PART-A

Q1.1] Define and explain Operating System with examples.

Ans An Operating System is a collection of software that manages computer hardware resources and provides common services for computer programs. It also can be defined as an interface between computer user and computer hardware.
eg:- Microsoft Windows, macOS, Linux, Android, iOS, etc.

Q1.2] Write down the function of OS.

Ans The function of OS are:-
1] Process Management
2] Memory Management
3] Device Management
4] File Management
5] Security
6] Error Detecting Aids
7] Job Accounting
8] Control Over System Performance
9] Coordination between Other Softwares and Users.

Q1.3] Explain: i] Symmetric Multiprocessor
ii] Asymmetric Multiprocessor

Ans i] In these types of systems, each processor contains a similar copy of the operating system and they all communicate with each other. All the processors are in a peer to peer relationship.

ii] In asymmetric systems, each processor is given a predefined task. There is a master processor that gives instruction to all the other processors. Asymmetric multiprocessor system contains a master slave relationship.

Q1.4] Why Layering is done in Operating Systems?

Ans The Layered structure breaks up the operating system into different Layers and retains much more control on the system
It is done to:
- To separately define layers.
- To interact with each other as per respective requirement.
- To easily create maintain, and update the system
- It is done, so that change in one layer will not affect other layers.
- Each layer can interact with above and below levels.

Q1.5 What are the different algorithms used in scheduling processes?

Ans The different algorithms used for process scheduling are FCFS (First come, first served), SJF (Shortest job first), priority scheduling, round-robin scheduling, etc.

PART-B

Q 2.1] Explain any two types of OS.

Ans The two types of Operating System are as follows.

i] Batch Operating System

The users who using a batch operating system do not interact with the computer directly. Each user prepares its job on an off-line device like punch cards and submits it to the Computer operator. To speed up the processing, jobs with similar needs are batched together and run as a group. The programmers exit their programs with the operator and the operator then sorts the programs with similar requirements into batches.

The problems that occurs with Batch Systems are as follows:

→ There is a lack of interaction between the user and the job.

→ CPU is being often idle, because the speed of the mechanical I/O devices is slower than the CPU.

→ It is difficult to provide the desired priority.

ii] Multiprogramming Operating System

In Multiprogramming OS, several processes are all loaded into memory and available to run. Whenever a process initiates an I/O operation, the Kernel selects a different process to run on the CPU. This approach allows the Kernel to keep the CPU active and performing work as much as possible, thereby reducing the amount of wasted time. By reducing this waste, multiprogramming allows all programs to finish sooner than they would otherwise.

Multiprogramming OS is known to be working on Non-Preemptive Scheduling (Non-Preemptive Scheduling is the scheduling technique in which the CPU is allocated to a process and hold by it till the process gets terminated.

Q2.2] Differentiate: Monolithic Kernel and MicroKernel with respective layouts.

Ans

Monolithic Kernel	MicroKernel
1] In monolithic Kernel, both User Services and Kernel Services are Kept in the same address space.	1] In microkernel, User Service and Kernel Services are Kept in seperate address Space.
2] It is not easy to extend monolithic Kernel	2] It is easy to extend MicroKernel.
3] Monolithic Kernel is larger than microKernel	3] MicroKernel are smaller in size.
4] Faster Execution	4] Slower Execution
5] Failure of one Component in a monolithic Kernel leads to the failure of the entire System	5] Failure of one component does not affect the working of the system.
6] Easy implementation	6] Complex implementation
7] Fewer lines of code need to be written in monolithic Kernel	7] More lines of Code is need to be written for a micro Kernel

8] Difficult to add new functionalities	8] Easier to add new functionalities.
9] Debugging and Management are complex	9] Debugging and Management are straightforward
10] Example:- UNIX, LINUX	10] Example:- MacOS, L4 LINUX

Q 2.3] Explain RTOS and its types in detail with example.

Ans. A real-time operating system (RTOS) is an operating system (OS) for real-time computing applications that processes data and events that have critically defined time constraints. Real-time operating system is a time crucial operating system. This means that the response to any event must come in specified time interval only. Hence, a delay in response will result in disastrous effects.

Types of Real-time Operating Systems:

1. Soft Real-time OS:

Soft real-time systems are less attractive. A critical real-time task gets priority over other tasks and retains the priority until.

it Completes. Soft real-time systems have limited utility than hard real-time systems. The Soft RTOS is a system in which the deadline for certain tasks can be delayed to some extent. For example, if the task deadline is 1:20:30 PM, then the task can on occasions complete at let us say 1:20:35 PM every. However, it cannot delay for too long say 1:30 PM.
For example, multimedia, virtual reality, Advanced Scientific Projects. like Undersea exploration and planetary rovers, etc.

2. Hard Real Time OS
Hard real-time systems guarantee that critical tasks complete on time. In hard time systems, secondary storage is limited or missing and the data is stored in ROM. In these Systems, virtual memory is almost never found. The Hard RTOS is a system which meets the deadline for every process at all times. For example, if the task deadline is 1:20:30PM then the task has to be ~~done~~ Complete before 1:20:30 PM every time.
For example: Missile launching systems, Aeroplane launching system, Traffic Control System, etc.

PART-C

Q.3.1] Explain Components of OS in detail.

Ans The components of OS are as follows:

1] Process Management
A process is program or a fraction of a program that is loaded in main memory. A process need certain resources including CPU time, Memory, Files, and I/O devices to accomplish its task. The process management component manages the multiple processes running simultaneously on the Operating System. A program in running state is called a process.

2] File Management
File management is one of the most visible services of an operating system. Computers can store information in several different physical forms; magnetic tape, disk, and drum are the most common forms. A file is derived as a set of correlated information and it is defined by the creator of the file. Mostly files represent data, source and object forms, and programs. Data files can be any type alphabetic, number, and alphanumeric.

3] Network Management
The definition of network management is often broad, as network management involves several

different components. Network management is the process of managing and administering a computer network. A computer network is a collection of various types of computer connected with each other. Network management comprises fault analysis, maintaining the quality of service, provisioning of network, and performance management.

4] Main Memory Management

Memory is a large array of words or bytes, each with its own address. It is a repository of quickly accessible data shared by the CPU and I/O devices. Main memory is a volatile storage device which means it loses its contents in the case of system failure or as soon as system power goes down. The main motivation behind Memory Management is to maximize memory utilization on the computer system.

5] Secondary Storage Management

The main purpose of a computer system is to execute programs. These programs, together with the data they access, must be in main memory during execution. Since the main memory is too small to permanently accommodate all data and program, the computer system must provide secondary

Storage to backup main memory.

6] I/O Device Management

One of the purpose of an operating System is to hide the peculiarities of specific hardware devices from the user. I/O Device Management provides an abstract level of H/W devices and keep the details from applications to ensure proper use of devices, to prevent errors, and to provide users with convenient and efficient programming environment.

7] Security Management

The operating System is primarily responsible for all task and activities happen in the Computer System. The various processes in an operating system must be protected from each other's activities. For that purpose, various mechanisms which can be used to ensure that the files, memory segment, cpu and other resources can be operated on only by those processes that have gained proper authorization from the operating system.

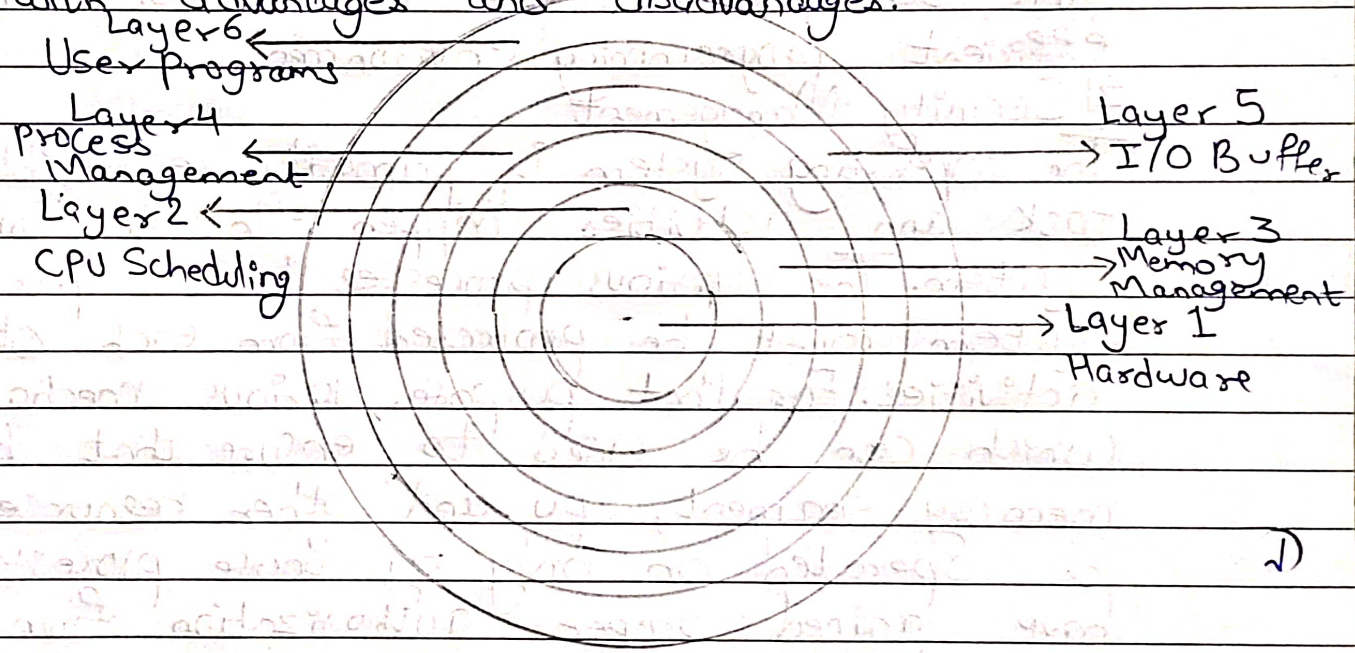
8] Command Interpreter System

One of the most important component of an operating system is its command interpreter. The command interpreter is the primary interface between the user and the rest of the System. Command Interpreter System executes a user command by calling one or more

number of underlying system programs or System calls.
Command Interpreter System allows human users to interact with the Operating System and provides convenient programming environment to the users.

Q3.2] Draw and explain layered structure of OS with advantages and disadvantages.

Ans.



The Layered Structure of OS are as follows:
1. Hardware: This Layer interact with the system hardware and coordinates with all peripheral devices used, such as a printer, mouse, keyboard, scanner, etc. These types of hardware devices are managed in the hardware layer. The hardware layer is the lowest and most authoritative layer in

the layered operating system architecture. It is attached directly to the core of the system.

2. CPU Scheduling: This layer deals with Scheduling the processes for the CPU. Many Scheduling queues are used to handle processes. When the processes enter the system, they are put into the job queue. The processes that are ready to execute in the main memory are kept in the ready queue. This layer is responsible for managing how many will stay out of the CPU

3. Memory Management: Memory management deals with memory and moving processes from disk to primary memory for execution and back again. This is handled by the third layer of the operating System. All memory management is associated with this layer. There are various types of memories in the computer like RAM, ROM. If you consider RAM, then it is concerned with Swaping in and Swapping out of memory. When our computer runs, some processes move to the main memory (RAM) for execution, and when programs, such as calculator, exit, it is removed from the main memory.

4. Process Management: This layer is responsible for managing the processes, i.e. assigning the processor to a process and deciding how many many processes is also managed in this

layer. The different algorithms used for process scheduling are FCFS (first come, first served), SJF (shortest job first), priority scheduling, round-robin scheduling, etc.

5. I/O Buffer: I/O devices are very important in computer systems. They provide users with the means of interacting with the system. This layer handles the buffer for the I/O devices and makes sure that they work correctly. Suppose you are typing from the keyboard. There is a keyboard buffer attached with the keyboard, which stores data for a temporary time. Similarly, all input/output devices have some buffer attached to them. This is because the input/output devices have slow processing or storing speed. The computer uses buffers to maintain the good timing speed of the processor and input/output devices.

6. User Programs: This is the highest layer in the layered operating system. This layer deals with the many user programs and applications that run in an operating system, such as word processors, games, browsers, etc. You can also call this an application layer because it is concerned with application programs.

Advantages Of Layered Structure.

There are several advantages of the layered structure of operating system design, such as:

1] Modularity: This design promotes modularity as each layer performs only the tasks it is scheduled to perform.

2] Easy debugging: As the layers are discrete so it is very easy to debug. Suppose an error occur in the CPU scheduling layer. The developer can only search that particular layer to debug, unlike the Monolithic System where all the services are present.

3] Easy update: A modification made in a particular layer will not affect the other layers.

4] No direct access to hardware: The hardware layer is the innermost layer present in the design. So a user can use the service of hardware but cannot directly modify or access it, unlike the Simple System in which the user had direct access to the hardware.

5] Abstraction: Every Layer is concerned with its function. So the functions and implementations of the other layers are abstract to it.

Disadvantages of Layered Structure.

Though this system has several advantages over the Monolithic and simple design, there

are also some disadvantages, such as:

1.] Complex and careful implementation: As a layer can access the services of the layers below it, so the arrangement of the layers ~~uses the~~ must be done carefully. For example, the backing storage layer uses the services of the memory management layer. So it must be kept below the memory management layer. Thus with great modularity comes complex implementation.

2.] Slower in execution: If a layer wants to interact with another layer, it requests to travel through all the layers present between the two interacting layers. Thus it increases response time, unlike the Monolithic system, which is faster than this. Thus an increase in the number of layers may lead to a very inefficient design.

3.] Functionality: It is not always possible to divide the functionalities. Many times, they are interrelated and can't be seperated.

4.] Communication: No communication between non-adjacent layers.

Q3.3] Explain in detail, The services provided by OS.

Ans The Services provided by OS are as follows:

1] User Interface.

The User interface is the Service that Practically enables users to interact with an operating System. So, it means interacting with the entire Computer System itself.

The users interface of operating system has two common forms: the Character User Interface (CUI) also known as Command line interface (CLI) and the Graphical User Interface (GUI).

2] Program Execution.

Another vital service of the operating system is program execution. Executing a program is not a simple task. To do that, operating systems tackle a myriad of operations, such as program loading, management of execution stacks, and process scheduling.

3] I/O Operations

A user can not directly control Input/Output (I/O) devices, such as monitors, speakers, keyboards, and mouse. So, operating systems mediate the communication between the user, I/O devices, and the Computer System.

4] File System Manipulation

The file system keeps and organizes all the files of a computer in persistent memory, usually a

Hard Disk Drive (HDD) or Solide-State Drive (SSD). However, it is normal for the user to create, modify, delete, and search data files in the file system. Thus, the operating system provides a service for the users to manipulate the file system, enabling them to execute the cited management operations.

5] Communications

In our particular scenario, the communication regards the data exchange between processes. These processes, in turn, may execute on the same computer or different computer. So, in the latter case, different computer communicate with each other through a network.

The operating system has specific mechanisms to enable processes running in the same computer to communicate. The most common examples are pipes and shared memory.

6] Error Detection

During a Computer system lifecycle, several errors can occur. For example, we can have CPU errors, memory bad allocations or accesses, a component failure causing a hardware error, or even an error caused by an I/O device.

First, the error detection service of operating systems must avoid a computer

System completely breaking down when an error happens. So, this service must catch errors and manage them, keeping the entire system as functional as possible.

7] Resource Allocation

Resource allocation means dedicating computing resources to processes and users. We have many resource types, such as CPU operating System controls these resources, it naturally decides which processes use them.

During their lifecycle, a process will typically require multiple computing ~~services~~ resources. There are mainly two resource management challenges: (i) Avoiding that a process demanding a particular resource never gets it
(ii) Avoiding that a process with a resource never releases it.

8] Protection and Security.

Regarding the operating system, we can understand security in different aspects. The first aspect consists of internal security for the processes. The operating system must be able to guarantee the correct execution of processes. Thus, the operating system must provide the required computing resources for the processes and adequately control them. Moreover, it must prevent other processes from inadvertently or maliciously interfering in

given process execution.

9] Accounting

Accounting consists of keeping track of the behaviour of both users and processes in the computing system.

For instance, the operating system analyzes which user require the execution of which processes. So, it can also investigate how many computing resources are requested by the processes executing on the computer.

Finally, the operating system can correlate the obtained information and generate statistics about the user of the computer system and the processes running on it.

ASSIGNMENT 2

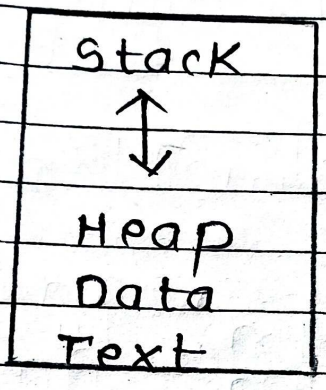
Page No.
Date

PART - A

Q.1. Define Process and its four sections.

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

Sections / Layout of Process:



Stack → Stack contains temporary data such as functions, parameters, addres local variables, etc.

Heap → Heap dynamically allocates mem to a proces during its run time.

Data → This section contains Global variables and static variables.

Text → It contains program counter c contains of processor register.

Teacher's Sign.: _____

Q.2. Draw and explain TCB.

1] Each process in operating system is represented by PCB (Process control block)

2] PCB is also called as Task control block (TCB).

Process state
Process ID
Program counter
Registers
Memory limits
⋮
etc.

Components of PCB:

1] Process ID (PID)
2] Program counter (PC)
3] Process state
4] CPU registers
5] Memory management information
6] Input / output (I/O) status information.
7] Accounting information
8] CPU scheduling information

Teacher's Sign.: _____

Q.3. Explain : i) Independent Processes
ii) Co-operating Processes

i) Independent Processes

1] Its state is not shared with any process.

2] The result of execution depends only on the input state.

3] The result of execution will always b the same for same input.

4] Termination of independent processes w not terminate any other.

ii) Co-operating Processes

1] Its state is shared along other process

2] The result on execution depends on the relative execution sequence and cannot be predicted in advance.

3] The result of execution will not always be the same for same input

4] Termination of cooperating processes may affect every other process.

Q.4. What are the two shared data items for Peterson's solution?

1] A classical software based solution to the critical section problem.

Teacher's Sign.:

2] Petersons solution is restricted to two processes that alternates execution between their critical section and remainder section.
Assume P_1 & P_2

Two shared items for Peterson's solution
1] Int turn - Indicates whose turn is to be entered into the critical section.
2] Boolean flag [2] - It is used to indicate that the process is ready to enter the critical section.

Q.5. Which three semaphores are used to solve producer consumer problem?

1] Full
The full variable is used to drag the space filled in the buffer by the producer process.
Initialize 2 zero.

2] Empty
The empty variable is used to track the empty space in the buffer
Initialized to the buffer's size

3] Mutex
It is used to achieve mutual exclusion

Teacher's Sign.:

Teacher's Sign.:

Page No.
Date

PART - B

Q.1. Explain Principle of Concurrency with advantages and disadvantages.

1] Concurrency is the execution of the multiple instruction, sequences at the same time.

2] It occurs in the operating system when multiple process threads are executing concurrently.

3] The running process threads always communicate with each other through shared memory or message passing.

4] Concurrency results in resource sharing which leads to issues like deadlocks and resource scarcity starvation.

5] The amount of time a process takes to execute cannot be simply estimated and we cannot predict which process will complete first, enabling us build techniques to deal with the problem that concurrency creates.

Teacher's Sign.: _____

Page No.
Date

Problems in concurrency:
1. Locating the programming errors
2. sharing global resources
3. Locking the channel
4. optimum allocation of resources

Advantages of concurrency:
1] Enables running of multiple application

2] Enables better resource utilization that is resources that are unused by one application can be used for another application.

3] Betterment of average response time.

4] Enables better performance by the operating system.

Disadvantages of concurrency:
1] It is necessary to protect multiple application from one another.

2] Since or additional mechanism to coordinat

3] Additional performance overheads & complexit in OS are needed for switching btw applicat

4] Running too many applications concurrently lead to severely degraded performance

Teacher's Sign.:

Page No.
Date

Q.2. Write down the solution for Producer / consumer problem.

Semaphores

1] Semaphores are variables used to indicate the number of resources available in a system at a particular time.

2] Semaphores variables are used to achieve process synchronization.

3] To solve Producer - consumer problems we use 3 semaphores:

1. Full
The Full variable is used to drag the space filled in the buffer by the producer process.
Initialize 2 zero

2. Empty
The empty variable is used to track the empty space in the buffer initialized to the buffer's size.

3. Mutex (Mutual exclusion)
Mutex ensures that any particular time only the Producer or the consumer is accessing the buffer.

Teacher's Sign.:

Date

Functions for solving Producer-consumer problems

1. signal ()
2. wait ()

Pseudo codes:

Producer processes

```
void Producer () {
    while (true) {
    // producer produces an item
    wait (empty);
    wait (mutex);
    add ();
    signal (mutex);
    signal (full);
    }
}
```

consumer processes

```
void consumer () {
    while (true) {
    // consumer consumes an item
    wait (full);
    wait (mutex);
    consume ();
    signal (mutex);
    signal (empty);
    }
}
```

Teacher's Sign.: _____

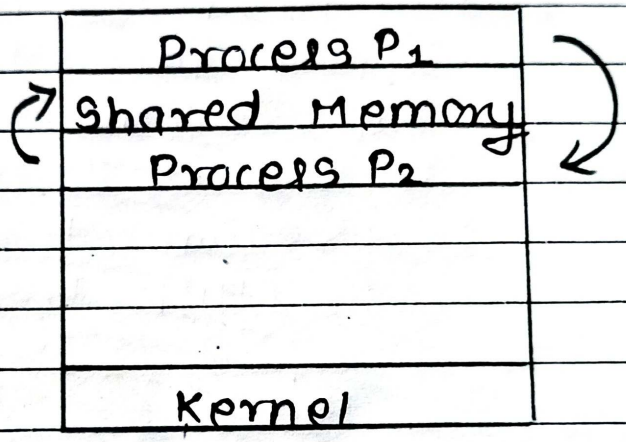
Q.3. Explain models of IPC

Interprocess communication model

→ Shared Memory
→ Message Passing

① Shared Memory

Structure of shared memory model :



1] In shared memory model a region of memory that is shared by cooperating processes is established.

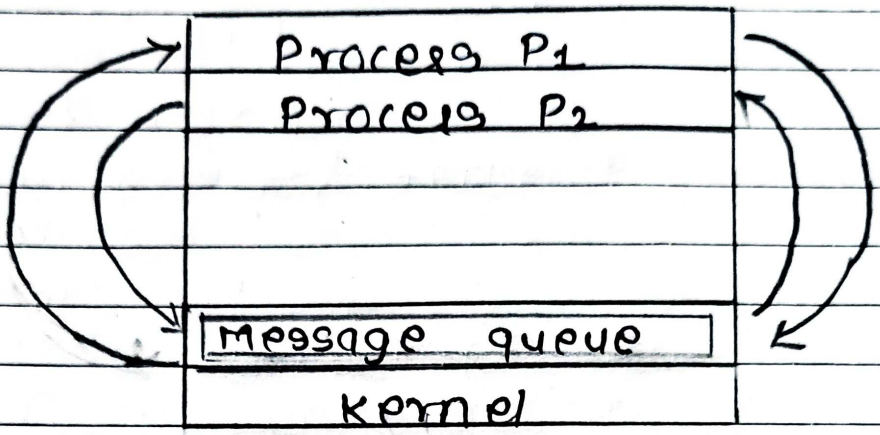
2] Process can then exchange information by reading and writing the data to the shared region.

Teacher's Sign.: _____

Page No.
Date

(2) Message Passing

Structure of Message Passing model.



1] In the Message Passing model communication takes place by means of messages exchanged between the cooperating processes.

2] Multiple processes can read and write data to the message queue and this messages are then stored until they are retrieved by the recepients.

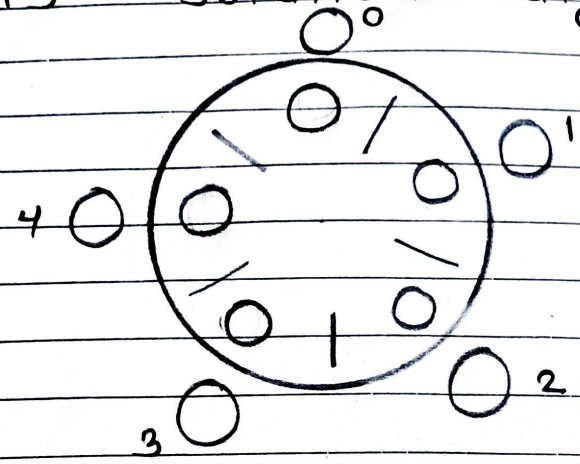
Teacher's Sign.:

Teacher's Sign.:

Page No.
Date

PART - C

Q.1. Explain Dining Philosophers problem with its solution algorithm.



1] The Dining Philosophers problem states that 'K' philosophers are seated around a circular table with one chopstick between each pair of philosophers.

2] The philosophers can either eat or think.

3] When a philosophers thinks; the philosopher interact with any other philosopher.

4] When a philosophers wants to eat; the philosopher have to pickup the two chopsticks present adjacent to him.

Teacher's Sign.:

Page No.
Date

Rules to Bound Philosophers:

1. The philosopher can pickup one chopstick at a time.

2. One cannot take a chopstick that is already in the hand of other philosopher

3. When a Philosopher have two chopsticks at that time he eats without releasing his chopsticks.

4. once he finishes eating he puts down both chopsticks and starts thinking again.

* Philosophers are Processes

* chopsticks are Resources

PROBLEM:
Since the chopsticks are limited no two philosophers who are adjacent to each other can eat.

SOLUTION:

Represent each chopstick with a semaferre phores

Teacher's Sign.:

A philosopher tries to grab a chopstick by executing a wait operation on that semaphores wait() operation implies lock and no interruption.

After eating the philosopher releases the chopsticks by executing the signal operation.

structure of philosopher [i];

```
do {
    wait (chopstick [i]);
    wait (chopstick [(i+1) % 5]);
    // eat
    signal (chopstick [i]);
    signal (chopstick [(i+1) % 5]);
    // think
}
while (TRUE);
```

Page No.
Date

Q.2. Explain critical section problem with any two of the following solutions:

i) Peterson's solution

Peterson's solution requires two shared data items:
int turn - Indicates whose turn it is to enter into the critical section
boolean flag [2] - indicates when a process wants to enter into critical section.

Structure of process Pi in Peterson's solution:

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

Teacher's Sign.:

ii) Dekker's solution

P1 :

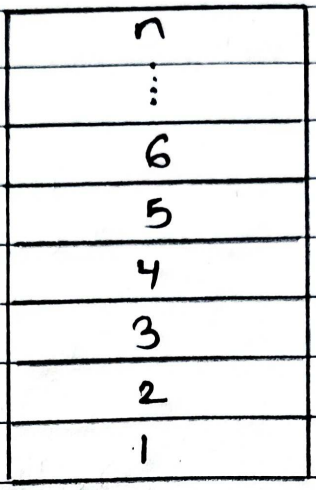
```
do {
    Flag [1] = TRUE ;
    while (Flag [2])
    {
      if (turn == 2)
      {
       Flag [1] = False;
       while (turn == 2);
       Flag [1] = True;
      }
    }
     critical section
    turn = 2;
    Flag [1] = False;
        Remainder Section
    }
    while (True);
```

P2 :

```
do {
    Flag [2] = True;
    while (Flag [1])
    {
      if (turn == 1)
      {
       Flag [2] = False;
       while (turn == 1) ;
       Flag [2] = true;
      }
    }
     critical section
    turn = 1 ;
    Flag [2] = False;
        Remainder Section
    }
    while (True);
```

Teacher's Sign.: _____

Q.3. Explain Bounded Buffer Problem in detail with its solution.



1] Bounded Buffer Problem is also known as Producer - consumer problem.

2] It is known to be a classical synchronization problem.

3] In this problem we use a buffe n slots and each slot is capab of storing 1 unit data.

4]

Producers :

1] The Producer tries to insert data into an empty slot of buffer.

Teacher's Sign.:

2] Problem → Producer must not insert data when the buffer is full.

Consumer:

1] The consumer tries to remove data from a filled slot in the buffer.

2] Problem → The consumer must not remove data when the buffer is empty.

Solution for P-C problem:

★ Semaphores

1) Full
Full variable is used to drag the space filled in the buffer by the producer process.
Initialize 2 zeroes.

2) Empty
used to track the empty space in the buffer.
Initialized to the buffer's size.

3) Mutex (mutual exclusion)
only one Producer or consumer can access the buffer

Teacher's Sign.:

Producer process

```
void Producer () {
    while (true) {
    wait (empty);
    wait (mutex);
    add ();
    signal (mutex);
    signal (full);
    }
}
```

Consumer Process

```
void consumer () {
    while (true) {
    wait (full);
    wait (mutex);
    consume ();
    signal (mutex);
    signal (empty);
    }
}
```

Teacher's Sign.:

ASSIGNMENT NO. 03

Subject: **OPERATING SYSTEMS**
Course: **BCA (A/B/TCS), BSC (CS/IT/Bioinformatics)**

Issue Date: **11-12-2023**
Submission Date: ----------

PART-A [Short Answer Type- 2M (Write any 3)]

Q.1.1	Why CPU scheduling is required?
Q.1.2	Define: 1) Waiting Time 2) Turnaround Time
Q.1.3	Define Throughput.
Q.1.4	Draw the system model of deadlock.
Q.1.5	What are the four necessary conditions for deadlock?

PART-B [Long Answer Type- 5M (Write any 2)]

Q.2.1	Draw and Explain Process Transition Diagram.
Q.2.2	Explain performance criteria of CPU scheduling.
Q.2.3	Define and explain Schedulers and its three types.

PART-C [Very Long Answer Type- 10M (Write any 2)]

Q.3.1 Given:

Process	**P1**	**P2**	**P3**	**P4**	**P5**
Arrival Time	4	0	6	3	1
Burst Time	1	5	3	2	1

Calculate following terminologies with the help of Shortest Remaining Time First Algorithm (Pre-emptive): 1) Completion Time 2) Turnaround time 3) Waiting Time 4) Response Time 6) Avg. turnaround time 7) Avg. Waiting time.

<table>
<tr><td>Q.3.2</td><td>Given:
<table><tr><td>Process</td><td>P1</td><td>P2</td><td>P3</td><td>P4</td><td>P5</td></tr><tr><td>Arrival Time</td><td>4</td><td>0</td><td>6</td><td>3</td><td>1</td></tr><tr><td>Burst Time</td><td>1</td><td>5</td><td>3</td><td>2</td><td>1</td></tr></table>Calculate following terminologies with the help of Shortest Job First Algorithm (Non Pre-emptive) : 1) Completion Time 2) Turnaround time 3) Waiting Time 4) Response Time 6) Avg. turnaround time 7) Avg. Waiting time</td></tr>
<tr><td>Q.3.3 e.</td><td>Define & explain FCFS algorithm with the help of an exampl</td></tr>
</table>

Q.1.1 Why CPU scheduling is required?

In Multiprogramming, if the long term scheduler picks more I/O bound processes then most of the time, the CPU remains idol. The task of Operating system is to optimize the utilization of resources.

If most of the running processes change their state from running to waiting then there may always be a possibility of deadlock in the system. Hence to reduce this overhead, the OS needs to schedule the jobs to get the optimal utilization of CPU and to avoid the possibility to deadlock.

Q.1.2 Define: 1) Waiting Time
2) Turnaround Time

Waiting Time:

Waiting time is the amount of time spent by a process waiting in the ready queue for getting the CPU.

Waiting time = Turn Around time – Burst time

Turnaround Time:

Turn Around time is the total amount of time spent by a process in the system.

When present in the system, a process is either waiting in the ready queue for getting the CPU or it is executing on the CPU.

Turn Around time = Burst time + Waiting time

OR

Turn Around time = Completion time – Arrival time

Q.1.3 Define Throughput.

Throughput is a measure of how many units of information a system can process in a given amount of time. It is applied broadly to systems ranging from various aspects of computer and network systems to organizations.

Related measures of system productivity include the speed with which a specific workload can be completed and response time, which is the amount of time between a single interactive user request and receipt of the response.

Q.1.4 Draw the system model of Deadlock.

A deadlock occurs when a set of processes is stalled because each process is holding a resource and waiting for another process to acquire another resource. In the diagram below, for example, Process 1 is holding Resource 1 while Process 2 acquires Resource 2, and Process 2 is waiting for Resource 1.

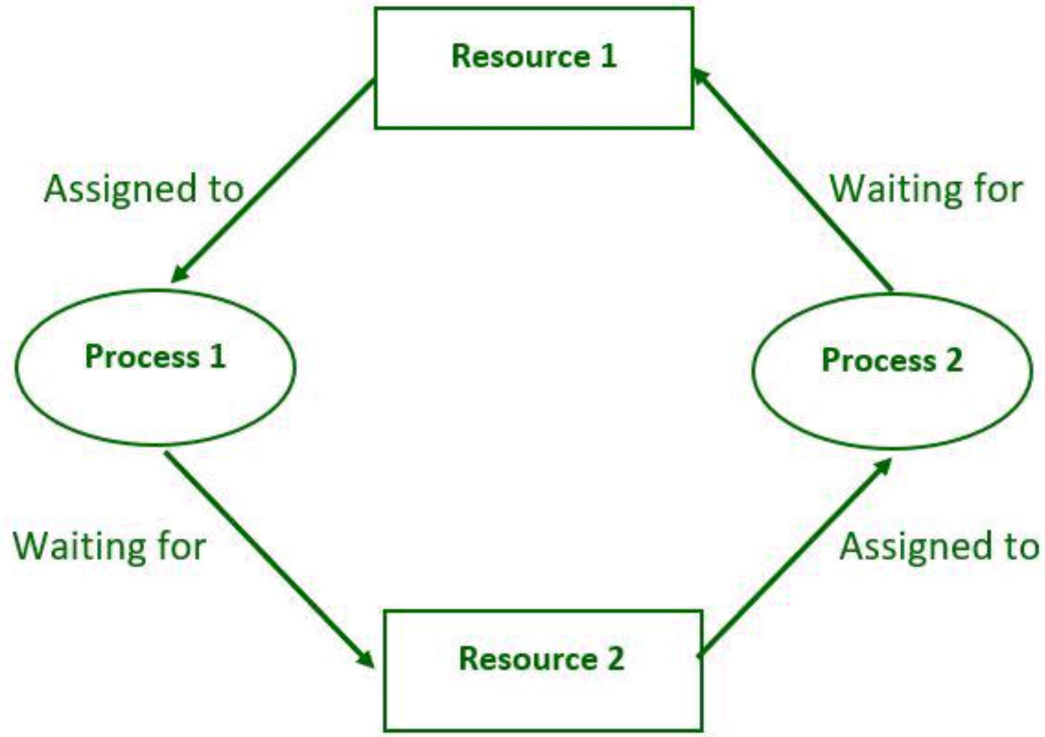


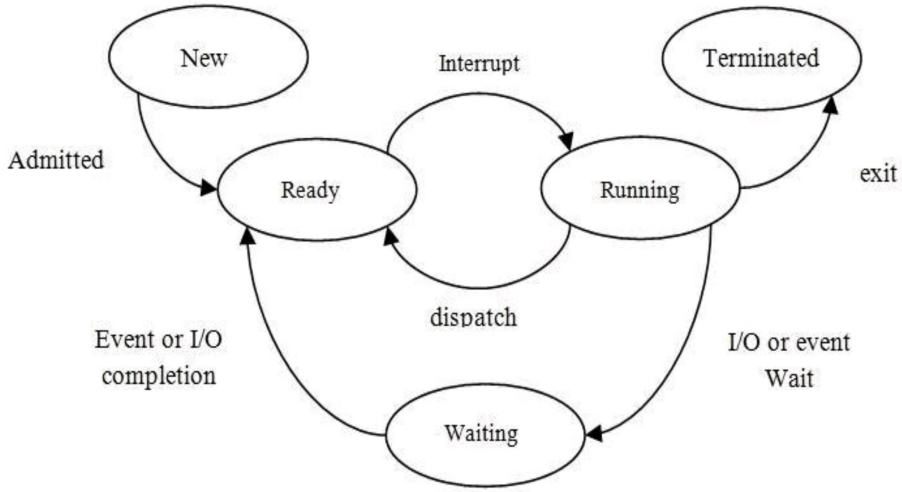
Figure: Deadlock in Operating system

Q.1.5 What are the four necessary conditions for Deadlock?

There are four different conditions that result in Deadlock. These four conditions are also known as Coffman conditions and these conditions are;

1.Mutual Exclusion
2.Hold and Wait
3.No pre-emption
4.Circular Wait

Q.2.1 Draw and Explain Process Transition Diagram



(I) Running to ready state:

- A process in the running state has all of the resources that it needs for further execution, including a processor.
- The long term scheduler picks up a new process from second memory and loads it into the main memory when there are sufficient resources available.
- The process is now in ready state, waiting for its execution.

(II) waiting to ready:

- Process waiting for some event such as completion of I/O operation, synchronization signal, etc.
- A process moves from waiting state to ready state if the event the process has been waiting for, occurs.
- The process is now ready for execution.

(III) Running to waiting:

- The process in the main memory that is waiting for some event.
- A process is put in the waiting state if it must wait for some event. For example, the process may request some resources or memory which might not be available.
- The process may be waiting for an I/O operation or it may be waiting for some other process to finish before it can continue execution.

(IV) blocked to ready:

- The process is in secondary memory but not yet ready for execution.

- The process moves from Blocked to Ready state if the event, the process has been waiting for occurs.

(v) Running to terminated:

- The process has finished execution.
- The OS moves a process from running state to terminated state if the process finishes execution or if it aborts.
- Whenever the execution of a process is completed in running state, it will exit to terminate state, which is the completion of process.

Q.2.2 Explain performance criteria of CPU scheduling.

There are some CPU scheduling criteria given below –

1. CPU Utilization

CPU utilization is a criterion used in CPU scheduling that measures the percentage of time the CPU is busy processing a task. It is important to maximize CPU utilization because when the CPU is idle, it is not performing any useful work, and this can lead to wasted system resources and reduced productivity.

A high CPU utilization indicates that the CPU is busy and working efficiently, processing as many tasks as possible. However, a too high CPU utilization can also result in a system slowdown due to excessive competition for resources.

2. Throughput

Throughput is a criterion used in CPU scheduling that measures the number of tasks or processes completed within a specific period. It is important to maximize throughput because it reflects the efficiency and productivity of the system. A high throughput indicates that the system is processing tasks efficiently, which can lead to increased productivity and faster completion of tasks.

3. Turnaround Time

Turnaround time is a criterion used in CPU scheduling that measures the time it takes for a task or process to complete from the moment it is submitted to the system until it is fully processed and ready for output. It is important to minimize turnaround time because it

reflects the overall efficiency of the system and can affect user satisfaction and productivity.

4. Waiting Time

Waiting time is a criterion used in CPU scheduling that measures the amount of time a task or process waits in the ready queue before it is processed by the CPU. It is important to minimize waiting time because it reflects the efficiency of the scheduling algorithm and affects user satisfaction.

5. Response Time

Response time is a criterion used in CPU scheduling that measures the time it takes for the system to respond to a user's request or input. It is important to minimize response time because it affects user satisfaction and the overall efficiency of the system. A short response time indicates that the system is processing tasks quickly and efficiently, leading to improved user satisfaction and productivity. On the other hand, a long response time can result in user frustration and decreased productivity. Some CPU scheduling algorithms that prioritize response time include Round Robin, Priority scheduling, and Multilevel Feedback Queue (MLFQ). These algorithms aim to prioritize tasks that require immediate attention, such as user input, to reduce response time and improve system efficiency.

Q.2.3 Define and explain Schedulers and its three types.

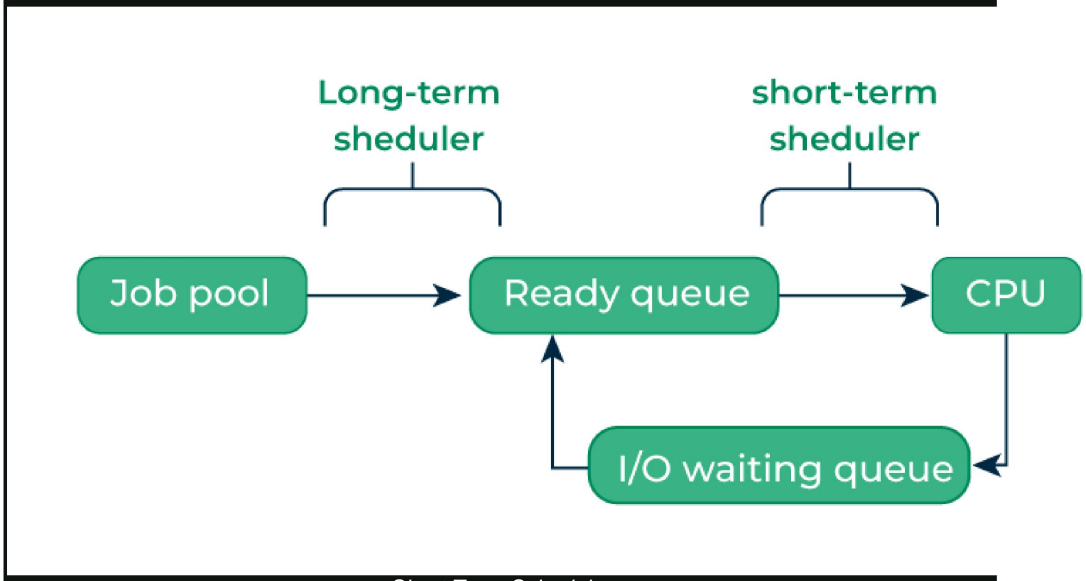
There are three types of process schedulers:

1. Long Term or Job Scheduler
It brings the new process to the 'Ready State'. It controls the Degree of Multi-programming, i.e., the number of processes present in a ready state at any point in time. It is important that the long-term scheduler make a careful selection of both I/O and CPU-bound processes. I/O-bound tasks are which use much of their time in input and output operations while CPU-bound processes are which spend their time on the CPU. The job scheduler increases efficiency by maintaining a balance

between the two. They operate at a high level and are typically used in batch-processing systems.

2. Short-Term or CPU Scheduler

It is responsible for selecting one process from the ready state for scheduling it on the running state. Note: Short-term scheduler only selects the process to schedule it doesn't load the process on running. Here is when all the scheduling algorithms are used. The CPU scheduler is responsible for ensuring no starvation due to high burst time processes.



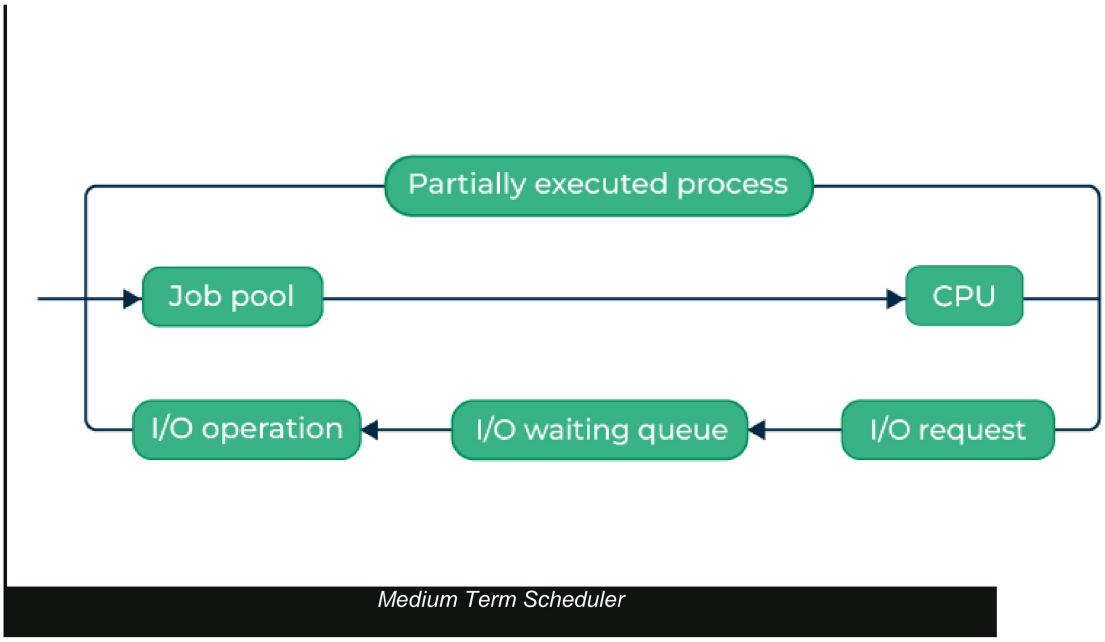
Short Term Scheduler

The dispatcher is responsible for loading the process selected by the Short-term scheduler on the CPU (Ready to Running State) Context switching is done by the dispatcher only. A dispatcher does the following:

- Switching context.
- Switching to user mode.
- Jumping to the proper location in the newly loaded program.

3. Medium-Term Scheduler

It is responsible for suspending and resuming the process. It mainly does swapping (moving processes from main memory to disk and vice versa). Swapping may be necessary to improve the process mix or because a change in memory requirements has overcommitted available memory, requiring memory to be freed up. It is helpful in maintaining a perfect balance between the I/O bound and the CPU bound. It reduces the degree of multiprogramming.


Partially executed process
Job pool
CPU
I/O operation
I/O waiting queue
I/O request
Medium Term Scheduler

Q.3.1

1. Scheduling the processes based on Burst Time and Arrival Time:

Process	Arrival Time	Burst Time
P1	4	1
P2	0	5
P3	6	3
P4	3	2
P5	1	1

2. Calculate the Completion Time:

To do this, we will start by looking at the process with the shortest remaining burst time. If two processes have the same remaining burst time, we will prioritize the one that arrived earlier.

Time 0: P2 arrives, starts execution (Burst Time: 5)

Time 1: P5 arrives, preempts P2 as its remaining burst time is 4 (Burst Time: 1)

Time 2: P4 arrives, preempts P5 as its remaining burst time is 1 (Burst Time: 2)

Time 3: P4 completes its execution.

Time 4: P1 arrives, preempts P4 as its remaining burst time is 1 (Burst Time: 1)

Time 5: P1 completes its execution.

Time 6: P3 arrives, preempts P2 as its remaining burst time is 4 (Burst Time: 3)

Time 7: P3 completes its execution.

Time 8: P2 completes its execution.

Thus, the Completion Times are: P1: 5, P2: 8, P3: 7, P4: 3, P5: 2

3. Calculate the Turnaround Time:

Turnaround Time = Completion Time - Arrival Time

P1: 5 - 4 = 1 P2: 8 - 0 = 8 P3: 7 - 6 = 1 P4: 3 - 3 = 0 P5: 2 - 1 = 1

4. Calculate the Waiting Time:

Waiting Time = Turnaround Time - Burst Time

P1: 1 - 1 = 0 P2: 8 - 5 = 3 P3: 1 - 3 = -2 P4: 0 - 2 = -2 P5: 1 - 1 = 0

(Note: Negative waiting times indicate that the process was not kept waiting at all.)

5. Calculate the Response Time:

Response Time is the time from the arrival of the process until the first time it gets the CPU.

P1: 4 - 4 = 0 P2: 0 - 0 = 0 P3: 6 - 6 = 0 P4: 3 - 3 = 0 P5: 1 - 1 = 0

6. Calculate Average Turnaround Time:

Average Turnaround Time = (Sum of Turnaround Times) / Number of Processes

Average Turnaround Time = (1 + 8 + 1 + 0 + 1) / 5 = 11 / 5 = 2.2

7. Calculate Average Waiting Time:

Average Waiting Time = (Sum of Waiting Times) / Number of Processes

Average Waiting Time = (0 + 3 + (-2) + (-2) + 0) / 5 = -1 / 5 = -0.2

(Note: Negative average waiting time is not possible, so there might be a mistake in the calculations.)

To summarize:

1. Completion Times: P1: 5, P2: 8, P3: 7, P4: 3, P5: 2
2. Turnaround Times: P1: 1, P2: 8, P3: 1, P4: 0, P5: 1
3. Waiting Times: P1: 0, P2: 3, P3: -2, P4: -2, P5: 0
4. Response Times: P1: 0, P2: 0, P3: 0, P4: 0, P5: 0
5. Avg. Turnaround Time: 2.2
6. Avg. Waiting Time: -0.2 (Note: This seems incorrect. The calculation should be reviewed.)

Q.3.2

1. Scheduling the processes based on Burst Time and Arrival Time:

Process	Arrival Time	Burst Time
P1	4	1
P2	0	5
P3	6	3
P4	3	2
P5	1	1

2. Sort the processes based on Burst Time (and Arrival Time if Burst Time is the same):

Process Order: P5, P1, P4, P3, P2

3. Calculate the Completion Time:

The processes will be executed in the order: P5, P1, P4, P3, P2.

Completion Times: P5: 1 P1: 2 P4: 4 P3: 7 P2: 12

4. Calculate the Turnaround Time:

Turnaround Time = Completion Time - Arrival Time

P5: 1 - 1 = 0 P1: 2 - 4 = -2 P4: 4 - 3 = 1 P3: 7 - 6 = 1 P2: 12 - 0 = 12

(Note: Negative turnaround times indicate that the process finished before it arrived.)

5. Calculate the Waiting Time:

Waiting Time = Turnaround Time - Burst Time

P5: 0 - 1 = -1 P1: -2 - 1 = -3 P4: 1 - 2 = -1 P3: 1 - 3 = -2 P2: 12 - 5 = 7

(Note: Negative waiting times are due to the non-preemptive nature of the algorithm.)

6. Calculate the Response Time:

For non-preemptive SJF, the response time is the same as the waiting time since a process begins execution only after all the processes with shorter burst times have completed.

7. Calculate Average Turnaround Time:

Average Turnaround Time = (Sum of Turnaround Times) / Number of Processes

Average Turnaround Time = (0 - 2 + 1 + 1 + 12) / 5 = 12 / 5 = 2.4

8. Calculate Average Waiting Time:

Average Waiting Time = (Sum of Waiting Times) / Number of Processes

Average Waiting Time = (-1 - 3 - 1 - 2 + 7) / 5 = 0

To summarize:

1. Completion Times: P5: 1, P1: 2, P4: 4, P3: 7, P2: 12
2. Turnaround Times: P5: 0, P1: -2, P4: 1, P3: 1, P2: 12
3. Waiting Times: P5: -1, P1: -3, P4: -1, P3: -2, P2: 7
4. Response Times: P5: -1, P1: -3, P4: -1, P3: -2, P2: 7
5. Avg. Turnaround Time: 2.4
6. Avg. Waiting Time: 0

Q.3.3

First-Come, First-Served (FCFS) Algorithm

Definition:
The First-Come, First-Served (FCFS) scheduling algorithm is the simplest CPU scheduling algorithm. In this algorithm, the process that arrives first gets executed first. The processes are executed in the order they arrive in the ready queue, without any priority.

Explanation with an Example:

Consider a set of processes arriving at a CPU for execution:

Process	Arrival Time	Burst Time
P1	0	5
P2	1	3
P3	2	2
P4	3	4

1. **Arrival Time**: Indicates the time at which a process arrives at the CPU.
2. **Burst Time**: Represents the amount of time a process requires to complete its execution.

Step-by-Step Execution:

1. **Process P1 (Arrival Time: 0, Burst Time: 5)**
   - P1 arrives at time 0 and starts execution immediately because no other process has arrived.
   - P1 completes its execution at time 5.
2. **Process P2 (Arrival Time: 1, Burst Time: 3)**
   - P2 arrives at time 1 but has to wait because P1 is currently executing (since P1 arrived before P2).
   - P2 starts its execution at time 5 (when P1 completes) and finishes at time 8.
3. **Process P3 (Arrival Time: 2, Burst Time: 2)**
   - P3 arrives at time 2 and also waits in the ready queue.
   - After P2 finishes, P3 starts its execution at time 8 and finishes at time 10.
4. **Process P4 (Arrival Time: 3, Burst Time: 4)**
   - P4 arrives at time 3 and waits because P2 and P3 are still executing.
   - P4 starts its execution at time 10 (after P3 completes) and finishes at time 14.

Gantt Chart Representation:
A Gantt chart can visually represent the execution of processes over time.

Time	P1	P2	P3	P4
0	P1			
1	P1			
2	P1			
3	P1			
4	P1			
5		P2		

Time	P1	P2	P3	P4
6		P2		
7		P2		
8		P2	P3	
9			P3	
10			P3	P4
11				P4
12				P4
13				P4
14				

In this example, the FCFS algorithm executes processes based solely on their arrival order, without considering the length of the processes or any other criteria.

ASSIGNMENT NO. 04

Subject: **OPERATING SYSTEMS**
Course: **BCA (A/B/TCS), BSC (CS/IT/Bioinformatics)**

Issue Date: **11-12-2023**
Submission Date: ----------

PART-A [Short Answer Type- 2M (Write any 3)]

Q.1.1	Define multiprogramming. Mention the formula of 'Degree of multiprogramming'
Q.1.2	State different types of RAM and ROM
Q.1.3	Draw the Memory layout of Resident monitor.
Q.1.4	In a designed OS there is 8MB of RAM assumed, processes incoming are of 2 MB, and each process requires 80% of time for I/O operation, Then Calculate the CPU utilization of the system.
Q.1.5	Define Contiguous and Non-contiguous memory allocation.

PART-B [Long Answer Type- 10M (Write any 2)]

Q.2.1	Write a short note on: 1) Basic Bare machine. 2) Resident Monitor
Q.2.2	Explain in detail with drawbacks and benefits : Static Partitioning memory allocation technique & Dynamic Partitioning memory allocation technique.
Q.2.3	Explain any two types Non- Contiguous memory allocation techniques with drawbacks and benefits.

Q.1.1 Define multiprogramming. Mention the formula of 'Degree of multiprogramming'

Multiprogramming is a computer system design technique where multiple programs are in memory at the same time, each taking turns using the CPU.
The main objective of multiprogramming is to maximize CPU utilization and increase the throughput of the system by allowing multiple jobs to share the processor efficiently.
The formula to calculate the degree of multiprogramming is:

$$\text{Degree of Multiprogramming} = \frac{\text{No of Processes in Memory}}{\text{Total No of Processes}}$$

Q.1.2 State different types of RAM and ROM

RAM (Random Access Memory):

1. SRAM (Static RAM):
2. DRAM (Dynamic RAM):
3. SDRAM (Synchronous DRAM):
4. DDR (Double Data Rate) SDRAM:
5. VRAM (Video RAM):

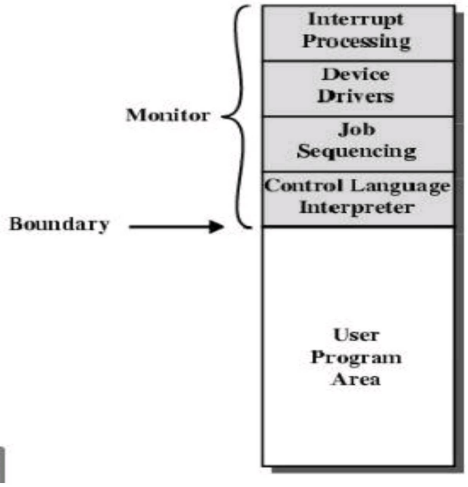
ROM (Read-Only Memory):

1. Mask ROM (MROM):
2. PROM (Programmable ROM):
3. EPROM (Erasable Programmable ROM):
4. EEPROM (Electrically Erasable Programmable ROM)

Q.1.3 Draw the Memory layout of Resident monitor.

The Resident monitors are divided into 4 parts as:

1. Control Language Interpreter
2. Loader
3. Device Driver
4. Interrupt Processing



Q.1.4 In a designed OS there is 8MB of RAM assumed, processes incoming are of 2 MB, and each process requires 80% of time for I/O operation, Then Calculate the CPU utilization of the system.

(Solve by yourself)

Q.1.5 Define Contiguous and Non-contiguous memory allocation.

Contiguous Memory Allocation:

In contiguous memory allocation, each process occupies a single contiguous block of memory. The memory space for a process is allocated as one continuous segment without any fragmentation between different parts of the process's memory.

Non-contiguous Memory Allocation:

In non-contiguous memory allocation, a process's memory space is divided into multiple non-contiguous segments or pages scattered throughout the physical memory. This approach allows for more flexible and efficient use of memory by accommodating processes in available fragmented memory spaces.

Q.2.1 Write a short note on: 1) Basic Bare machine.
2) Resident Monitor

1) Basic Bare Machine:

A Basic Bare Machine refers to the simplest form of a computer system that can execute a set of basic instructions without any operating system or software support. It represents the fundamental hardware architecture and minimal software required to perform basic computing tasks.

Key Characteristics:

Hardware Components: Consists of essential hardware components, including the CPU, memory, I/O devices (e.g., keyboard, monitor), and storage devices (e.g., disk drive).

Minimal Software Support: Operates with minimal software, typically a bootstrap loader or firmware, to initialize the system and execute basic instructions.

Limited Functionality: Provides basic computing capabilities and lacks advanced features, multitasking support, or user-friendly interfaces commonly found in modern operating systems.

Direct Hardware Interaction: Users interact directly with the hardware through low-level instructions, requiring a deep understanding of the underlying system architecture.

2) Resident Monitor:

A Resident Monitor, also known as a Resident Operating System or Monitor System, is a type of operating system that remains permanently in memory while managing the execution of user programs and providing system services.

Key Characteristics:

Memory Resident: Resides in memory continuously, enabling quick access to system services and resources without frequent reloading or swapping.

Process Management: Manages the execution of multiple user processes, including process scheduling, memory allocation, and resource allocation.

System Services: Provides essential system services, such as file management, I/O operations, device drivers, and inter-process communication.

User Interaction: Facilitates user interaction through command-line interfaces, system calls, or application programming interfaces (APIs) for executing programs and accessing system resources.

Resource Management: Ensures efficient utilization of system resources, including CPU, memory, I/O devices, and storage, to optimize system performance and responsiveness.

Basic Functionality: Offers a basic set of functionalities to support user programs and ensure the proper operation of the computer system without the need for a full-fledged operating system.

Q.2.2 Explain in detail with drawbacks and benefits :
Static Partitioning memory allocation technique & Dynamic Partitioning memory allocation technique

Static Partitioning Memory Allocation Technique:

In Static Partitioning, the system's memory is divided into fixed-size partitions or regions, and each partition is assigned to a specific process. The size of each partition remains constant, and the partitions are created during system initialization or configuration.

Benefits:

Simplicity: Static Partitioning is straightforward to implement and manage, as the memory is divided into fixed-size partitions without the need for dynamic adjustments.

Predictability: Provides predictable memory allocation and ensures that each process has a dedicated memory space, avoiding conflicts or overlaps between processes.

Efficiency: Reduces fragmentation by allocating memory in fixed-size partitions, optimizing memory utilization and minimizing wasted space.

Drawbacks:

Wastage of Memory: May lead to internal fragmentation, where the allocated memory partition is larger than the actual memory requirements of the process, resulting in wasted memory space.

Limited Flexibility: Lacks flexibility in adapting to varying memory demands and dynamic changes in the system, as the partition sizes are fixed and cannot be adjusted dynamically.

Inefficient Resource Utilization: May result in inefficient use of memory resources, especially when processes have different memory requirements or when the total available memory is not fully utilized.

Dynamic Partitioning Memory Allocation Technique:

In Dynamic Partitioning, the system's memory is divided into variable-sized partitions or blocks, and each partition is allocated to a process based on its memory requirements. The size and number of partitions can change dynamically as processes are loaded, executed, and terminated.

Benefits:

Flexibility: Offers greater flexibility in adapting to varying memory demands and accommodating processes with different memory requirements dynamically.

Optimized Memory Utilization: Reduces internal fragmentation by allocating memory in variable-sized partitions that match the actual memory needs of processes, minimizing wasted memory space.

Improved System Responsiveness: Enables efficient utilization of memory resources and supports the concurrent execution of multiple processes, enhancing system responsiveness and performance.

Dynamic Resource Management: Facilitates dynamic resource management and optimization by adjusting partition sizes and reallocating memory based on the current system state and workload.

Drawbacks:

Complexity: Dynamic Partitioning is more complex to implement and manage compared to Static Partitioning, as it involves dynamically creating, resizing, and deallocating memory partitions during system operation.

Fragmentation Issues: May still experience fragmentation, such as external fragmentation, where free memory blocks are scattered throughout the memory space and cannot be effectively utilized for new allocations.

Overhead and Latency: Introduces additional overhead and latency in memory management operations, such as partition creation, resizing, and memory mapping, which may impact system performance and responsiveness.

Q.2.3 Explain any two types Non- Contiguous memory allocation techniques with drawbacks and benefits.

1) Paging:

Paging divides the logical address space of a process into fixed-size blocks called pages. The physical memory is divided into fixed-size frames that correspond to the size of the pages. During execution, pages of a process are loaded into available frames in physical memory, and the mapping between logical pages and physical frames is maintained using a page table.

Benefits:

Flexible Memory Allocation: Enables flexible memory allocation and supports the dynamic loading and execution of processes, allowing efficient use of physical memory resources.

Memory Protection: Provides memory protection and isolation between processes by maintaining separate page tables and ensuring that processes cannot access memory regions allocated to other processes.

Supports Virtual Memory: Facilitates the implementation of virtual memory systems, allowing processes to execute using a larger logical address space than the available physical memory.

Drawbacks:

Fragmentation: May lead to fragmentation, such as internal fragmentation within pages, where the allocated memory within a page is not fully utilized, resulting in wasted space.

Overhead: Introduces additional overhead in managing page tables, address translation, and memory mapping operations, which may impact system performance and responsiveness.

Complexity: Requires complex memory management algorithms and policies, such as page replacement algorithms (e.g., LRU, FIFO) and memory allocation strategies, to optimize memory utilization and maintain system performance.

2) Segmentation:

Segmentation divides the logical address space of a process into variable-sized segments that correspond to different sections or regions of the program, such as code, data, stack, and heap. Each segment is assigned a base address and a size, and segments are loaded into non-contiguous memory regions in physical memory.

Benefits:

Supports Modular Programming: Facilitates modular programming and supports the organization of programs into distinct segments, such as code, data, and stack, to improve code readability, maintainability, and reusability.

Flexibility in Memory Management: Provides flexibility in memory management and supports dynamic allocation, relocation, and sharing of segments between processes, allowing efficient use of memory resources.

Enhanced Memory Protection: Enables enhanced memory protection and access control by maintaining separate segment descriptors and access rights for each segment, ensuring secure and isolated execution of processes.

Drawbacks:

Fragmentation: May lead to fragmentation, such as external fragmentation between segments, where the available memory space is fragmented into non-contiguous regions that cannot be effectively utilized for new allocations.

Complex Address Translation: Requires complex address translation mechanisms, such as segment tables or segment descriptors, to map logical addresses to physical addresses and manage the mapping between segments and memory regions.

Segmentation Overhead: Introduces additional overhead in managing segment descriptors, address translation, and memory protection mechanisms, which may impact system performance and increase memory access latency.

I/O buffering and its Various Techniques

Read Courses Jobs

A buffer is a memory area that stores data being transferred between two devices or between a device and an application.

Uses of I/O Buffering :

- Buffering is done to deal effectively with a speed mismatch between the producer and consumer of the data stream.
- A buffer is produced in main memory to heap up the bytes received from modem.
- After receiving the data in the buffer, the data get transferred to disk from buffer in a single operation.
- This process of data transfer is not instantaneous, therefore the modem needs another buffer in order to store additional incoming data.
- When the first buffer got filled, then it is requested to transfer the data to disk.
- The modem then starts filling the additional incoming data in the second buffer while the data in the first buffer getting transferred to disk.
- When both the buffers completed their tasks, then the modem switches back to the first buffer while the data from the second buffer get transferred to the disk.
- The use of two buffers disintegrates the producer and the consumer of the data, thus minimizes the time requirements between them.
- Buffering also provides variations for devices that have different data transfer sizes.

DEFINITION

RAID (redundant array of independent disks)

The term was coined by David Patterson, Garth A. Gibson, and Randy Katz at the University of California,Berkeley in 1987.

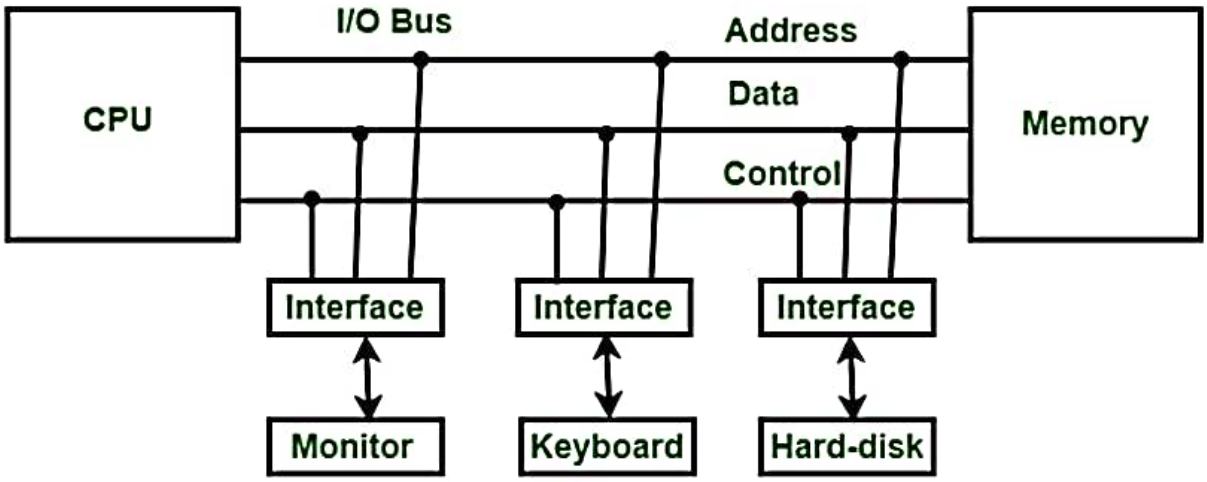
What is RAID?

RAID (redundant array of independent disks) is a way of storing the same data in different places on multiple hard disks or solid-state drives (SSDs) to protect data in the case of a drive failure. There are different RAID levels, however, and not all have the goal of providing redundancy.

Introduction to Input-Output Interface

Read Courses Jobs

Input-Output Interface is used as an method which helps in transferring of information between the internal storage devices i.e. memory and the external peripheral device . A peripheral device is that which provide input and output for the computer, it is also called Input-Output devices. For Example: A keyboard and mouse provide Input to the computer are called input devices while a monitor and printer that provide output to the computer are called output devices. Just like the external hard-drives, there is also availability of some peripheral devices which are able to provide both input and output.



Input-Output Interface

In micro-computer base system, the only purpose of peripheral devices is just to provide **special communication links** for the interfacing them with the CPU. To resolve the differences between peripheral devices and CPU, there is a special need for communication links.

What is the file?

The file can be explained as the smallest unit of storage on a computer system. The user can perform file operations like open, close, read, write, and modify.

Different types of files are:

- Executable file

In an executable file, the binary code that is loaded in the memory for execution is stored. It is stored in an exe type file.

- Source file

The source file has subroutines and functions that are compiled later.

- Object file

An object file is a sequence of bytes used by the linker.

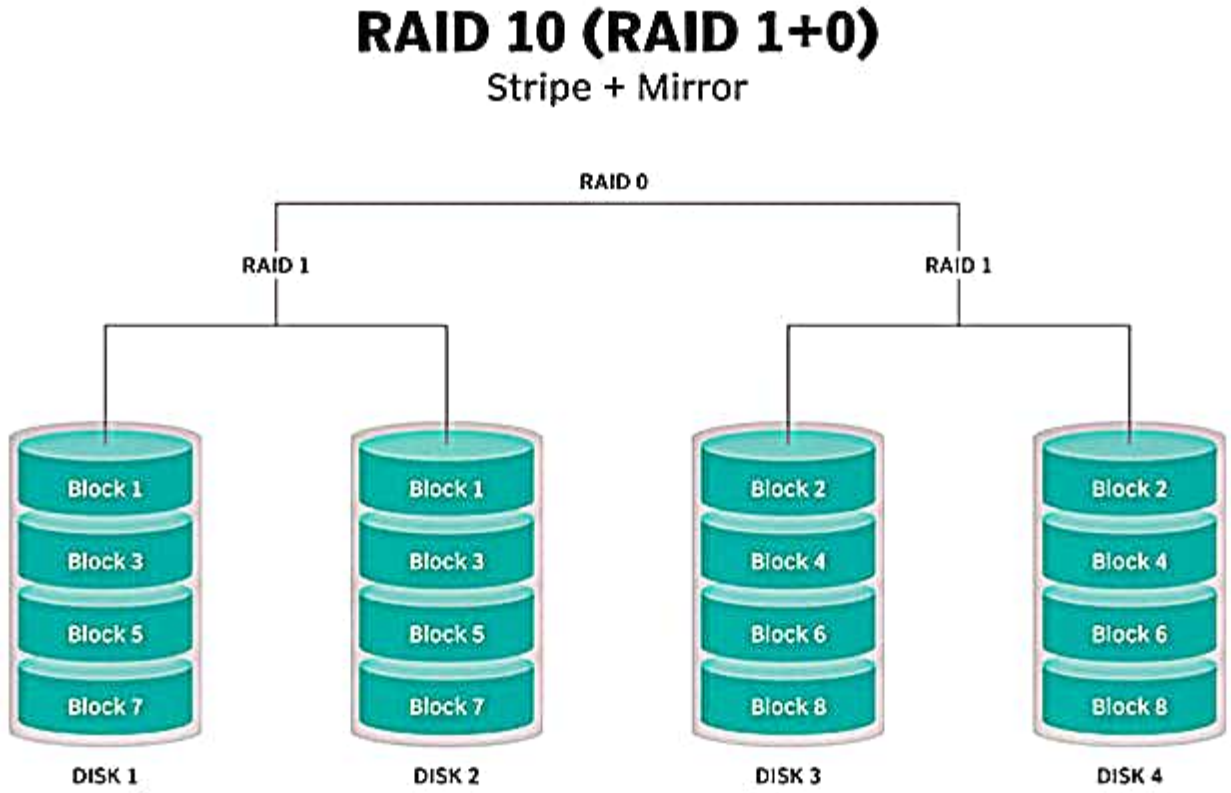
- Text file

A text file is a sequence of characters.

- Image file

An image file is a sequence of visual information, for example, vector art.

RAID 10 (RAID 1+0). Combining RAID 1 and RAID 0, this level is often referred to as RAID 10, which offers higher performance than RAID 1, but at a much higher cost. In RAID 1+0, the data is mirrored and the mirrors are striped.


RAID 10 (RAID 1+0)
Stripe + Mirror
RAID 0
RAID 1
RAID 1
Block 1
Block 3
Block 5
Block 7
Block 1
Block 3
Block 5
Block 7
Block 2
Block 4
Block 6
Block 8
Block 2
Block 4
Block 6
Block 8
DISK 1
DISK 2
DISK 3
DISK 4

Kernel I/O Subsystem in Operating System

The kernel provides many services related to I/O. Several services such as scheduling, caching, spooling, device reservation, and error handling – are provided by the kernel's I/O subsystem built on the hardware and device-driver infrastructure. The I/O subsystem is also responsible for protecting itself from errant processes and malicious users.

1. **I/O Scheduling** –
   To schedule a set of I/O requests means to determine a good order in which to execute them. The order in which the application issues the system call is the best choice. Scheduling can improve the overall performance of the system, can share device access permission fairly to all the processes, and reduce the average waiting time, response time, and turnaround time for I/O to complete.
   OS developers implement schedules by maintaining a wait queue of the request for each device.
   When an application issues a blocking I/O system call, The request is placed in the queue for that device. The I/O scheduler rearranges the order to improve the efficiency of the system.

2. **Buffering –**
   A buffer is a memory area that stores data being transferred between two devices or between a device and an application. Buffering is done for three reasons.
   1. The first is to cope with a speed mismatch between the producer and consumer of a data stream.
   2. The second use of buffering is to provide adaptation for data that have different data-transfer sizes.
   3. The third use of buffering is to support copy semantics for the application I/O, "copy semantic " means, suppose that an application wants to write data on a disk that is stored in its buffer. it calls the **write()** system's call, providing a pointer to the buffer and the integer specifying the number of bytes to write.

3. **Caching –**
   A *cache* is a region of fast memory that holds a copy of data. Access to the cached copy is much easier than the original file. For instance, the instruction of the currently running process is stored on the disk, cached in physical memory, and copied again in the CPU's secondary and primary cache.
   The main difference between a buffer and a cache is that a buffer may hold only the existing copy of a data item, while a cache, by definition, holds a copy on faster storage of an item that resides elsewhere.

4. **Spooling and Device Reservation** –
   A *spool* is a buffer that holds the output of a device, such as a printer that cannot accept interleaved data streams. Although a printer can serve only one job at a time, several applications may wish to print their output concurrently, without having their output mixes together.
   The OS solves this problem by preventing all output from continuing to the printer. The output of all applications is spooled in a separate disk file. When an application finishes printing then the spooling system queues the corresponding spool file for output to the printer.

5. **Error Handling** –
   An Os that uses protected memory can guard against many kinds of hardware and application errors so that a complete system failure is not the usual result of each minor mechanical glitch, Devices, and I/O transfers can fail in many ways, either for transient reasons, as when a network becomes overloaded or for permanent reasons, as when a disk controller becomes defective.
6. **I/O Protection** –
   Errors and the issue of protection are closely related. A user process may attempt to issue illegal I/O instructions to disrupt the normal function of a system. We can use the various mechanisms to ensure that such disruption cannot take place in the system.
   To prevent illegal I/O access, we define all I/O instructions to be privileged instructions. The user cannot issue I/O instruction directly.

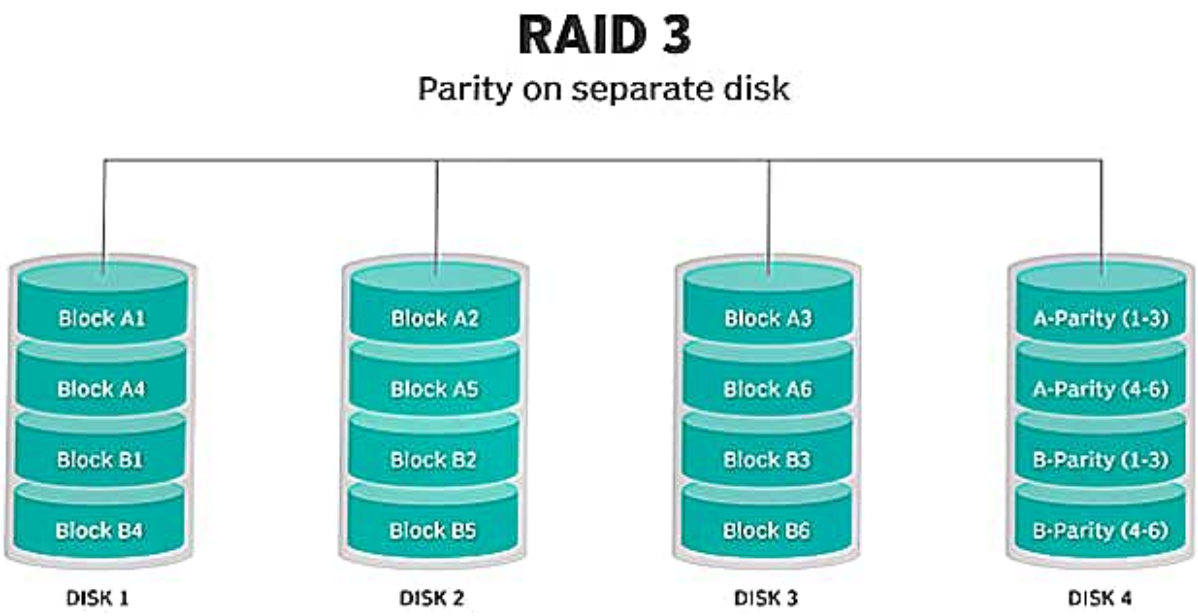
RAID 3. This technique uses striping and dedicates one drive to storing parity information. The embedded ECC information is used to detect errors. Data recovery is accomplished by calculating the exclusive information recorded on the other drives. Because an I/O operation addresses all the drives at the same time, RAID 3 cannot overlap I/O. For this reason, RAID 3 is best for single-user systems with long record applications.

Advantages of RAID 3

- Good throughput when transferring large amounts of data.
- High efficiency with sequential operations.
- Disk failure resiliency.

Disadvantages of RAID 3

- Not suitable for transferring small files.
- Complex to implement.
- Difficult to set up as software RAID.


RAID 3
Parity on separate disk
Block A1
Block A4
Block B1
Block B4
DISK 1
Block A2
Block A5
Block B2
Block B5
DISK 2
Block A3
Block A6
Block B3
Block B6
DISK 3
A-Parity (1-3)
A-Parity (4-6)
B-Parity (1-3)
B-Parity (4-6)
DISK 4

RAID 5. This level is based on parity block-level striping. The parity information is striped across each drive, enabling the array to function, even if one drive were to fail. The array's architecture enables read and write operations to span multiple drives. This results in performance better than that of a single drive, but not as high as a RAID 0 array. RAID 5 requires at least three disks, but it is often recommended to use at least five disks for performance reasons.

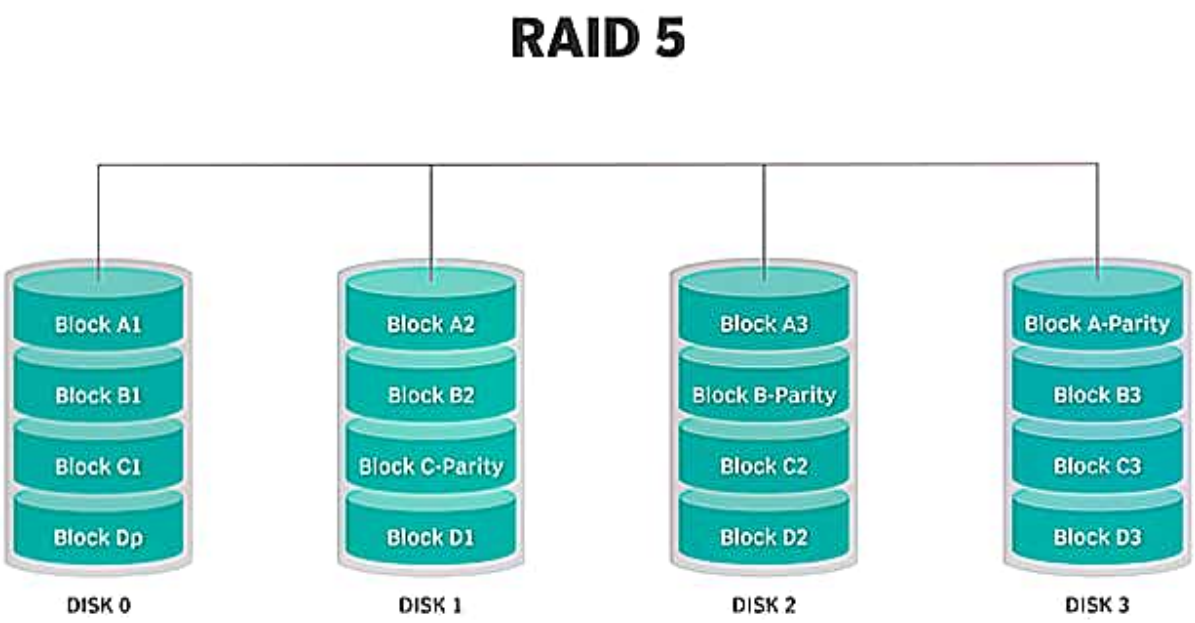
RAID 5 arrays are generally considered to be a poor choice for use on write-intensive systems because of the performance impact associated with writing parity data. When a disk fails, it can take a long time to rebuild a RAID 5 array.

Advantages of RAID 5

- High performance and capacity.
- Fast and reliable read speed.
- Tolerates single drive failure.

Disadvantages of RAID 5

- Longer rebuild time.
- Uses half of the storage capacity (due to parity).
- If more than one disk fails, data is lost.
- More complex to implement.


RAID 5
Block A1
Block B1
Block C1
Block Dp
DISK 0
Block A2
Block B2
Block C-Parity
Block D1
DISK 1
Block A3
Block B-Parity
Block C2
Block D2
DISK 2
Block A-Parity
Block B3
Block C3
Block D3
DISK 3

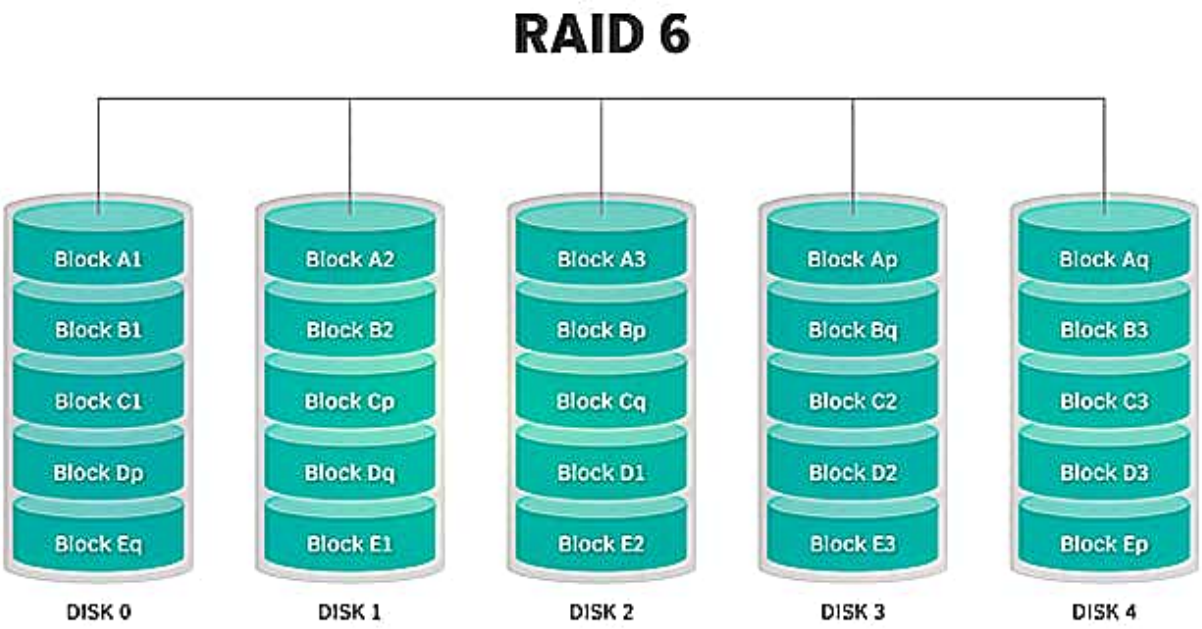
RAID 6. This technique is similar to RAID 5, but it includes a second parity scheme distributed across the drives in the array. The use of additional parity enables the array to continue functioning, even if two disks fail simultaneously. However, this extra protection comes at a cost. RAID 6 arrays often have slower write performance than RAID 5 arrays.

Advantages of RAID 6

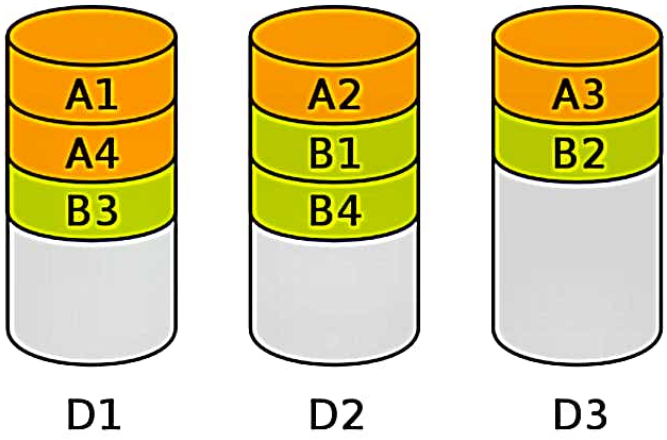
- High fault and drive-failure tolerance.
- Storage efficiency (when more than four drives are used).
- Fast read operations.

Disadvantages of RAID 6

- Rebuild time can take up to 24 hours.
- Slow write performance.
- Complex to implement.
- More expensive.


RAID 6
Block A1
Block B1
Block C1
Block Dp
Block Eq
Block A2
Block B2
Block Cp
Block Dq
Block E1
Block A3
Block Bp
Block Cq
Block D1
Block E2
Block Ap
Block Bq
Block C2
Block D2
Block E3
Block Aq
Block B3
Block C3
Block D3
Block Ep
DISK 0
DISK 1
DISK 2
DISK 3
DISK 4

In computer data storage, **data striping** is the technique of segmenting logically sequential data, such as a file, so that consecutive segments are stored on different physical storage devices.



An example of data striping. Files A and B, of four blocks each are spread over disks D1 to D3.

Striping is useful when a processing device requests data more quickly than a single storage device can provide it. By spreading segments across multiple devices which can be accessed concurrently, total data throughput is increased. It is also a useful method for balancing I/O load across an array of disks. Striping is used across disk drives in redundant array of independent disks (RAID)

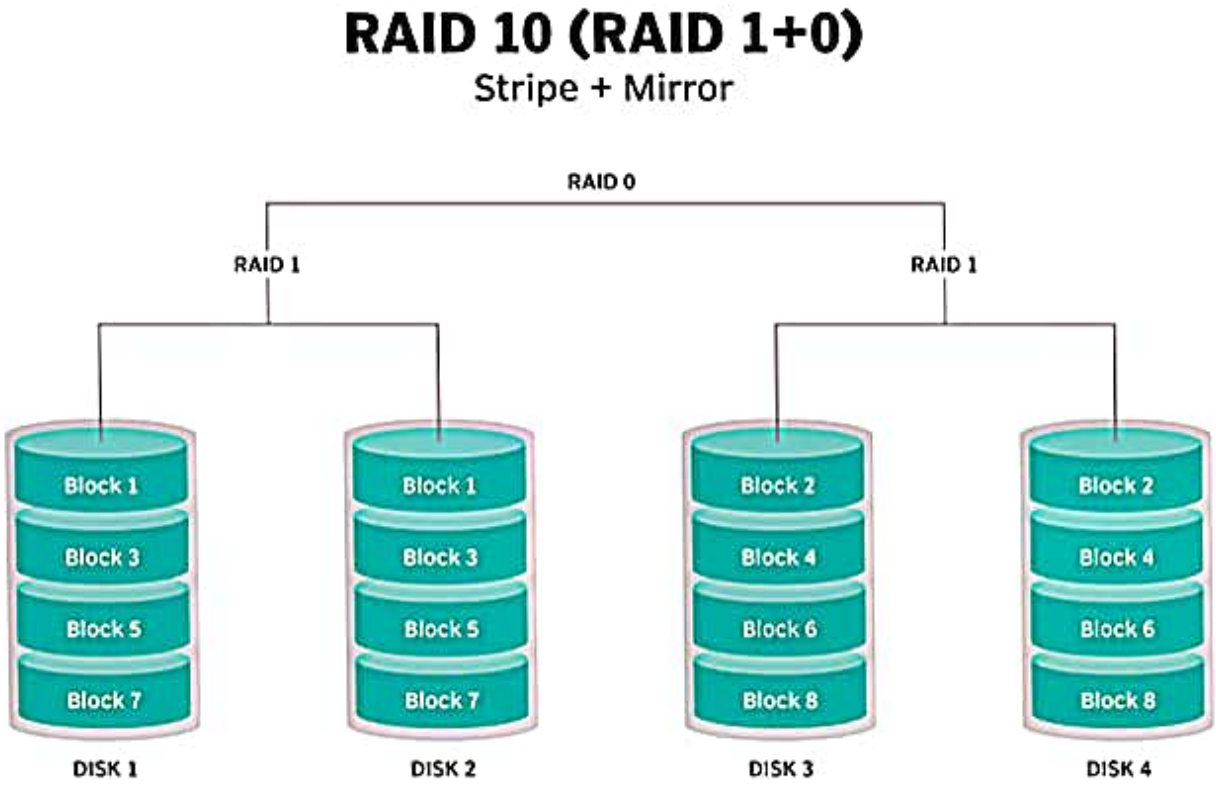
What is RAID 10?

RAID 10, also known as RAID 1+0, is a RAID configuration that combines disk mirroring and disk striping to protect data. It requires a minimum of four disks and stripes data across mirrored pairs. As long as one disk in each mirrored pair is functional, data can be retrieved. If two disks in the same mirrored pair fail, all data will be lost because there is no parity in the striped sets.

RAID, which stands for redundant array of independent disks, comes in several different configurations. A RAID 1 configuration copies data from one drive to another, mirroring and duplicating data to provide improved fault tolerance and data protection. Data is fully protected as the mirror copy is available if the originating drive is disabled or unavailable. Because it makes a full duplicate of the data, RAID 1 requires twice as much storage capacity as the original data.

RAID 0 doesn't provide any data protection; its sole purpose is to enhance drive access performance. It does that by spreading the data out across two or more drives. That way multiple read/write heads on the drives can write or access portions of data simultaneously, thus speeding up overall processing.

RAID 10 provides data redundancy and improves performance. It is the a good option for I/O-intensive applications -- including email, web servers, databases and operations that require high disk performance. It's also good for organizations that require little to no downtime.


RAID 10 (RAID 1+0)
Stripe + Mirror
RAID 0
RAID 1
RAID 1
Block 1
Block 3
Block 5
Block 7
Block 1
Block 3
Block 5
Block 7
Block 2
Block 4
Block 6
Block 8
Block 2
Block 4
Block 6
Block 8
DISK 1
DISK 2
DISK 3
DISK 4