulting in unused storage that are outside the allocated regions, meaning it cannot be used along with the main memory for larger memory tasks.

They end up being isolated and cannot be totally eliminated from the memory space. This is called external fragmentation. It can be removed by compaction which shuffles contents of the memory to place all free memory together.

Q. Given memory partitions of 100K, 500K, 200K, 300K and 600K (in order). How would each of the first fit, best fit and worst fit algorithms place processes of 212K, 417K, 112K and 426K ? which algorithm makes the most efficient use of memory?

Sol

Process	Memory
P_1	212K
P_2	417K
P_3	112K
P_4	426K

first fit:

Memory partition	Process Number	Memory Requested	Status	Internal fragmentation
100K	-	-	free	-
500K	P_1	212K	Busy	288K
200K	P_3	112K	Busy	88K
300K	-	-	free	-
600K	P_2	417K	Busy	183K

Total available: 1700K Total used: 741K

Page No.
Date

Best fit:

Memory Partition	Process Number	Memory Requested	Status	Internal fragmentation
100 K	–	–	free	–
500 K	P_2	417 K	Busy	83 K
200 K	P_3	112 K	Busy	88 K
300 K	P_1	212 K	Busy	88 K
600 K	P_4	426 K	Busy	174 K

Total available: 1700 K Total used: 1167 K.

Worst fit:

Memory partition	Process Number	memory Requested	Status	Internal Fragmentation
100 K	–	–	free	–
500 K	P_2	417 K	Busy	83 K
200 K	–	–	free	–
300 K	P_3	112 K	Busy	188 K
600 K	P_1	212 K	Busy	388 K

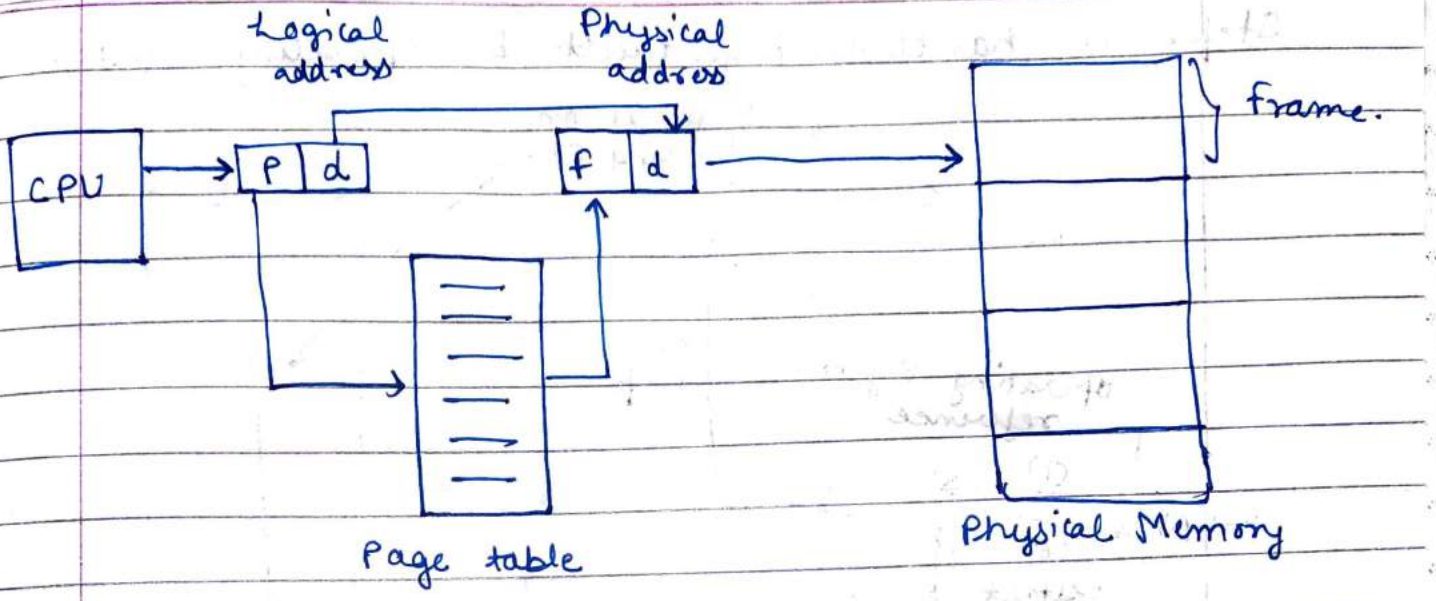
Total available: 1700 K Total used: 741 K.

Best fit algorithm makes most efficient use of memory, because in this, all the processes are allocated memory partition with minimum internal fragmentation.

Paging

Paging is a memory management scheme that eliminates the need for contiguous allocation of physical memory. This scheme permits the physical address space of a process to be non-contiguous. Paging avoids external fragmentation and the need for compaction.

- Logical address: An address generated by the CPU.
- Logical address space: The set of all logical addresses generated by a program.
- Physical address: An address actually available on memory unit.
- Physical address space: The set of all physical addresses corresponding to the logical addresses.



The mapping from virtual to physical address is done by memory management unit which is a hardware device and this mapping is known as paging technique.

- The physical address space is conceptually divided into a number of fixed-size blocks called frames.
- The logical address space is also splitted into fixed-size blocks called pages.

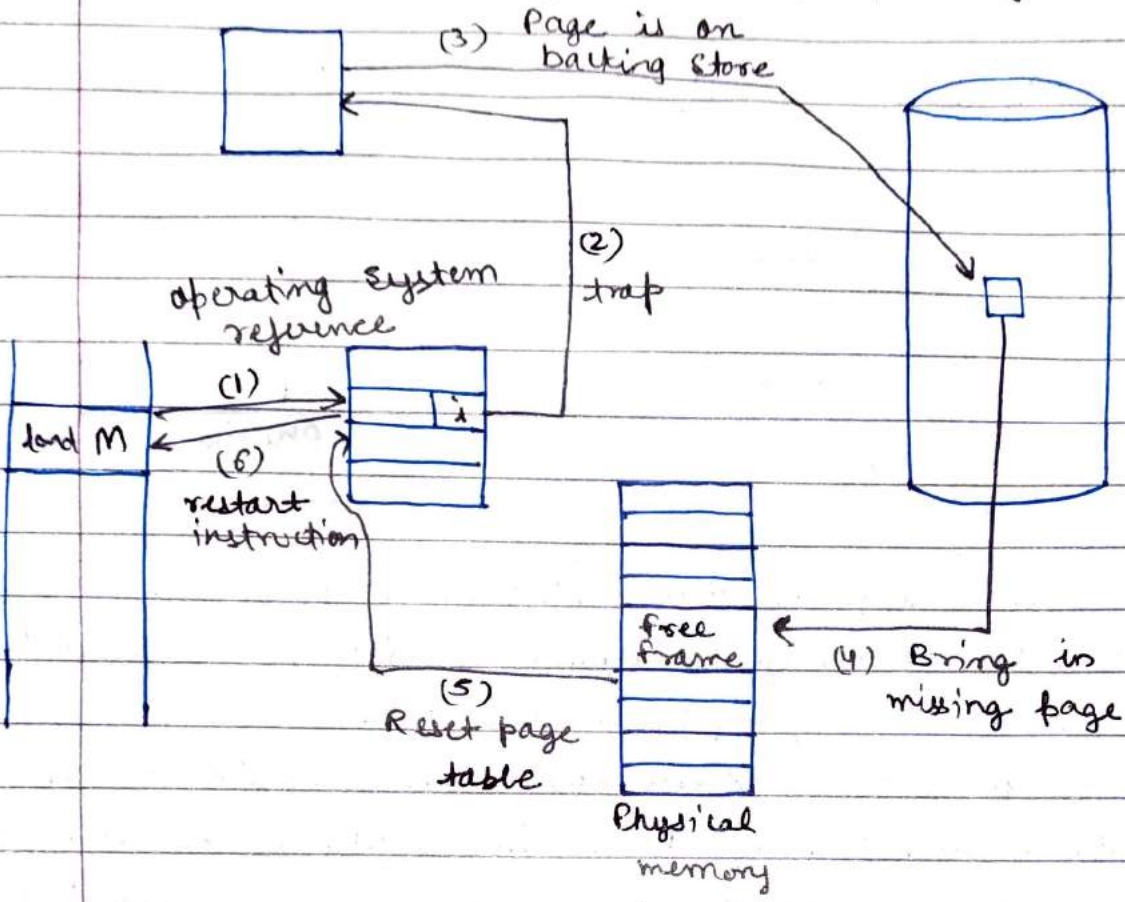
Page size = frame size.

Page fault or Page error:

An interrupt that occurs when a program requests data that is not currently in real memory. The interrupt triggers the OS to fetch the data from a virtual memory and load it into RAM. An invalid page fault or page error occurs when the OS cannot find the data in virtual memory.

A page fault occurs when an access to a page that has not been brought into main memory takes place.

Steps in handling page fault by operating system



Segmentation

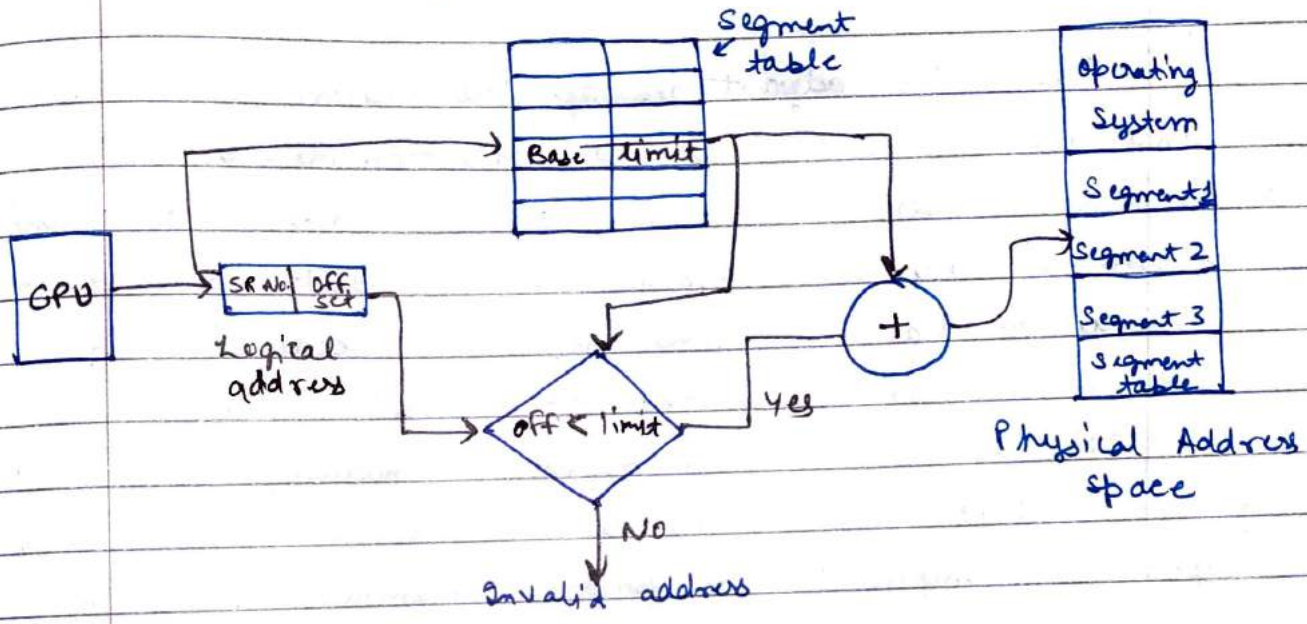
Segmentation is a memory-management scheme that supports the programmer view of memory. A logical address space is a collection of segments. Each segment has a name and a length.

The address specify both the segment name and the offset within the segment. The programmer therefore specifies each address by two quantities : a segment name and an offset.

For simplicity of implementation, segments are numbered and are referred to by segment number. Segment number indexes a process segment table. Each entry in segment table has a segment base and segment limit. Segment base contains the physical address where segment resides in memory, whereas segment limit specifies the length of segment.

Page No.
Date

Segmentation is a memory managment technique in which, the memory is divided into variable size parts. Each part is known as segment which can be allocated to a process. The details about each segment are stored in a table called segment table.



Segmentation	Paging
1) Program is divided into variable size segments.	1) Program is divided into fixed size pages.
2) User is responsible for dividing program into segments.	2) OS is responsible for dividing pages.
3) It is slower	3) It is faster.
4) It is visible to users.	4) It is not visible to user.
5) It eliminates internal fragmentation	5) It can't eliminates internal fragmentation.
6) It can't eliminates external fragmentation	6) It can eliminates external fragmentation.
7) OS maintains a list of free holes in memory.	7) OS maintains a list of free frame.

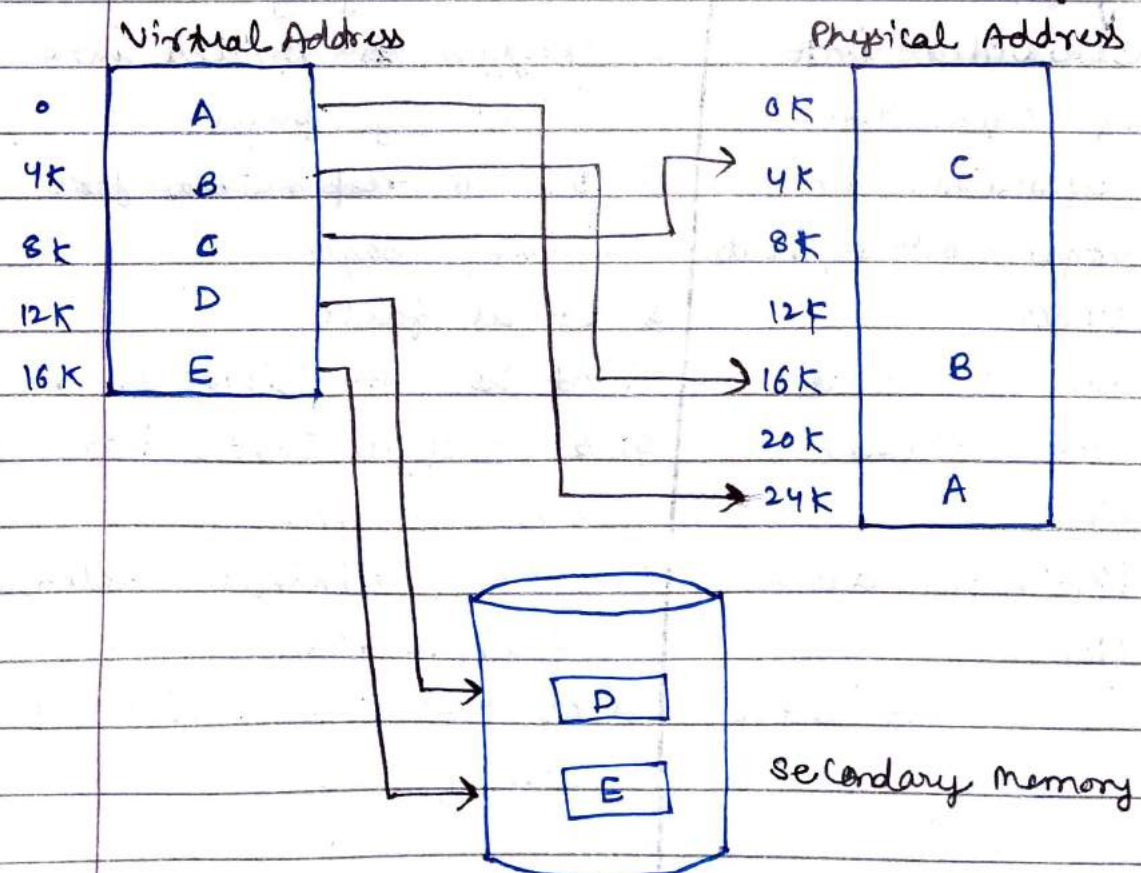
Page No.
Date

Virtual Memory

A computer can address more memory than the amount physically installed on the system. This extra memory is actually called virtual memory and it is a section of hard disk that's set up to emulate the computer's RAM.

The main advantage of this scheme is that programs can be larger than physical memory. Virtual memory serves two purposes. first, it allows us to extend the use of physical memory by using disk. Second, it allows us to have memory protection because each virtual address is translated to a physical address.

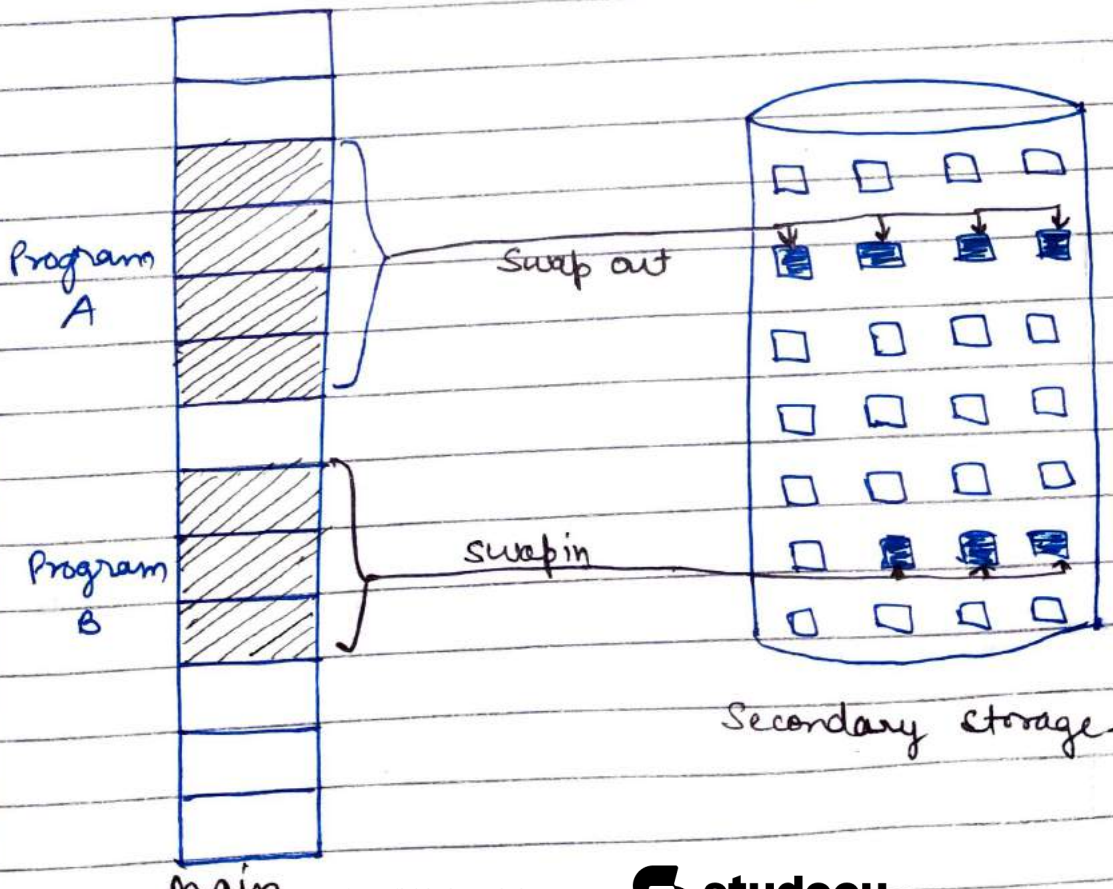
Virtual memory is a memory management capability of an OS that uses hardware and software to allow a computer to compensate for physical memory shortage by temporarily transferring data from RAM to disk storage.



Demand Paging

The process of loading the page into memory on demand whenever page fault occurs is known as demand paging.

Process: If CPU tries to refer a page that is currently not available in the main memory, it generates an interrupt indicating memory access fault. The OS puts interrupted process in a blocking state. For the execution to proceed the OS must brings the required page into the memory. The OS will search for the required page in the logical address space. The required page will be brought from logical address space to physical address space. The page table will updated automatically. The signal will be sent to the CPU to continue the program execution and it will place the process back into ready state.



Page Replacement Algorithm

(1) First in first Out (FIFO):

A FIFO replacement algorithm associates with each page, the time when that page was brought into memory. When a page must be replaced, the oldest page is chosen. we can create a FIFO queue to hold all pages in memory. we replace the page at the head of the queue. when a page is brought into memory, we insert it at the tail of the queue.

(2) Optimal Replacement Algorithm:

The optimal policy selects for replacement that page for which the time to the next reference is the longest. An optimal page replacement algorithm has the lowest page fault rate of all algorithms. This algorithm is impossible to implement because it would require the operating system to have perfect knowledge of future events.

(3) LRU page replacement:

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) algo.

Q. Consider the following reference string
1, 2, 3, 4, 2, 1, 5, 6, 2, 1, 2, 3, 7, 6, 3, 2, 1, 2, 3, 6

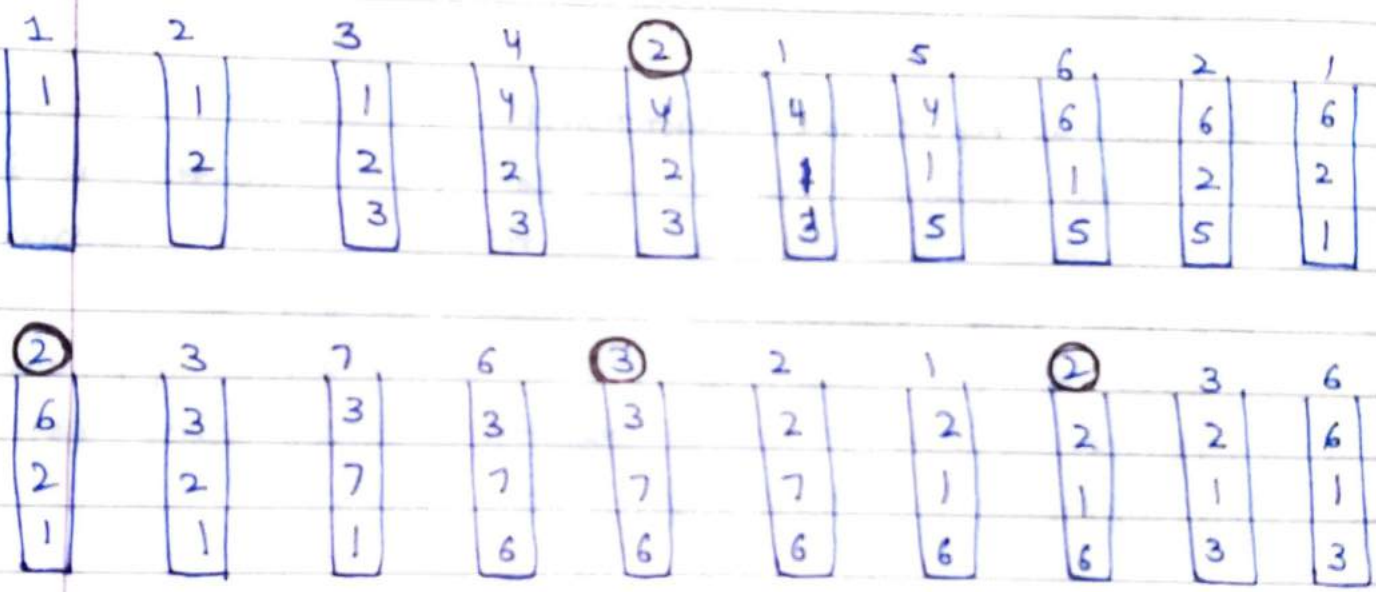
i) FIFO

ii) LRU page replacement algorithm?

Assuming three and four frames in each case and frames are initially empty.

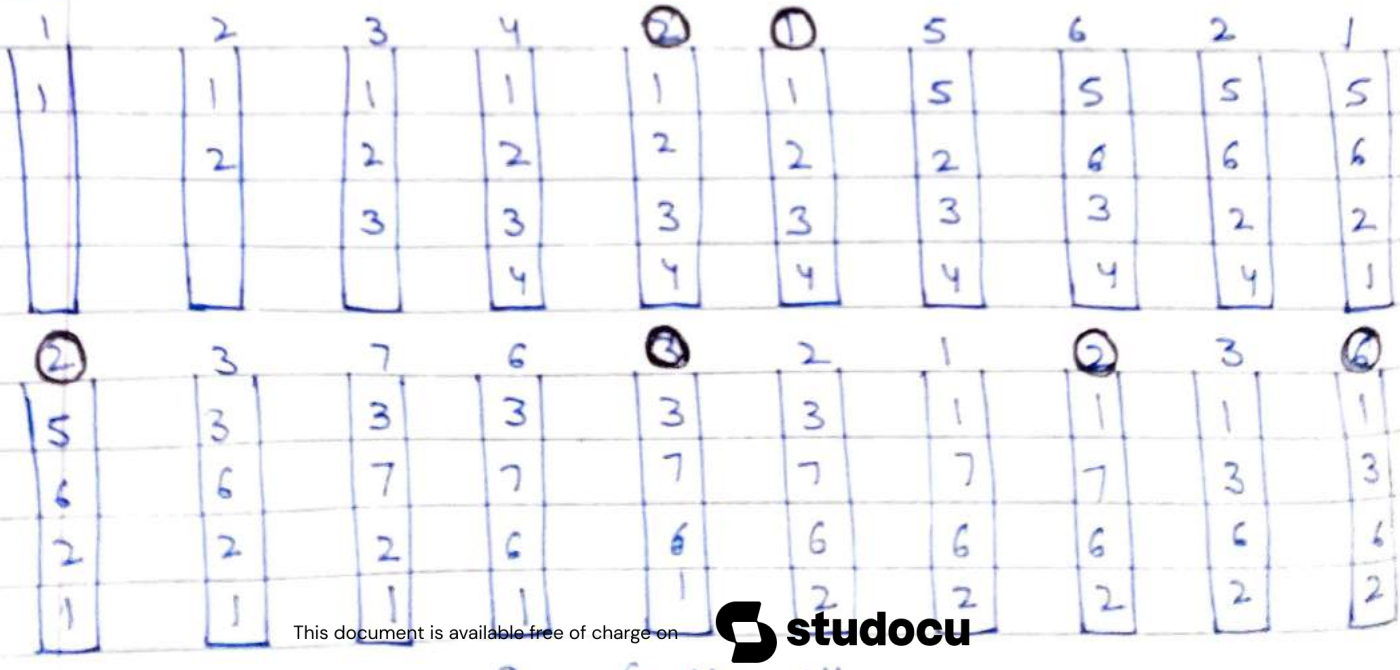
Sol. i) FIFO

3 frame



Page fault = 16

4 frame.



ii) LRU page replacement algorithm.

3 frame:

1	2	3	4	(2)	1	5	6	2	1
1	1	1	4	4	4	5	5	5	1
	2	2	2	2	2	2	6	6	6
		3	3	3	1	1	1	2	2

(2)	3	7	6	(3)	2	1	(2)	(3)	6
1	1	7	7	7	2	2	2	2	2
6	3	3	3	3	3	3	3	3	3
2	2	2	6	6	6	1	1	1	6

Page faults = 15.

4 frame:

1	2	3	4	(2)	(1)	5	6	(2)	(1)
1	1	1	1	1	1	1	1	1	1
	2	2	2	2	2	2	2	2	2
		3	3	3	3	5	5	5	5
			4	4	4	4	6	6	6

(2)	3	7	(6)	(3)	(2)	1	(2)	(3)	(6)
1	1	1	6	6	6	6	6	6	6
2	2	2	2	2	2	2	2	2	2
5	3	3	3	3	3	3	3	3	3
6	6	7	7	7	7	1	1	1	1

Page fault = 10.

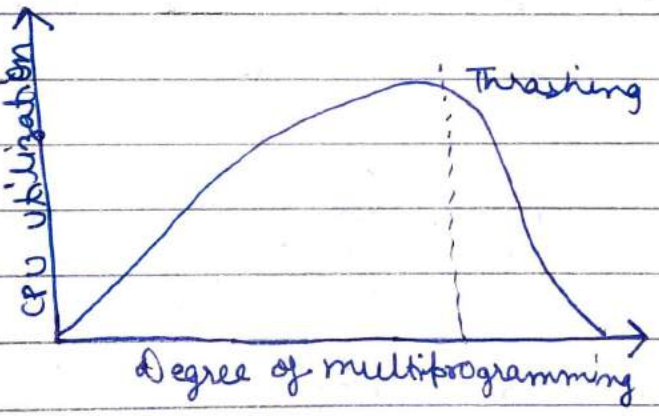
Thrashing:

Thrashing occurs when a system spends more time in processing page faults rather than executing instructions.

Thrashing is a computer activity that makes little progress. Usually this happen due to limited resources or exhaustion of memory. It arises when a page fault occurs. Page fault arises when memory access of virtual memory space does not map to content of RAM.

The effect of thrashing is:

1. CPU becomes idle.
2. Decreasing the utilization increases the degree of multiprogramming



A process is thrashing if it is spending more time in paging than execution. Thrashing is discrete phenomenon. Thrashing is busy swapping pages in and out or doing more paging then executing.

Unit - 5

I/O Management & Disk Scheduling

Page No. _____
Date

I/O devices are the collection of interfaces that different functional units of an information processing system use to communicate with each other.

Inputs are the signals received by the unit and outputs are the signals sent from it. I/O devices are used by a person to communicate with a computer

Categories of I/O devices are:

1) Human readable:

It is suitable for communicating with the computer user. Ex- Printers, Keyboard

2) Machine Readable:

It is suitable for communicating with electronic equipment. Ex- Disk and tape driver, sensors.

3) Communication:

It is suitable for communicating with remote devices. Ex- Digital drivers, modems.

Techniques / Organization of I/O function

1) Programmed I/O:

Programmed I/O takes place under direct control of CPU. CPU issues a command on behalf of a process to an I/O module. CPU then waits for the operation to be completed before proceeding.

2) Interrupt driven I/O:

In this, CPU issues a command on behalf of a process to an I/O module. If the I/O instruction is non-blocking, CPU continues to execute next instruction from the same process.

If the I/O instruction is blocking, OS takes over suspends the current process and assigns CPU to another process

3) Direct Memory Access (DMA):

DMA module control exchange of data between memory and an I/O module. CPU sends a request to transfer a block of data to the DMA module and is interrupted only after the entire block has been transferred.

Buffer

A buffer is a region of memory used to temporarily hold data while it is being moved from one place to another.

A buffer is a memory area that stores data being transferred between two devices or between a device and application. Buffering is done to cope with a speed mismatch between the producer and consumer of a data stream.

Buffering provide adaptations for devices that have different data transfer sizes.

I/O buffering techniques:

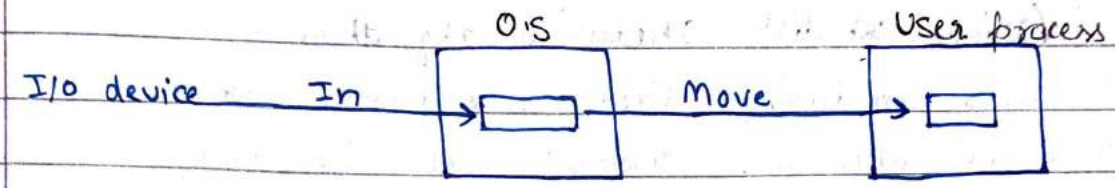
1. Single buffering

When a user process issues an I/O request, the OS assigns a buffer in the system portion of main memory to the operation.

The technique can be used as follows:

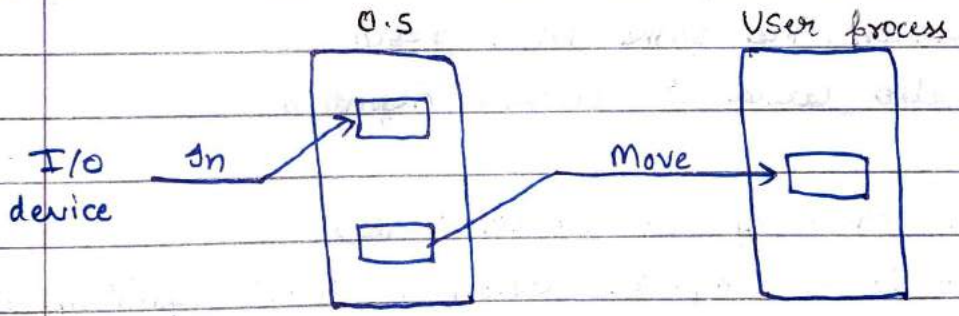
Input transfers are made to the system buffer. When the transfer is complete, the process moves the block into user space and request another block. This is called

reading ahead, it is done in the expectation that the block will be needed sometimes in future. This approach will generally provide a speed up compared to the lack of system buffering. The O.S must keep track of the assignment of the system buffers to user processes.



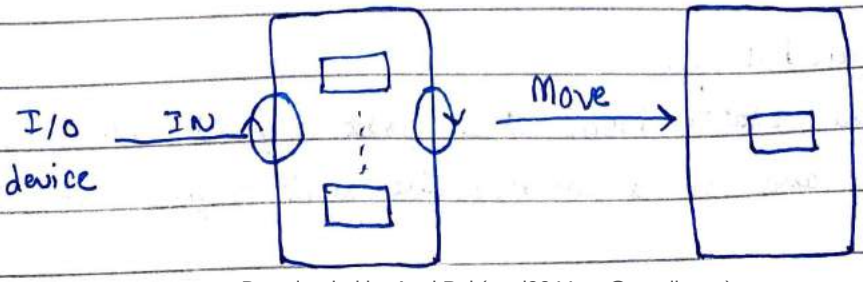
2. Double buffering

A programming technique that uses two buffers to speed up a computer that can overlap I/O with processing. Data in one buffer are being processed while the next set of data is read into the other one.



3. Circular buffering

When more than two buffers are used, the collection of buffers is itself called circular buffer, with each buffer being one unit in the circular buffer.



Services provided in Kernel I/O Subsystem

1) I/O scheduling
2) Buffering
3) Caching
4) Spooling
5) Error handling

Various disk scheduling algorithms:

1. FCFS scheduling (first Come first Serve)
2. SSTF scheduling (Shortest Seek Time first)
3. SCAN scheduling

As the name suggests, this algorithm scans all the cylinders of the disk back and forth. Head starts from one end of the disk and move towards the other end servicing all the requests in between. After reaching the other end, head reverse its direction and move towards the starting end servicing all the requests in between. The same process repeats.
It is also called as Elevator Algorithm.

4. C-SCAN scheduling (Circular - SCAN)

It is an improved version of SCAN algorithm. Head starts from one end of the disk and move towards the other end servicing all the requests in between. After reaching the other end, head reverse its direction. It then returns to the starting end without servicing any request in between. The same process repeats.

5. LOOK Scheduling:

Head starts from the first request at one end of the disk and moves towards the last request at the other

end servicing all the requests in between. After reaching the last request at the other end, head reverse its direction. It then returns to the first request at the starting end servicing all the requests in between. The same process repeats.

Numericals:

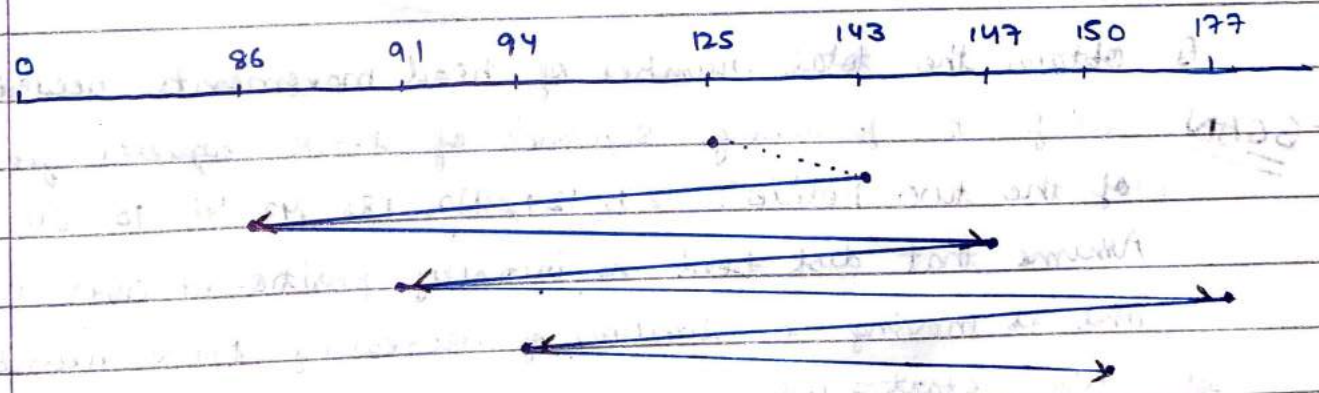
(1) FCFS scheduling

Q. Suppose the moving head disk with 200 tracks is currently serving a request for track 143 and has just finished a request for track 125. If the queue of request is kept in FIFO order 86, 147, 91, 177, 94, 150. What is total head movement for the following scheduling:

- FCFS

Sol. queue = 86, 147, 91, 177, 94, 150

head start = 143

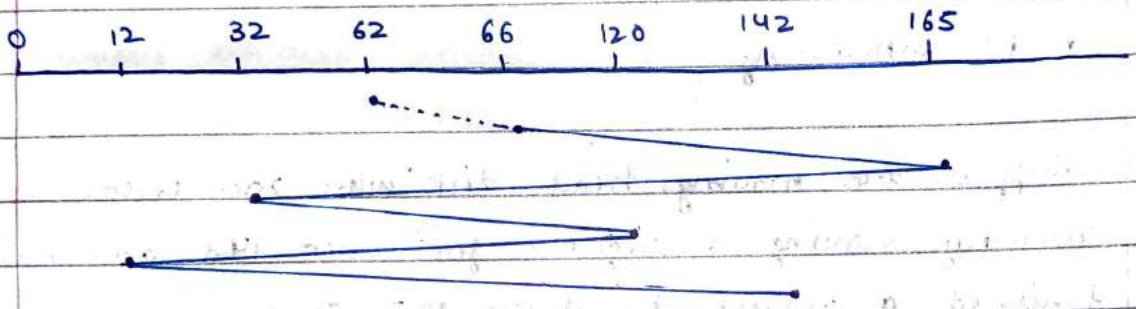


Total distance = (143-86) + (147-86) + (147-91) + (177-91) + (177-94) + (150-94)

= 399.

Q. Suppose the head of moving disk is currently servicing a request at track 62. Consider a process requesting to read from the following tracks:
66, 165, 32, 120, 12, 142.

Sol. Queue = 66, 165, 32, 120, 12, 142
head start = 62

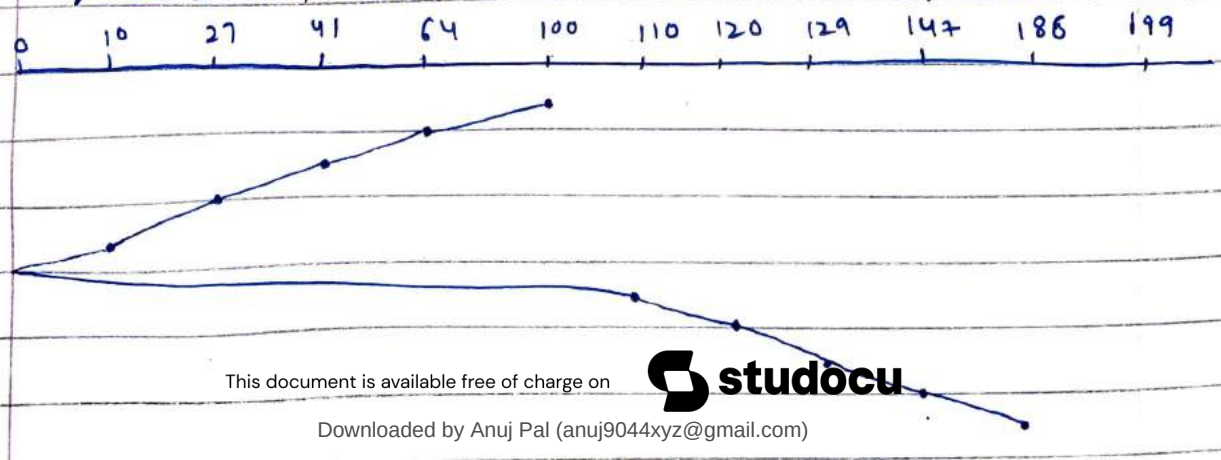


Total head movement = (66-62) + (165-66) + (165-32) + (120-32) + (120-12) + (142-12)
= 562.

(2) SSTF scheduling

Q. Obtain the total number of head movements needed to satisfy the following sequence of track requests for each of the two policies: 27, 129, 110, 186, 147, 41, 10, 64, 120. Assume that disk head is initially positioned over track 100 and is moving in direction of decreasing track number.

Sol. head start = 100
queue = 27, 129, 110, 186, 147, 41, 10, 64, 120



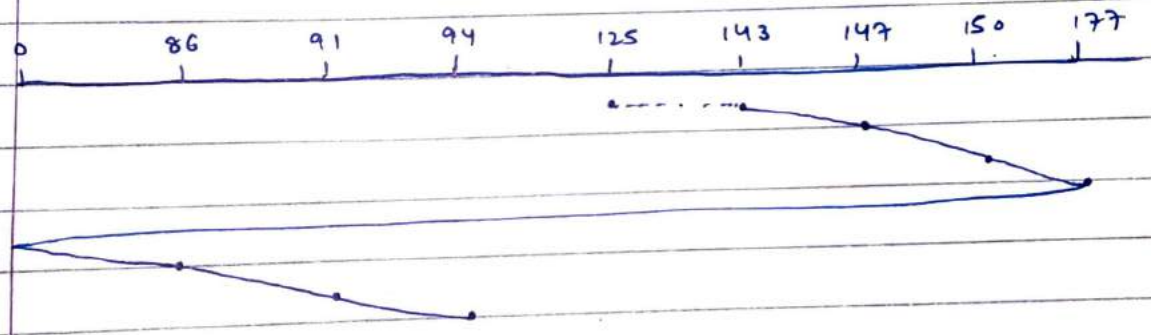
Total head movement = (100-64) + (64-41) + (41-27) + (27-10)
+ (10-0) + (110-0) + (120-110) + (129-120)
+ (147-129) + (186-147)
= 286.

Q. Suppose the moving head disk with 200 tracks is currently serving a request for track 143 and has just finished a request for track 125. If the queue of request is kept in FIFO order 86, 147, 91, 177, 94, 150. what is total head movement for ~~SSTF~~ scheduling?

C-SCAN

Sol. head start = 143

queue = 86, 147, 91, 177, 94, 150

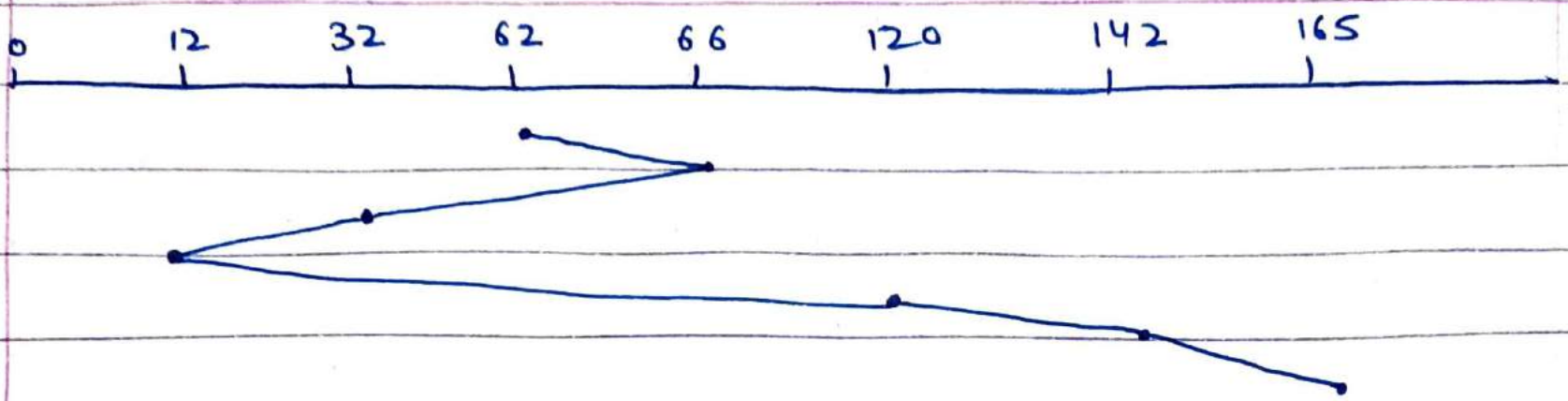


total head movement = (147 - 143) + (150 - 147) + (177 - 150) +
(177 - 0) + (86 - 0) + (91 - 86) + (94 - 91)
= 305.

Q. Suppose the head of moving head disk is currently servicing a request at track 62. Consider a process requesting to read from the following tracks: 66, 165, 32, 120, 12, 142. with SSTF.

Sol. head start = 62

queue = 66, 165, 32, 120, 12, 142.



total head movement = (66-62) + (66-32) + (32-12) + (120-12) + (142-120) + (165-142)

= 212.

(3) SCAN Scheduling

Boot Blocks

Operating system requires some information while booting. If the disk is divided into number of partitions, the operating system is stored in the first partition of the disk. If the disk does not contain an OS, this block can be empty. This is called boot block.

Bit Vector:

The free space list is implemented as a bit map or bit vector. Each block is represented by 1 bit. If the block is free, the bit is 1; if the block is allocated, the bit is 0.

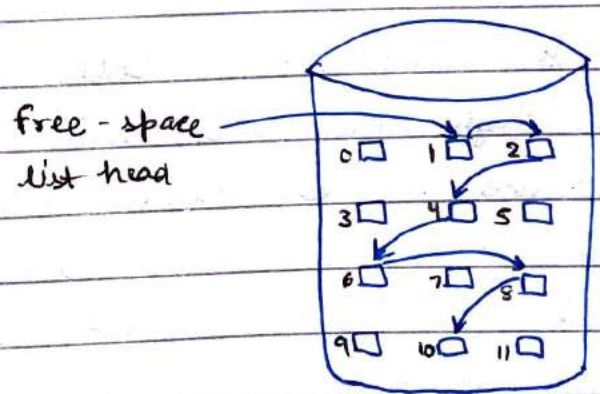
for ex, consider a disk where blocks 2, 3, 4, 5, 8, 9, 10, 11, 12, 13, 17, 18, 25, 26 and 27 are free and rest of the blocks are allocated.

The free-space bit map would be:

00111100111111000110000001110000 ....

Linked list:

Another approach to free-space management is to link together all the free disk blocks, keeping a point to the first free block in a special location on the disk and caching it in memory. This first block contains a pointer to the next free disk blocks and so on.



Grouping:

A modification of free list approach stores the addresses of n free blocks in the first free block. The first n-1 of these blocks is actually free. The last block contains the addresses of another n free block and so on. The addresses of a large number of free blocks can now be found quickly.

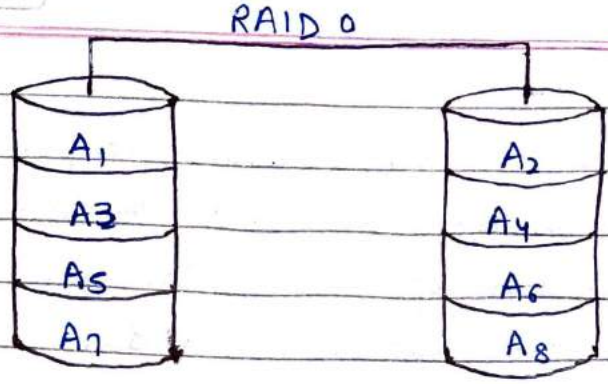
RAID

RAID stands for Redundant Array of Inexpensive disks which was later interpreted to Redundant Array of Independent Disks. This technology is now used in almost all the IT organizations looking for data redundancy and better performance. It combines multiple available disks into 1 or more logical drive and gives you the ability to survive one or more drive failures depending upon the RAID level used.

With the increasing demand in the storage and data world wide the prime concern for the organization is moving towards the security of their data. Here term security does not mean security from vulnerable attacks rather than from hard disk failure and any such relevant accidents which can lead to destruction of data.

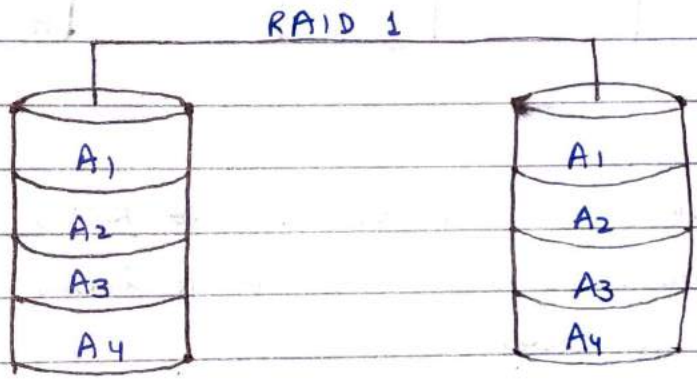
1) RAID 0

This level strips the data into multiple available drives equally giving a very high read and write performance but offering no fault tolerance or redundancy. This level does not provides any of the RAID factor and cannot be considered in an organization looking for redundancy instead it is preferred where high performance is required.



2) RAID 1

This level performs mirroring of data in drive 1 to drive 2. It offers 100% redundancy as array will continue to work even if either disk fails. So organization looking for better redundancy can opt for this solution but again cost can become a factor.

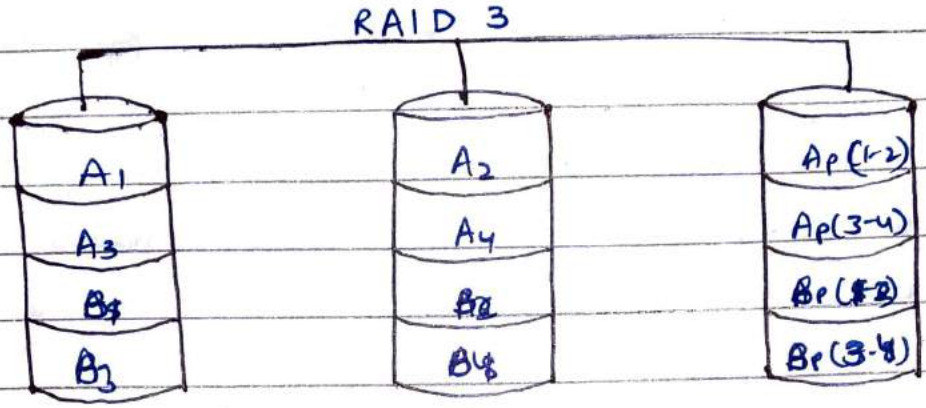


3) RAID 2

This level uses bit-level data stripping rather than block level. To be able to use RAID 2 make sure the disk selected has no self-disk error checking mechanism as this level uses external Hamming code for error detection. This is one of the reason RAID is not in the existence in real IT world as most of the disks used these days come with self error detection. It uses an extra disk for storing all the parity information.

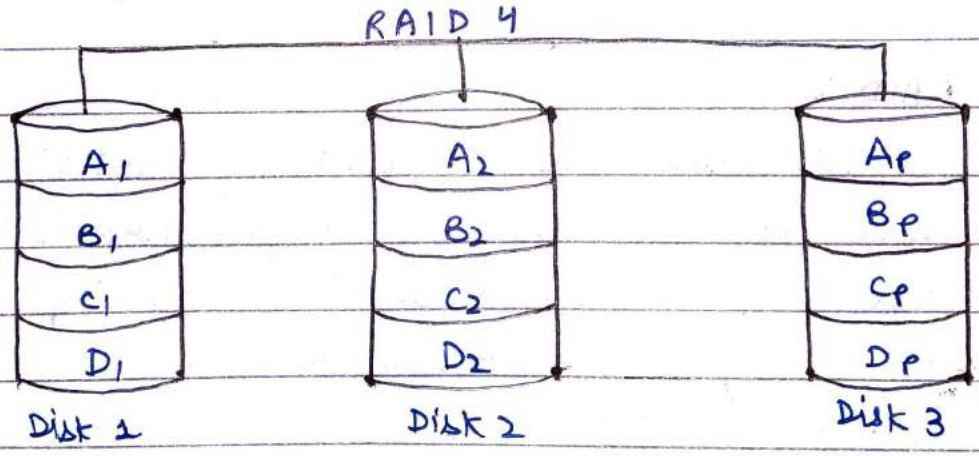
4) RAID 3

This level uses byte level stripping along with parity. One dedicated drive is used to store the parity information and in case of any drive failure the parity is restored using this extra drive. But in case the parity drives crashes then the redundancy gets affected again so not much considered in organizations.



5) RAID 4

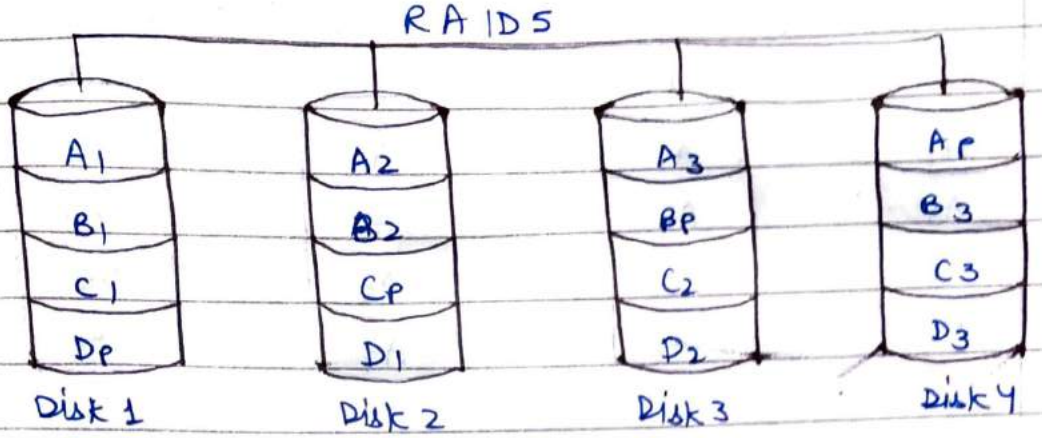
This level is very similar to RAID 3 apart from the feature where RAID 4 uses block level stripping rather than byte level.



6) RAID 5

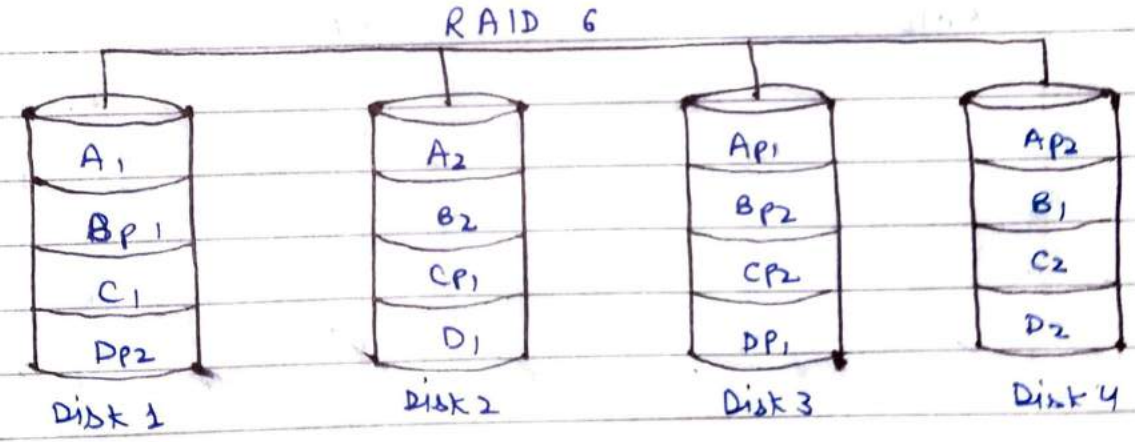
It uses block level striping and with this level distributed parity concept came into the picture leaving behind the traditional dedicated parity as used in RAID 3 and RAID 4. Parity information is written to a different disk in the array for each stripe. In case of single disk failure

data can be recovered with the help of distributed parity without affecting the operation and other read write operations.



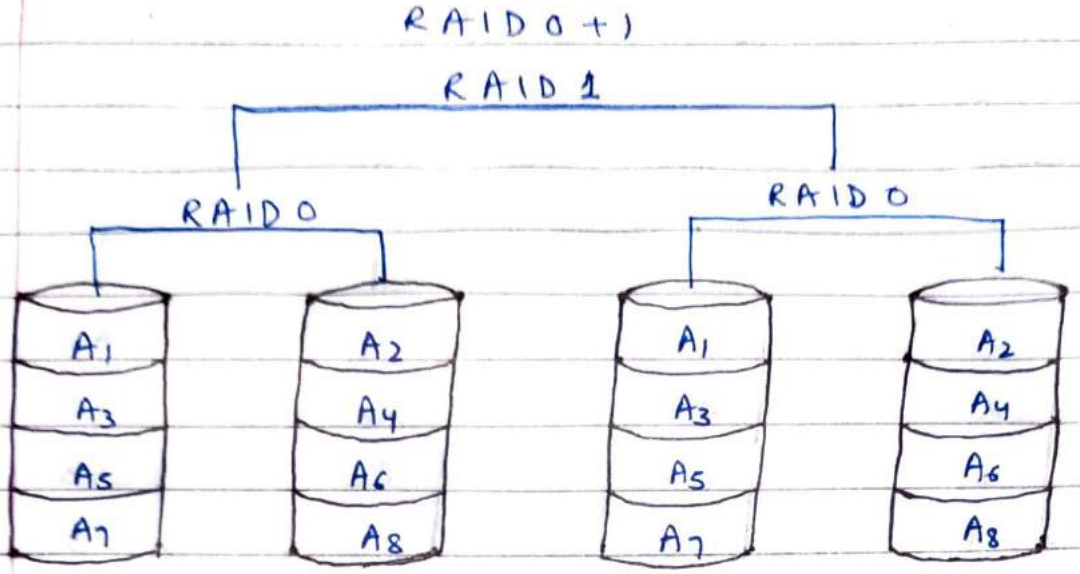
7) RAID 6

This level is an enhancement version of RAID 5 adding extra benefit of dual parity. This level user blocks level stripping with DUAL distributed parity. So now you can get extra redundancy.



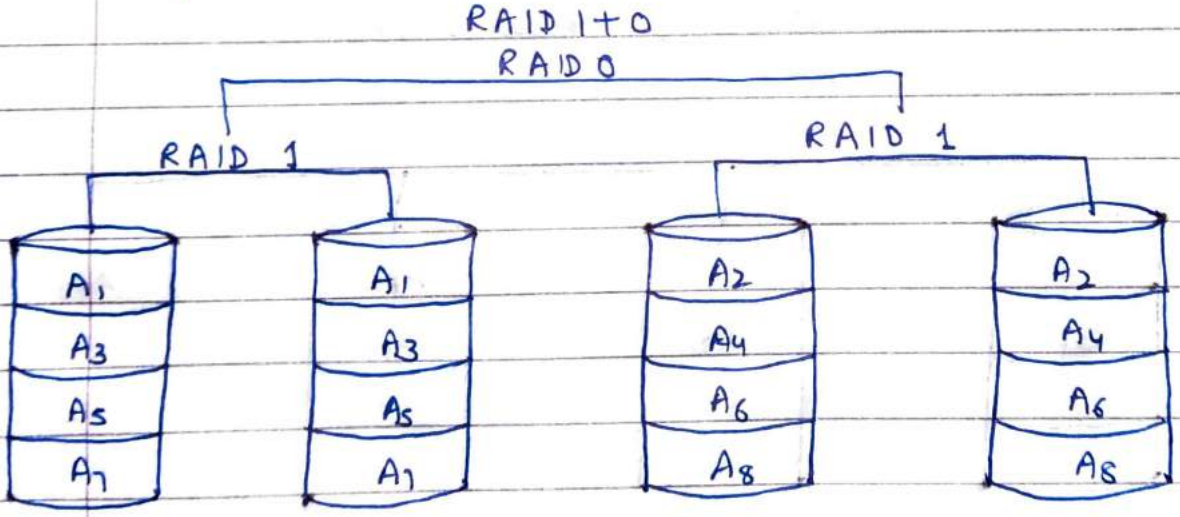
8) RAID 0+1

This level uses RAID 0 and RAID 1 for providing redundancy. Stripping of data is performed before mirroring. In this level, the overall capacity of usable drives is reduced as compared to other RAID levels.



9) RAID 1+0

This level performs mirroring of data prior striping which makes it more efficient and redundant as compared to RAID 0+1. This level can survive multiple simultaneous drive failures.



File Systems

File systems permits users to created data collections. called files with desirable properties:

1) Long term existence:

Files are stored on disk or other secondary storage and do not disappear when a users logs off.

2) Sharable between processes:

Files have names and can have associated access permissions that permit controlled sharing.

3) Structure:

Depending on file systems, a file can have an internal structure that is convenient for particular applications.

File is a collection of data created by user. It provides a means to store data organized as file as well as a collection of function that can be performed on file.

Operations performed on file systems:

1. Create
2. Delete
3. Open
4. Close
5. Read
6. write

File contains various attributes:

1. Name of the file
2. Identifier
3. Type
4. Location
5. Size
6. Protection
7. Date & time

File Organization Methods:

File organization refers to the way, data is stored in a file. File organization is very important because it determines the methods of access, efficiency, flexibility and storage devices to use.

There are four methods of organizing files:

1. Sequential file organization

Records are stored and accessed in a particular sorted order using a key field. Retrival requires searching sequentially through the entire file record by record to the end.

2. Random file organization

Records are stored randomly but accessed directly. To access a file which is stored randomly, a record key is used to determine where a record is stored on the storage media.

3. Serial file Organization

Records in a file are stored and accessed one after another.

4. Indexed- Sequential file organization

Similar to sequential method only that, an index is used to enable the computer to locate individual records on the storage media.

Access Mechanisms:

1. Sequential Access
2. Direct Access
3. Index Access

Directory

The directory can be viewed as a symbol table that translates file names into their directory entries. We can see that the directory itself can be organized in many ways. The organization must be allowed to insert entries, to delete entries, to search for a named entry.

Directory is implemented in two ways:

1. Linear list

It uses a linear list of file names with pointers to the data blocks. Linear list uses a linear search to find a particular entry. Simple for programming but time consuming to execute

2. Hash table

Hash table decreases the directory search time. Insertion and deletion are also straightforward. Hash table takes the value computed from the file name. Then it returns a pointer to the file name in the linear list. Hash table uses fixed size.

Operations performed on a directory are:

1. Searching a file
2. Create a file
3. Delete a file
4. Rename a file
5. List directory