

SOLVED PAPERS (2011-12 TO 2018-19) (SR-IB to SR-30B)

**SHORT QUESTIONS**

I/O Management, I/O buffering, Disk storage and disk scheduling mechanisms, File structure, File organization and access methods, File sharing, File system protection and security.

**UNIT-5 : I/O MANAGEMENT AND DISK SCHEDULING**

Multiprogramming with fixed partitions, Multiprogramming with variable partitions, Protection schemes, Paging Segmentation, Pageed memory Cache memory organization, Locality of reference.

**UNIT-4 : MEMORY MANAGEMENT**

Memory management Basic bare machine, Resident monitor, Deadlock detection, Recovery from deadlocks.

**UNIT-3 : CPU SCHEDULING**

Processor / Consumer Problem, Principle of concurrency, Concurrency of processes, Process Concept, Deadlock Solution, Deadlock Detection, Deadlock Prevention, Deadlock Avoidance, Deadlock Deadlock Resolution, Deadlock Detection, Deadlock Prevention, Deadlock Avoidance, Deadlock Prevention.

**UNIT-2 : CONCURRENT PROCESSES**

Resource Kernel, Monolithic and Microkernel Systems, Resource Reclamation Problem, Sleeping Barrier Problem, Inter-Process Communication, Inter-Process Problem, Mutual Exclusion, Critical Section Problem, Dekker's solution, Peterson's Solution, Smaphores, Test-and-set operation, Classical Problem in Concurrency, Dining Philosophers problem, Producer-consumer Problem, Real-time processes.

**UNIT-1 : COMPUTER SYSTEMS**

CONTENTS  
RCS 401 : OPERATING SYSTEMS

Price: Rs. 120/- only

11th Edition : 2018-19

10th Edition : 2017-18

9th Edition : 2016-17

8th Edition : 2015-16

7th Edition : 2014-15

6th Edition : 2013-14

5th Edition : 2012-13

4th Edition : 2011-12

3rd Edition : 2010-11

2nd Edition : 2009-10

**Operating System (CS : Sem-4)**

Information contained in this work is derived from sources believed to be reliable. Every effort has been made to ensure accuracy or completeness of any information published herein, and neither the publisher nor the authors shall be responsible for any errors, omissions, or damages arising out of use of this information.

No part of this publication may be reproduced or transmitted, in any form or by any means, without permission.

© All Rights Reserved

Delhi Office : 1-6590, East Rohnas Nagar, Shahdara, Delhi-110032  
Email : pagequantum@gmail.com Website : www.pagequantum.com

Phone : 0120-4160479  
Quantum Page Pvt. Ltd.  
Plot No. 59/27, Site - 4, Industrial Area,  
Shahdabad, Ghaziabad-201 010

PUBLISHED BY :

Param Singh

2

The operating systems perform the following functions:

- Functions of an operating system :
- 1. An operating system acts as an intermediary between the user of a computer and computer hardware.
- 2. An operating system is software that manages the computer hardware.
- 3. The hardware must provide appropriate mechanisms to ensure the correct operation of the computer system and to prevent user programs from interfering with the proper operation of the system.

**Answer**

**Ques 1.1** What is operating system? Write the major functions of an operating system.

**Long Answer Type and Medium Answer Type Questions**

**Questions-Answers**

- A multi-processor is a computer system with two or more central processing units, with each one sharing the common main memory as well as peripherals.
- The feature of a batch system is the lack of interaction between the user and the job while the job is executing.
- The feature of a real time system is the lack of interaction between the user and the job while the job is executing.
- Multi-processing management
- Device management
- Processor management
- Memory management
- Function of operating system :
- Between a user of a computer and the computer hardware.
- An operating system is a program that acts as an intermediary between a user of a computer and the computer hardware.

**CONCEPT OUTLINE : PART-1**

**PART-1**

Introduction : Operating System and Function  
Time System, Multi-processor Systems, Multi-threaded Systems,  
Multi-tasking Systems, Multi-processor Sharing Systems, Real  
Time-Sharing, Interrelative, Time-Sharing, Multi-user Systems,  
Batch, Function of Operating System : Batch, Function of  
Operating System, Multi-processor Systems, Multi-threaded Systems,

A. Concept Outline : Part-2 ..... 1-9B  
B. Long and Medium Answer Type Questions ..... 1-10B

**Part-2** ..... (1-9B to 1-20B)

**Long Answer Type and Medium Answer Type Questions**

**Questions-Answers**

A. Concept Outline : Part-1 ..... 1-2B  
B. Long and Medium Answer Type Questions ..... 1-2B

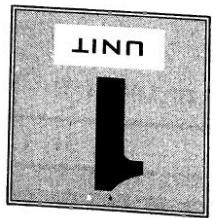
**Part-1** ..... (1-2B to 1-9B)

**Introduction : Operating System and Function**

**• Classification of Operating System : Batch**  
**• Interrelative**  
**• Time-Sharing**  
**• Real Time System**  
**• Multi-user Systems**  
**• Multi-processor Systems**  
**• Multi-threaded Systems**  
**• Multi-processing Systems**  
**• File Management**  
**• Device Management**  
**• Processor Management**  
**• Memory Management**  
**• Function of operating system :**

**Part-1** ..... (1-2B to 1-9B)

## Computer System





S.No.	Multi-processing	Multi-programming	Multi-programming
3.	It permits parallel processing.	Context switching takes place.	
2.	It utilizes single CPU.	It utilizes multiple CPUs.	
1.	Multi-processing refers to simultaneous execution of multiple programs in main memory at the same time and executes them concurrently utilizing single CPU.	Programs in main memory are interleaved and executed sequentially using multiple CPUs.	

Answer

- Ques 1.7** Write down the difference between multi-programming and multi-processing system  
**AKTU 2013-14, Marks 06**
- and multi-programming operating system.
6. Thus, in multi-threaded applications, multiple tasks can process in parallel with other applications that are running on the system.
5. We can prioritize each of these, so that they operate independently.
4. In a multi-threaded program, for example application might be divided into four threads : a user interactive thread, a data acquisition thread, application threads, but also among each thread an application.
3. The OS divides processing time not only among different applications, but also among each thread within an application.
2. Each of the threads can run in parallel.
1. Multi-threading extends the idea of multi-tasking into applications, so we can sub-divide specific operations within a single application into individual threads.
- ii. Multi-threading :**
1. Multi-programming : Refer Q. 1.2, Page 1-3B, Unit-1.

Answer

- Ques 1.8** Explain the following terms clearly:  
**AKTU 2011-12, Marks 06**
- i. Multi-programming
- Properties of time-sharing operating system:
- a. In symmetric multi-processing, each processor runs identical copy of the operating system and they communicate with one another as needed. All the CPU share the common memory.
  - b. In asymmetric multi-processing, each processor is assigned a specific task it uses master-slave relationship.
- ii. Multi-threading
- Properties of real time operating system:
- 1. Interactivity of a system.
  - 2. The CPU switches rapidly from one user to another i.e., the CPU is shared between a number of interactive users.
  - 3. Instead of having a job defined by spooler card images, each program reads its next control instructions from the terminal and output is normally printed immediately on the screen.
  - 4. Real time systems are usually dedicated, embedded systems.
  - 5. They typically read from and react to sensor data.
  - 6. The system must guarantee response to events within fixed periods of time to ensure correct performance.
  - 7. Distributes computation among several physical processors.
  - 8. They do not share memory or a clock, instead, each processor has its own local memory.
  - 9. They communicate with each other through various communication lines.
- iii. Multi-processor system**
- Properties of distributed operating system:
- 1. Distributes computation among several processors.
  - 2. The processors do not share memory or a clock, instead, each processor has its own local memory.
  - 3. Multi-processor systems are often used to share resources to enable better utilization of resources and some times memory.
  - 4. Multi-processor system consists of two types : symmetric multi-processing and asymmetric multi-processing.
  - 5. In multi-processing system, it is possible for two processes to run in parallel.
  - 6. Multi-processor systems have more than one processor in close communication. They share the computer bus, system and input-output devices and sometimes memory.
  - 7. Multi-processor system : Refer Q. 1.2, Page 1-3B, Unit-1.

Answer

- 1-6-B (CS/IT-Sem-4) Computer System

- Ques 1.9** Explain the following terms:  
**AKTU 2013-14, Marks 06**
- i. Multi-programming system
- Properties of real time operating system:
- 1. Interactivity of a system.
  - 2. The CPU switches rapidly from one user to another i.e., the CPU is shared between a number of interactive users.
  - 3. Instead of having a job defined by spooler card images, each program reads its next control instructions from the terminal and output is normally printed immediately on the screen.
  - 4. Real time systems are usually dedicated, embedded systems.
  - 5. They typically read from and react to sensor data.
  - 6. The system must guarantee response to events within fixed periods of time to ensure correct performance.
  - 7. Distributes computation among several physical processors.
  - 8. They do not share memory or a clock, instead, each processor has its own local memory.
  - 9. They communicate with each other through various communication lines.
- ii. Multi-processor system
- Properties of distributed operating system:
- 1. Distributes computation among several processors.
  - 2. The processors do not share memory or a clock, instead, each processor has its own local memory.
  - 3. Multi-processor systems have more than one processor in close communication. They share the computer bus, system and input-output devices and sometimes memory.
  - 4. Multi-processor system consists of two types : symmetric multi-processing and asymmetric multi-processing.
  - 5. In multi-processing system, it is possible for two processes to run in parallel.
  - 6. Multi-processor systems have more than one processor in close communication. They share the computer bus, system and input-output devices and sometimes memory.
  - 7. Multi-processor system : Refer Q. 1.2, Page 1-3B, Unit-1.

Answer

- Ques 1.10** Discuss essential properties of time-sharing system and distributed operating system.  
**AKTU 2012-13, Marks 06**
- Properties of time-sharing operating system:
- 1. Users CPU scheduling and multi-processing to provide economical interactive use of a system.
  - 2. The CPU switches rapidly from one user to another i.e., the CPU is shared between a number of interactive users.
  - 3. Instead of having a job defined by spooler card images, each program reads its next control instructions from the terminal and output is normally printed immediately on the screen.
  - 4. Real time systems are usually dedicated, embedded systems.
  - 5. They typically read from and react to sensor data.
  - 6. The system must guarantee response to events within fixed periods of time to ensure correct performance.
  - 7. Distributes computation among several physical processors.
  - 8. They do not share memory or a clock, instead, each processor has its own local memory.
  - 9. They communicate with each other through various communication lines.
- Ques 1.11** Discuss essential properties of time-sharing system and distributed operating system.  
**AKTU 2012-13, Marks 06**
- Properties of distributed operating system:
- 1. They communicate with each other through various communication lines.
  - 2. They share the computer bus, system and input-output devices and sometimes memory.
  - 3. They share the computer bus, system and input-output devices and sometimes memory.
  - 4. They share the computer bus, system and input-output devices and sometimes memory.
  - 5. They share the computer bus, system and input-output devices and sometimes memory.
  - 6. They share the computer bus, system and input-output devices and sometimes memory.
  - 7. They share the computer bus, system and input-output devices and sometimes memory.
  - 8. They share the computer bus, system and input-output devices and sometimes memory.
  - 9. They share the computer bus, system and input-output devices and sometimes memory.

Answer

- 1-5-B (CS/IT-Sem-4)

- Ques 1.12

WPS-AUHING 2012

buffering? What are the advantages of buffering over

OR

Ques 1.9. What is spooling?

S.No.	Batch processing system	Multi-programming system
1.	In batch systems, the programs are submitted to the computer in the form of job control instructions, data and some instructions, data and some operators in the form of job control instructions, data and some programs, and hence multiple concurrent execution of multiple programs is allowed.	programs, and hence multiple concurrent execution of multiple programs is allowed.

Answer

- Ques 1.10. Differentiate between batch processing system and multi-programming system with example.
- Answer
1. The spooling is capable of overlapping I/O operation for one job with processor operations for another job.
2. Spooling is capable of overlapping I/O operation for one job with processor operations for another job.
- Advantages of spooling over buffering:
1. Spooling is also used for processing data at remote sites. The CPU sends the data via communication lines to a remote site. Then the printer pulls them off the buffer at its own rate.
  2. In print spooling, documents are loaded into a buffer and then the printer pulls them off the buffer at its own rate.
  3. Spooling is also used for processing application data at different rates.
  4. The most common spooling application is print spooling.
  5. The most common spooling application is print spooling.
  6. In print spooling, documents are loaded into a buffer and then the printer pulls them off the buffer at its own rate.
  7. Spooling is also used for processing data at remote sites. The CPU sends the data via communication lines to a remote site. Then the printer pulls them off the buffer at its own rate.
  8. Spooling is a waiting station where data can reset while the slower device catches up.
  9. Spooling is useful because it provides a waiting station where data can access data at different rates.

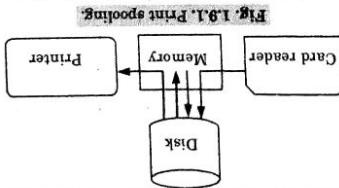


Fig 1.9. Print spooling

- Answer
1. SPoolING is acronym for Simultaneous Peripheral Operations Online.
2. 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.

1-B (CS/IT-Sem-4) Computer System

What is spooling? What are the advantages of spooling over buffering?

OR

Ques 1.9. What is spooling?

S.No.	Base	Symmetric multi-processor is simple.
3.	Failure	If a processor fails, the computing capability of the system reduces to the master processor to handle the task. If a slave processor fails, its task is switched to the master processor. If a slave processor fails, its task is switched to the master processor to handle the task. If a master processor fails, a slave processor takes over the job.
4.	Asymmetric multi-processor is simple.	If a master processor is unable to handle the job, a slave processor takes over the job.

Answer

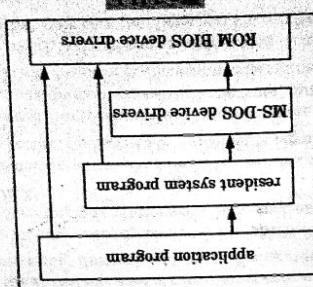
Ques 1.8. Describe the differences between symmetric and asymmetric multi-processing.

Answer

1. Architecture	All processor in symmetric multi-processing	All processor in asymmetric multi-processing
2. Communication	All processors need not to communicate as they have same architecture.	Processors need to communicate as they have different architecture.
3. Failure	If a processor fails, the shared memory is controlled by a master processor.	If a processor fails, a slave processor takes over the job.
4. Asymmetry	Asymmetry is shared memory.	Asymmetry is shared memory.
5.	Usually more expensive.	Such systems are less expensive.

Operating Systems

Operating Systems	
• Operating system service :	i-10 B (CS/IT-Sem-4)
• i. Program execution ii. LO operation iii. File system manipulation iv. Error detection v. Protection and security	a. Several manners, a concurrency control mechanism b. File management or prioritizing system c. File management also accessed in a very simple manner in a batch system.
Ques 111. What is an operating system ? Discuss the operating system structure.	The users are not allowed to interact with the computer in such a way that the CPU remains busy for maximum about of time. The programs should be scheduled in operating system.
Answer	Ques 111. What is an operating system ? Discuss the operating system structure. Explaining layered structure of an operating system. Also explaining advantages and disadvantages of the layered approach to system design.
Ques 111. Structure of OS : Refer Q. 1.1, Page 1-2B, Unit-1.	1. Simple structure: MS-DOS written to provide the most functionality in the least space . 2. Althrough MS-DOS has some structure, its interfaces and levels of Not divided into modules.
Structure of OS :	b. Although MS-DOS has some structure, its interfaces and levels of functionality are not well separated.



CONCEPT OUTLINE : PART-2	
• Structure of OS :	i. Simple structure ii. Layered approach iii. Micro-kernels iv. Modules
• System components :	i. Memory management ii. Processor management iii. Device management iv. File management

PART-2	
Operating Systems, Components, Operating System, Services, Real-time Kernel, Monolithic and Micro-kernel System.	Qorulating System Structure, Layered Structure, System Components, Services, Real-time Kernel, Monolithic and Micro-kernel System.

2. Only one program is in Multi-programming operating system allows sharing of UO execution at a time, any time - critical devices among multiple users, more sophisticated UO management is required.	Only one program is in Multi-programming operating system allows sharing of UO execution at a time, any time - critical devices among multiple users, more sophisticated UO management is required.
3. Since files are also accessed in several manners, a concurrency and control mechanism is not required, making access is not required, making control mechanisms advanced protection, simple matter in a batch file management also a very simple manner, file management or prioritizing system.	File management or prioritizing system.
4. The users are not allowed to interact with the computer in such a way that the CPU remains busy for maximum about of time. The programs should be scheduled in operating system.	The users are not allowed to interact with the computer in such a way that the CPU remains busy for maximum about of time. The programs should be scheduled in operating system.
5. Example: Execution of batch files such as autoexec.bat by the same computer.	Example: Execution of batch files such as autoexec.bat by the same computer.

## 1-12B (CS/IT-Sem-4) Computer System

a. The main advantage of the layered approach is simplicity of construction

1. The layers are selected so that each uses functions (operations) and services of only lower-level layers.

2. This approach simplifies debugging and system verification.

3. The first layer can be debugged without any concern for the rest of the system, because it uses only the basic hardware to implement its functions.

4. The first layer can be debugged, its correct functioning can be assumed while the second layer is debugged, and so on.

5. Once the first layer is debugged, its correct functioning can be assumed while the second layer is debugged, and so on.

6. If an error is found during the debugging of a particular layer, the error must be on that layer, because the layers below it are already debugged.

7. Thus, the design and implementation of the system is simplified.

**Disadvantages :**

1. The major difficulty with the layered approach involves appropriate layering, careful planning is necessary.

2. A final problem with layered implementation is that they tend to be less efficient than other types. At each layer, the parameters may be modified, data may need to be passed, and so on. Each layer adds overhead to the system call; the net result is a system call that takes longer than one on a non-layered system.

3. These limitations have caused a small backlash against layering. However, layers with more functionality are being designed, providing most of the advantages of modularized code while avoiding the difficult problems of layer definition and interaction.

4. Services like kernel has a set of core components and links in addition to services of other duration time.

b. Such a strategy uses dynamically loadable modules and is common in modern implementations of UNIX, such as Solaris, Linux, and Mac OS X.

c. Such a design allows the kernel to provide core services yet also allows certain features to be implemented dynamically.

d. The overall result resembles a layered system in that each kernel module has only core functions and knowledge of how to load and unload a layered system in that any module can call any other module.

e. The approach is like the microkernel approach in that the primary section has been removed, proceeded system can call any other module.

f. The kernel has a set of core components and links in addition to services of other duration time.

g. Moves as much from the kernel into "user" space communication modules:

h. Takes place between user modules using message passing.

i. Here the kernel has a set of core components and links in addition to services of other duration time.

j. Such a strategy uses dynamically loadable modules and is common in modern implementations of UNIX, such as Solaris, Linux, and Mac OS X.

k. Allows certain features to be implemented dynamically.

l. The overall result resembles a layered system in that each kernel module has only core functions and knowledge of how to load and unload a layered system in that any module can call any other module.

m. The approach is like the microkernel approach in that the primary section has been removed, proceeded system can call any other module.

n. The overall result resembles a layered system in that each kernel module has only core functions and knowledge of how to load and unload a layered system in that any module can call any other module.

o. The overall result resembles a layered system in that each kernel module has only core functions and knowledge of how to load and unload a layered system in that any module can call any other module.

p. The overall result resembles a layered system in that each kernel module has only core functions and knowledge of how to load and unload a layered system in that any module can call any other module.

q. The overall result resembles a layered system in that each kernel module has only core functions and knowledge of how to load and unload a layered system in that any module can call any other module.

r. The overall result resembles a layered system in that each kernel module has only core functions and knowledge of how to load and unload a layered system in that any module can call any other module.

s. The overall result resembles a layered system in that each kernel module has only core functions and knowledge of how to load and unload a layered system in that any module can call any other module.

t. The overall result resembles a layered system in that each kernel module has only core functions and knowledge of how to load and unload a layered system in that any module can call any other module.

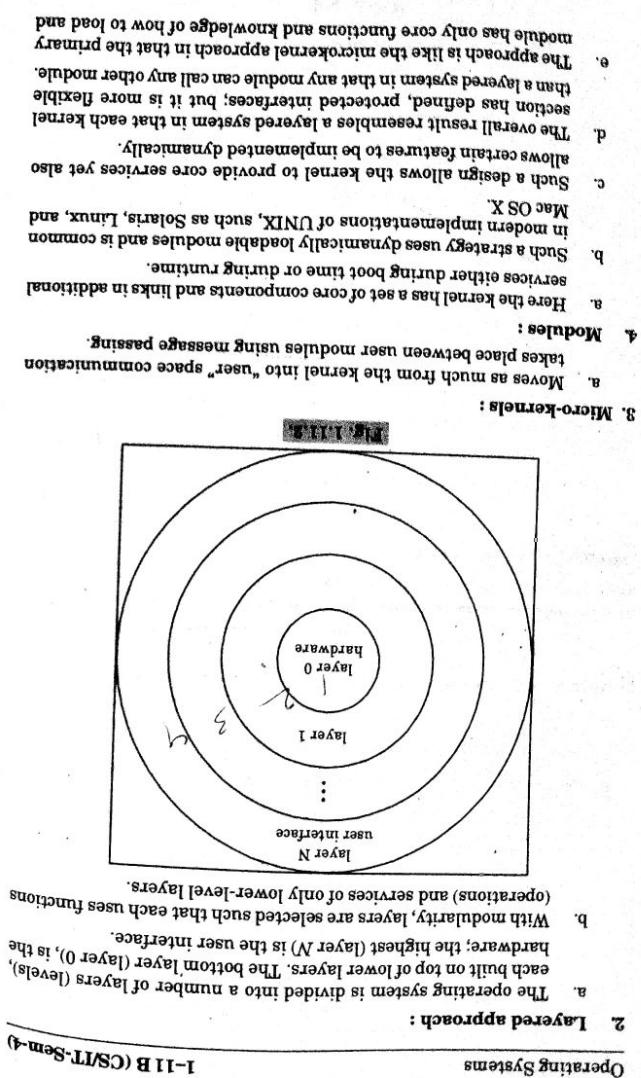
u. The overall result resembles a layered system in that each kernel module has only core functions and knowledge of how to load and unload a layered system in that any module can call any other module.

v. The overall result resembles a layered system in that each kernel module has only core functions and knowledge of how to load and unload a layered system in that any module can call any other module.

w. The overall result resembles a layered system in that each kernel module has only core functions and knowledge of how to load and unload a layered system in that any module can call any other module.

x. The overall result resembles a layered system in that each kernel module has only core functions and knowledge of how to load and unload a layered system in that any module can call any other module.

y. The overall result resembles a layered system in that each kernel module has only core functions and knowledge of how to load and unload a layered system in that any module can call any other module.



## Operating Systems

I-13B (CS/IT-Sem-4)

Components of an operating system are :

1. Memory management :

An operating system does the following activities for memory management :

a. Keeps tracks of primary memory, i.e., what part of it are in use by whom, what part are not in use.

b. In multi-programming, the OS 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 :

An operating system does the following activities for processor management :

a. Keeps tracks of processor and status of processes. The program responsible for this task is known as the CPU controller.

b. Allocates the processor (CPU) to a process.

c. De-allocates the processor when a process is no longer required.

d. Allocates the processor when a process gets the service when and for how much time.

3. File management :

An operating system manages device communication via their respective drivers. It does the following activities for device management :

a. Keeps tracks of all devices. Program responsible for this task is known as the I/O controller.

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

c. Allocates the device in the efficient way.

4. Device management :

An operating system performs the following activities for file management :

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

b. Decides who gets the resources.

c. Allocates the resources.

d. De-allocates the resources.

5. Error detection :

An operating system does the following activities for file management :

a. The main function of operating system is to detect the errors like bad sectors on hard disk, memory overflow and errors related to I/O devices.

b. After detecting the errors, operating system takes an appropriate action for consistent computing.

## Ques 1.18 Define the services provided by the operating system.

### Answer

I-14B (CS/IT-Sem-4)

AKTU 2016-17, Marks 10

Computer System

Explain the services provided by operating system are as follows :

a. The purpose of computer system is to allow the users to execute programs in an efficient manner.

b. The operating system provides an environment where the user can conveniently run these programs.

c. To run a program, the program is required to be loaded into the RAM first and then to assign CPU time for its execution.

d. De-allocates the memory when a process no longer needs it or has been terminated.

e. Allocates the memory when a process requests it to do so.

f. Keeps tracks of processor and status of processes. The program responsible for this task is known as the CPU controller.

g. Allocates the processor (CPU) to a process.

h. Decides which processor gets the service when and for how much time.

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

j. The I/O service cannot be provided by user-level programs and it must be provided by the operating system.

k. This involves the use of I/O devices.

l. Each program requires an input and after processing the input sumitted by user it produces output.

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

n. This is an important task that is also performed by the operating system.

o. Manipulates various types of files like opening a file, saving a file and deleting a file from the storage disk.

p. Operates various types of shared memory, in which processes read and write to a shared section of memory.

q. Communication may be implemented via shared memory, in which formats are moved between processes by the operating system or message passing, in which packets of information in predefined formats are moved between processes by the operating system.

r. The main function of operating system is to detect the errors like bad sectors on hard disk, memory overflow and errors related to I/O devices.

s. Provides the services of shared memory among various applications.

t. Provides the services of shared memory among various users.

u. Provides the services of shared memory among various processes.

v. Provides the services of shared memory among various threads.

w. Provides the services of shared memory among various processes.

x. Provides the services of shared memory among various threads.

y. Provides the services of shared memory among various processes.

z. Provides the services of shared memory among various threads.

aa. Provides the services of shared memory among various processes.

ab. Provides the services of shared memory among various threads.

ac. Provides the services of shared memory among various processes.

ad. Provides the services of shared memory among various threads.

ae. Provides the services of shared memory among various processes.

af. Provides the services of shared memory among various threads.

ag. Provides the services of shared memory among various processes.

ah. Provides the services of shared memory among various threads.

ai. Provides the services of shared memory among various processes.

aj. Provides the services of shared memory among various threads.

ak. Provides the services of shared memory among various processes.

al. Provides the services of shared memory among various threads.

am. Provides the services of shared memory among various processes.

an. Provides the services of shared memory among various threads.

ao. Provides the services of shared memory among various processes.

ap. Provides the services of shared memory among various threads.

aq. Provides the services of shared memory among various processes.

ar. Provides the services of shared memory among various threads.

as. Provides the services of shared memory among various processes.

at. Provides the services of shared memory among various threads.

au. Provides the services of shared memory among various processes.

av. Provides the services of shared memory among various threads.

aw. Provides the services of shared memory among various processes.

ax. Provides the services of shared memory among various threads.

ay. Provides the services of shared memory among various processes.

az. Provides the services of shared memory among various threads.

ba. Provides the services of shared memory among various processes.

bb. Provides the services of shared memory among various threads.

bc. Provides the services of shared memory among various processes.

bd. Provides the services of shared memory among various threads.

be. Provides the services of shared memory among various processes.

bf. Provides the services of shared memory among various threads.

bg. Provides the services of shared memory among various processes.

bh. Provides the services of shared memory among various threads.

bi. Provides the services of shared memory among various processes.

bj. Provides the services of shared memory among various threads.

bk. Provides the services of shared memory among various processes.

bl. Provides the services of shared memory among various threads.

S.No.	Shell	Kernel
1.	When a user logs in, the login program checks the user name and password, and then starts another program called the shell.	The kernel is the hub of the operating system. It allocates memory to programs and handles them to respond to system calls.
2.	Kernel gives the hardware information with user and kernel interface.	The kernel is the part of the operating system that interacts with the hardware.

Answer

AKTU 2014-15, Marks 05

What are the differences between shell and kernel?

OR

AKTU 2011-12, Marks 05

i. Monolithic kernel and micro-kernel  
ii. Shell and Kernel  
iii. Difference between the following:

Ques 1.6. Difference between the following:

- Concurrent executions of re-enterant functions do not affect the behaviour of one another. The operating system makes sure that non-reentrant functions are executed mutually exclusively by the kernel.
- Non-reentrant executions of re-enterant functions may modify local data, but they do not modify any global data.
- Normality the operating system is composed of re-enterant and non-operating systems.
- Concurrent executions of re-enterant functions do not affect the behaviour of one another. The operating system makes sure that non-reentrant functions may modify local data, but they do not modify any global data.
- Kernel provides the means to execute the programs, nor may it share the memory or hardware resources. The re-enterant functions may modify local data, but they do not modify any global data.
- Kernel is the main component of most computer systems.

Answer

1-16 B (CS/IT-Sem-4) Computer System

- Ques 1.7. Describe various operations performed by the kernel:
- It controls the state of the process whether the process is running or process is waiting for the request of the user.
  - It provides the memory for the processes those checks whether the process is running or process is waiting for the request of the user.
  - The kernel also maintains a time table for all the processes those are running or processes those are running on the system that is kernel runs the allocation and deallocation of process.
  - The kernel also manages a time table for the processes those are running or processes those are running on the system that is kernel runs the waiting and suspended jobs into the different memory area.
  - When a kernel determines that the logical memory does not fit to store the programs, then it uses the concept of the physical memory which will store the programs into temporary manner. Therefore, physical memory of the system can be used as temporary memory.

Ques 1.8. Describe re-entrant kernels.

- The kernel puts the waiting and suspended jobs into the different memory areas.
- The kernel also prepares the schedule time which will provide the time to various processes of the CPU.
- The kernel also manages a time table for all the processes those are running or processes those are running on the system that is kernel runs the allocation and deallocation of process.
- The kernel runs the processes those are running on the system that is kernel runs the waiting for the request of the user.
- It provides the lowest-level abstraction layer for the resources (especially processors and I/O devices) that application software must control to perform its function.
- Kernel provides the lowest-level abstraction layer for the resources (especially processors and I/O devices) that application software must control to perform its function.
- Kernel is considered as the heart of an operating system.
- It provides an interface between applications and actual data processing at the hardware level.
- Kernel is the main component of most computer systems.

- Ques 1.8. What is kernel? Describe various operations performed by kernel?
- Answer

- 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.
  - Protection and security:
- a. Protection involves ensuring that all access to system resources is controlled.
- b. 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.
- c. Protection and security:
- d. At a time, it is the responsibility of an operating system to allocate secondary storage etc., to each process for its better utilization.
- e. In the multitasking environment, when multiple jobs are running at the required resources (like CPU, main memory), tape drive or secondary storage etc., to each process for its better utilization.
- f. Protection and security:
- g. In the same kernel, a process does not modify kernel programs, but may be allowed to execute the programs, nor may it share the memory or hardware resources. The re-enterant functions may modify local data, but they do not modify any global data.
- h. In the same kernel, a process can execute in the non-reentrant kernel, a process does not modify kernel programs, but may be allowed to execute the programs, nor may it share the memory or hardware resources. The re-enterant functions may modify local data, but they do not modify any global data.

6. Resource allocation:

1-16 B (CS/IT-Sem-4)

Operating Systems

### iii. Monolithic kernel and micro-kernel : Operating Systems

S.No.	Monolithic kernel	Micro-kernel
1.	Kernel size is large.	Kernel size is small.
2.	OS is complex to design.	OS is easy to design, implement and install.
3.	All the operating system services are included in the kernel.	Kernel provides only TCP and low level device management services.
4.	Request may be serviced faster.	Request may be serviced slower than monolithic kernel.
5.	No message passing and no context switching are required while the kernel is performing the job.	Micro-kernel requires message passing and context switching.

System calls :	
1.	System calls provide an interface to the services made available by an operating system.
2.	System calls handle each system call handed by the operating system.
3.	Typecast a number is associated with each system call and the system invokes the necessary system calls through function calls in the API and invokes a system call interface that invokes the actual system made available by the operating system.
4.	The system call interface interprets function calls in the API and invokes a system call interface that invokes the actual system made available by the operating system.
5.	The caller need know nothing about how the system calls is implemented or what it does during execution.
6.	The caller need know nothing about how the system calls is implemented or what it does during execution.
7.	The system, most of the details of the operating system interface are hidden from the programmer by the API and the routine support library.
8.	Five system calls used in process management.
9.	What do you understand by system call ? Enumerate OR
10.	What do you understand by system call ? List and explain three system calls used for process management.
11.	What do you understand by system call ? List and explain three system calls used for process management.
12.	One approach is for the program to ask the user for the names of the two files. In an interactive system, this approach will require a sequence of system calls, first to write the two file names are displayed, the program must open the input file and create the output file. Each of these operations requires another system call. Once the key board message on the screen and then to read from the key board the characters that define the two files.
13.	One approach is for the program to read data from one file and copy them to another file. The first input that the program will need is the names of the two files: The first input that the program will need is the output file.
14.	An operating system makes system calls available, let's first use an example to illustrate how system calls are used: writing a simple program to read data from one file and copy them to another file.
15.	Language instructions.

Answer	
1.	AKTU 2011-12, Marks 05
2.	AKTU 2011-12, Marks 05
3.	AKTU 2011-12, Marks 05
4.	AKTU 2011-12, Marks 05
5.	AKTU 2011-12, Marks 05
6.	AKTU 2011-12, Marks 05

to file of this name or that the file is protected against access.

5. Once the two file names are obtained, the program must open the input file and create the output file. Each of these operations requires another system call.

6. When the program tries to open the file, it may find that there is no file of that name or that the file is protected against access.

7. The first input that the program will need is the names of the two files: The first input that the program will need is the output file.

8. The first input that the program will need is the names of the two files: The first input that the program will need is the output file.

9. An operating system makes system calls available, let's first use an example to illustrate how system calls are used: writing a simple program to read data from one file and copy them to another file.

10. Language instructions.

11. All the system calls are generally available as routines written in C and C++, although certain low-level tasks (for example, task where hardware must be accessed directly), may need to be written using assembly language instructions.

12. All the system calls provide an interface to the services made available by an operating system.

13. System calls provide an interface to the services made available by an operating system.

14. An operating system makes system calls available, let's first use an example to illustrate how system calls are used: writing a simple program to read data from one file and copy them to another file.

15. An operating system makes system calls available, let's first use an example to illustrate how system calls are used: writing a simple program to read data from one file and copy them to another file.

16. One approach is for the program to read data from one file and copy them to another file.

17. One approach is for the program to read data from one file and copy them to another file.

18. One approach is for the program to read data from one file and copy them to another file.

19. One approach is for the program to read data from one file and copy them to another file.

20. One approach is for the program to read data from one file and copy them to another file.

21. One approach is for the program to read data from one file and copy them to another file.

22. One approach is for the program to read data from one file and copy them to another file.

23. One approach is for the program to read data from one file and copy them to another file.

24. One approach is for the program to read data from one file and copy them to another file.

25. One approach is for the program to read data from one file and copy them to another file.

26. One approach is for the program to read data from one file and copy them to another file.

27. One approach is for the program to read data from one file and copy them to another file.

28. One approach is for the program to read data from one file and copy them to another file.

29. One approach is for the program to read data from one file and copy them to another file.

30. One approach is for the program to read data from one file and copy them to another file.

31. One approach is for the program to read data from one file and copy them to another file.

32. One approach is for the program to read data from one file and copy them to another file.

33. One approach is for the program to read data from one file and copy them to another file.

34. One approach is for the program to read data from one file and copy them to another file.

S. No.	Process switch	Mode switch
1.	Process switching occurs when one process switches from old process to another.	Mode switching is either of these two modes.
2.	When process saves the context of the kernel mode to memory.	Mode switching can be done using system call. This is a special instruction that sets the system state to kernel mode.
3.	Example : If an interrupt occurs, the system needs to save the context of running process on CPU, so necessary to restore the saved context when interrupt is over, i.e., process switching is useful for resuming of a process.	Example : If a user process needs to access things controlled by kernel, it is necessary to perform mode switching.

Answer

AKTU 2012-13, Marks 2.5

Ques 1.21. Differentiate between process switch and mode switch.

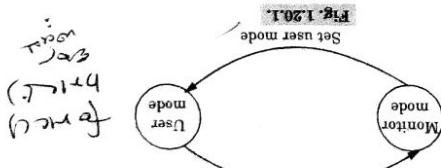


Fig. 1.20.1

- c. The privileged instructions are executed only in the monitor mode.
- d. These provide greater protection for the operating system.
- e. These instructions in other mode, i.e., user mode.
- f. Fig. 1.20.1 shows the dual mode protection.

1-20 B (CS/IT-Sem-4)

Computer System

- a. At the booting time, the hardware starts in the monitor mode. This feature helps to initialize system in many ways.
- b. When monitor mode of the system, it is in monitor mode. When the operating system is loaded, the hardware switches from user mode to monitor mode when interrupts occur. Within the operating system to the operating system from unauthorized users.
- c. Within the bit, we are able to distinguish between a task that is switched in user mode or monitor mode.
- d. In dual mode operation, two separate modes are used for working of processor mode. These modes are user mode and monitor mode.
- e. The monitor mode helps to initialize system in many ways.
- f. This feature helps to do many tasks that are used in the computer hardware. The mode bit is 0 for monitor and 1 for user.
- g. Within the mode of the system, mode bit is used in the computer for distinguishing mode of the system.
- h. The dual mode operation provides the protection to the operating system.

AKTU 2012-13, Marks 05

OR

AKTU 2012-13, Marks 05

What is the reason behind dual mode operation of computer?

AKTU 2012-13, Marks 05

What are the dual modes of operation in the system? Give arguments in favour of your answer.

AKTU 2012-13, Marks 05

What is a secure operating system without having dual mode of operation in the system? Give arguments in favour of your answer.

AKTU 2012-13, Marks 05

What is the dual mode of operation in the system? Give arguments in favour of your answer.

AKTU 2012-13, Marks 05

What is the reason behind dual mode operation of computer?

AKTU 2012-13, Marks 05

What are the dual modes of operation in the system? Give arguments in favour of your answer.

AKTU 2012-13, Marks 05

What is a secure operating system without having dual mode of operation in the system? Give arguments in favour of your answer.

AKTU 2012-13, Marks 05

What is the dual mode of operation in the system? Give arguments in favour of your answer.

AKTU 2012-13, Marks 05

What is the reason behind dual mode operation of computer?

AKTU 2012-13, Marks 05

What are the dual modes of operation in the system? Give arguments in favour of your answer.

AKTU 2012-13, Marks 05

What is a secure operating system without having dual mode of operation in the system? Give arguments in favour of your answer.

AKTU 2012-13, Marks 05

What is the dual mode of operation in the system? Give arguments in favour of your answer.

AKTU 2012-13, Marks 05

What is the reason behind dual mode operation of computer?

AKTU 2012-13, Marks 05

What are the dual modes of operation in the system? Give arguments in favour of your answer.

AKTU 2012-13, Marks 05

What is a secure operating system without having dual mode of operation in the system? Give arguments in favour of your answer.

AKTU 2012-13, Marks 05

What is the dual mode of operation in the system? Give arguments in favour of your answer.

AKTU 2012-13, Marks 05

What is the reason behind dual mode operation of computer?

AKTU 2012-13, Marks 05

What are the dual modes of operation in the system? Give arguments in favour of your answer.

AKTU 2012-13, Marks 05

What is a secure operating system without having dual mode of operation in the system? Give arguments in favour of your answer.

AKTU 2012-13, Marks 05

What is the dual mode of operation in the system? Give arguments in favour of your answer.

AKTU 2012-13, Marks 05

What is the reason behind dual mode operation of computer?

AKTU 2012-13, Marks 05

What are the dual modes of operation in the system? Give arguments in favour of your answer.

AKTU 2012-13, Marks 05

What is a secure operating system without having dual mode of operation in the system? Give arguments in favour of your answer.

AKTU 2012-13, Marks 05

## Concurrent Processes



Part-1 ..... (2-2B to 2-11B)

- Concurrent Processes: Process Concept
- Principle of Concurrency
- Producer Consumer Problem
- Mutual Exclusion Problem
- Critical Section Problem
- Deadlock Problem
- Person Solution

Part-2 ..... (2-2B to 2-26B)

A. Concept Outline: Part-1 ..... 2-2B

B. Long and Medium Answer Type Questions ..... 2-2B

A. Concept Outline: Part-2 ..... 2-11B

B. Long and Medium Answer Type Questions ..... 2-11B

- Semaphores
- Test and Set Operations
- Classical Problem in Concurrency
- Dining Philosopher Problem
- Spinning Barber Problem
- Interprocess Communication Models and Schemes
- Process Generation

2-1-B (CS/IT-Sem-4)

VERY IMPORTANT QUESTIONS	
Following questions are very important. These questions may be asked in your SESSIONALS as well as UNIVERSITY EXAMINATIONS.	Operating Systems

Q. 1. Write down the different types of an operating system and also write functions of an operating system.

Ans Refer Q. 1.2 and Q. 1.1.

Q. 2. Differentiate between the following:

Ans i. Hard real time and soft real time system.

ii. Multi-processing and multi-programming.

Ans iii. Refer Q. 1.3.

Ans iv. Refer Q. 1.7.

Ans v. Refer Q. 1.11.

Q. 3. Explain layered structure of an operating system.

Ans Ans Refer Q. 1.13.

Q. 4. Define the services provided by the operating system.

Ans Ans Refer Q. 1.13.

Q66

Q. 5. Differentiate between the following:

Ans i. Shell and kernel

Ans ii. Monolithic kernel and micro-kernel

Ans iii. Refer Q. 1.16.

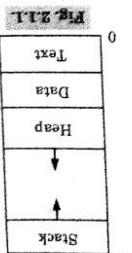
Q. 6. How is a system call handled by the system? Enumerate

Ans Ans Refer Q. 1.17 and Q. 1.18.

the five system calls used in process management?

Scanned by CamScanner

S.No.	Process	Program
i.	Process and program	i. Busy wait and blocking wait
ii.	Process and program	ii. ARTU 2012-13, Marks 06
iii.	Differentiate between the following :	iii. Question
iv.	Differentiate between the following :	iv. ARTU 2012-13, Marks 06



- Ques 2.2. The structure of a process in memory is shown in Fig. 2.11.  
2-A-B (CST-Sem-4)  
Operating Systems  
PART-I  
Concurrent Processes : Process Concept, Principle of Concurrency, Thread Concurrency Problem, Mutual Exclusion, Critical Section Problem, Deadlock Problem, Person Solution.

Ques 2.2. Explain the term process. ARTU 2012-13, Marks 06

Concept Outline : PART-I	
Requirements of a solution to critical section problem :	
i.	Mutual exclusion
ii.	Requiring is a sequential program in execution.
iii.	Bound waiting
iv.	Process

Ques 2.2. Explain the term process. ARTU 2012-13, Marks 06	
What do you mean by process model or process?	
OR	
Ques 2.2. Explain the term process. ARTU 2012-13, Marks 06	
Ques 2.2. Explain the term process. ARTU 2012-13, Marks 06	

Ques 2.2. Explain the term process. ARTU 2012-13, Marks 06	
1.	A process is a sequential program in execution
2.	A process defines the manipulation of memory for the computer
3.	Process maintains the environment for the computer
4.	Process is an entity of memory which is managed by the computer
5.	Process is a sequence of instructions that are executed
6.	Process maintains the environment for the computer
7.	A process may include a heap, which is memory that is dynamically allocated during process runtime.

### 2-5 B (CS/IT-Sem-4)

#### Operating Systems

#### Ques 2.3. Busy wait and blocking wait

**Answer**

- Producer process produce data item that consumer 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 in a buffer and the consumer reads data from the buffer. Consumer cannot retrieve any data if the buffer is full. Similarly, a producer cannot deposit its data if the buffer is empty.
- If the buffer is not full, a producer can deposit its data in the buffer. The consumer should be allowed to retrieve a data item if buffer contains.
- Concurrence is based on following principles:

#### Solution to producer consumer problem :

```

void main()
{
    int i;
    int j;
    int binarySemaphore = 1;
    int count = 0;
    semWait();
    while (true)
    {
        if (count == 1)
            append();
        else
            produce_data_item();
        count = count + 1;
    }
}

```

#### Solution to producer consumer problem :

```

void main()
{
    int i;
    int j;
    int binarySemaphore = 1;
    int count = 0;
    semWait();
    while (true)
    {
        if (count == 1)
            append();
        else
            produce_data_item();
        count = count + 1;
    }
}

```

#### Ques 2.4. Explain the principle of concurrence.

**Answer**

- Concurrence is based on following principles:
- Relative speed of execution of processes is not predictable.
- System interrupts are not predictable.
- Schedule policies may vary.
- Concurrence is the tendency for things to happen at the same time in a system.
- It also refers to techniques that makes program more usable.
- Concurrency can be implemented and is used a lot on single processing units, it may benefit from multiple processing units with respect to speed.
- An operating system is called a multi-tasking operating system, this is a synonym for supporting concurrent.
- 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.
- We turn distributed-net computations in the background, this is concurrency.
- It turns distributed-net computations in the background, this is solution.

#### Ques 2.5. Discuss producer consumer problem.

**Answer**

- Concurrence is the tendency for things to happen at the same time in a system.
- It also refers to techniques that makes program more usable.
- Concurrence is the tendency for things to happen at the same time in a system.
- Concurrence can be implemented and is used a lot on single processing units, it may benefit from multiple processing units with respect to speed.
- An operating system is called a multi-tasking operating system, this is a synonym for supporting concurrent.
- 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.
- We turn distributed-net computations in the background, this is concurrency.
- It turns distributed-net computations in the background, this is solution.

#### Ques 2.6. Explain the principle of concurrence.

**Answer**

S.No.	Busy wait	Blocking wait	Concurrent Processes
1.	Busy wait is a loop that reads the status register over and over until the busy bit becomes clear.	Blocking wait occurs when process resources are needed for another task.	It occurs when scheduling is expected wait time.
2.	It occurs when scheduling overheads is larger than overheads needed for another task.	In this, schedule based algorithm is inappropriate.	It is schedule based.
3.	Producer processes wait indefinitely within a semaphore.	Producer processes wait indefinitely within a semaphore.	System interrupt

## 2-6-B (CS/IT-Sem-4) Concurrent Processes

### Operating Systems

`semWait(B);  
consume();  
p = count;  
count = count - 1;`

1. Mutual exclusion must be enforced by allowing only one process at a time into its critical section. Among all processes that have critical sections for the same resource or shared object.

`A interfering with other processes.  
It must not be possible for a process requiring access to a critical section that has been delayed indefinitely, no deadlock or starvation.`

3. It must not be possible for a process to permit entry to its critical section unless it is in the same section itself.

`What do you understand by critical section problem?  
OR  
Ques 2.7. What are the requirements of a solution to the critical section problem?`

1. Mutual exclusion is a way to make sure that if one process is using a shared modifiable data, the other processes will be excluded from doing the same thing.

`Formally, while one process executes the shared variable, all other processes desiring to do so at the same time should be kept waiting when that process has finished executing the shared variable, one of them needs to be informed only when processes share a shared memory.`

2. Mutual exclusion needs to be enforced only so that processes can proceed with one another, they should be allowed to proceed concurrently.

`Critical section :  
1. A critical section is a code segment in a process in which a shared resource is accessed.  
2. Each process has a segment of code called a critical section. Critical section can execute in critical section.  
3. In a critical section the process may be changing common variables.  
4. When one process is executing in its critical section, no other process is to be allowed to execute in its critical section. The execution of critical section by the process is mutually exclusive in time.`

`A solution to the critical section problem must satisfy the following three requirements:  
1. Mutual exclusion: If process P is executing in its critical section, then no other processes can be executing in the critical section.  
2. Progress: If no process is executing in its critical section next, then some processes wish to enter their critical sections, then only those processes that are not executing in their critical sections can participate in deciding which will enter its critical section next, and this selection cannot be postponed indefinitely.`

4. When one process is executing in its critical section, no other process is updated in a file and so on. At any moment at most one process can update a table, writing a file and so on.

`4. When one process is executing in its critical section, the other processes are made about relative speeds of processes or sections.`

`2. Progress: If no process is executing in its critical section next, then some processes that are not executing in their critical sections, then only those processes can participate in deciding which will enter its critical section next, and this selection cannot be postponed indefinitely.`

`3. No process outside its critical section should block other processes.`

`4. No process should wait arbitrarily long to enter its critical section.`

Mutual exclusion should meet the following requirements:

Answer

Ques 2.6 Discusses the requirements of mutual exclusion.

`a. No two processes may at the same moment be inside their critical sections.`

`b. No assumptions are made about relative speeds of processes or sections.`

`c. No process should wait arbitrarily long to enter its critical section.`

`d. No process should wait arbitrarily long to enter its critical section.`

Four conditions of mutual exclusion are:

Answer

`1. Mutual exclusion needs to be enforced only when processes share a shared resource in order to ensure the system integrity.`

`2. Mutual exclusion needs to be enforced only when processes share a shared resource in order to proceed concurrently.`

`3. Mutual exclusion needs to be enforced only when processes share a shared resource in order to keep waiting, one of them needs to be informed only so that processes can proceed with one another, they should be allowed to proceed.`

`4. Mutual exclusion needs to be enforced only when processes share a shared resource in order to prevent modifying data when processes access shared memory.`

`5. Mutual exclusion needs to be enforced only when processes share a shared resource in order to prevent deadlock or starvation.`

`6. Mutual exclusion needs to be enforced only when processes share a shared resource in order to prevent deadlock or starvation.`

`Ques 2.7. What are the four conditions of mutual exclusion?`

of mutual exclusion?

Answer

`1. Mutual exclusion is a way to make sure that if one process is using a shared modifiable data, the other processes will be excluded from doing the same thing.`

`2. Formaly, while one process executes the shared variable, all other processes desiring to do so at the same time should be kept waiting, when that process has finished executing the shared variable, one of them needs to do so so that processes can proceed with one another, they should be allowed to proceed.`

`3. Mutual exclusion needs to be enforced only when processes share a shared resource in order to prevent deadlock or starvation.`

`4. Mutual exclusion needs to be enforced only when processes share a shared resource in order to prevent deadlock or starvation.`

`5. Mutual exclusion needs to be enforced only when processes share a shared resource in order to prevent deadlock or starvation.`

`6. Mutual exclusion needs to be enforced only when processes share a shared resource in order to prevent deadlock or starvation.`

`Ques 2.7. What are the requirements of a solution to the critical section problem?`

`OR`

`What are the requirements of a solution to the critical section problem?`

`1. Mutual exclusion is a way to make sure that if one process is using a shared modifiable data, the other processes will be excluded from doing the same thing.`

`2. Formally, while one process executes the shared variable, all other processes desiring to do so at the same time should be kept waiting, when that process has finished executing the shared variable, one of them needs to do so so that processes can proceed with one another, they should be allowed to proceed.`

`3. Mutual exclusion needs to be enforced only when processes share a shared resource in order to prevent deadlock or starvation.`

`4. Mutual exclusion needs to be enforced only when processes share a shared resource in order to prevent deadlock or starvation.`

`5. Mutual exclusion needs to be enforced only when processes share a shared resource in order to prevent deadlock or starvation.`

`6. Mutual exclusion needs to be enforced only when processes share a shared resource in order to prevent deadlock or starvation.`

`Ques 2.7. What are the requirements of a solution to the critical section problem?`

`OR`

`What are the requirements of a solution to the critical section problem?`

`1. Mutual exclusion must be enforced by allowing only one process at a`

`time into its critical section. Among all processes that have critical sections for the same resource or shared object.`

`2. A process that has in its non-critical section must do so without interfering with other processes.`

`3. It must not be possible for a process requiring access to a critical section that has been delayed indefinitely, no deadlock or starvation.`

`4. When no process is in a critical section, any process that requests entry to its critical section must be permitted to enter without delay.`

`5. No assumption are made about relative speeds of processes entry to its critical section must be permitted to enter without delay.`

`6. When no process is in a critical section, any process requesting access to a critical section that has been delayed indefinitely, no deadlock or starvation.`

`Ques 2.7 B (CS/IT-Sem-4)`

- 2-8 B (CS/IT-Sem-4) **Concurrent Processes**  
**Ques 28.** What do you understand by critical section ? Discuss Bækry algorithm. Also, show how it satisfies the requirements of a mechanism to control access to critical section.

- Answer**  
**Critical section :** It solves the problem of multiple processes solution.  
 1. Bækry algorithm is used in multiple process solution.  
 2. Bækry algorithm is for two processes based solely on software.  
 3. Each of these processes loop indefinitely, repeatedly entering and reentering its critical section.  
 4. A process ( $P_i$  and  $P_j$ ) that wishes to execute its critical section first enters the flag and proceeds to critical section.  
 5. Leaves the flag and executes the blackboard.  
 6. Leaves this executive until it is allowed to enter its critical section. This process repeats this executive until it is allowed to enter its critical section. This  
 7. Procedure is known as busy waiting.  
 8. In this case, process with the lowest name is served first.

- Bækry algorithm :**  
 1. Busy waiting occurs when a process is in its critical section, any other process that tries to enter its critical section must loop continuously in the entry code.  
 2. This kind of problem occurs in multiprogramming system.

- do**  
 1. Busy waiting :  
 2. Busy waiting wastes CPU cycles that some other process might be able to use productively.  
 3. Busy waiting wastes CPU cycles that some other process might be able to use productively.  
 4. Deekker's solution :  
 5. Otherwise that process will wait for its turn.  
 6. Process reenters in the flag to check the blackboard.  
 7. It repeats this executive until it is allowed to enter its critical section. This procedure is known as busy waiting.  
 8. If  $P_i$  and  $P_j$ , receive the same number and if  $i < j$ , then  $P_i$  is served first.

- Ques 210.** Give the principle, mutual exclusion in critical section problem. Also discuss how well these principles are followed in Dekker's solution.  
**AKTU 2016-17, Marks 10**

- Ques 210.** Give the principle, mutual exclusion in critical section problem. Also discuss how well these principles are followed in Dekker's solution.  
**AKTU 2016-17, Marks 10**

- Answer**  
**//critical section**  
**//wait(mutex);**  
**//signal(mutex);**  
**//remainder section**  
**do**  
 1. Busy waiting occurs when a process is in its critical section, any other process that tries to enter its critical section must loop continuously in the entry code.  
 2. This kind of problem occurs in multiprogramming system.

- do**  
 1. Busy waiting :  
 2. Busy waiting wastes CPU cycles that some other process might be able to use productively.  
 3. Busy waiting wastes CPU cycles that some other process might be able to use productively.  
 4. Deekker's solution :  
 5. Otherwise that process will wait for its turn.  
 6. Process reenters in the flag to check the blackboard.  
 7. It repeats this executive until it is allowed to enter its critical section. This procedure is known as busy waiting.  
 8. If  $P_i$  and  $P_j$ , receive the same number and if  $i < j$ , then  $P_i$  is served first.

- Ques 210.** Give the principle, mutual exclusion in critical section problem. Also discuss how well these principles are followed in Dekker's solution.  
**AKTU 2016-17, Marks 10**

- Ques 210.** Give the principle, mutual exclusion in critical section problem. Also discuss how well these principles are followed in Dekker's solution.  
**AKTU 2016-17, Marks 10**

- Answer**  
**Critical section :** Refer Q. 2.7, Page 2-7B, Unit-2.  
**Deekker's algorithm :**  
 1. Bækry algorithm is used in multiple process solution.  
 2. It solves the problem of critical section for n processes.  
 3. Each process requesting entry to critical section is given a number token such that the number on the token is larger than the maximum number issued earlier.  
 4. This algorithm was developed for a distributed environment.  
 5. The algorithm permits processes to enter the critical section in the order of their taken numbers.  
 6. The Bækry algorithm cannot guarantee that two processes do not receive the same number.

- Ques 210.** Give the principle, mutual exclusion in critical section problem. Also discuss how well these principles are followed in Dekker's solution.  
**AKTU 2016-17, Marks 10**

- Ques 210.** Give the principle, mutual exclusion in critical section problem. Also discuss how well these principles are followed in Dekker's solution.  
**AKTU 2016-17, Marks 10**

- Answer**  
**Critical section :** Refer Q. 2.7, Page 2-7B, Unit-2.  
**Deekker's algorithm :**  
 1. Bækry algorithm is used in multiple process solution.  
 2. It solves the problem of critical section for n processes.  
 3. Each process requesting entry to critical section is given a number token such that the number on the token is larger than the maximum number issued earlier.  
 4. This algorithm was developed for a distributed environment.  
 5. The algorithm permits processes to enter the critical section in the order of their taken numbers.  
 6. The Bækry algorithm cannot guarantee that two processes do not receive the same number.

- Answer**  
**Critical section :** Refer Q. 2.7, Page 2-7B, Unit-2.  
**Deekker's algorithm :**  
 1. Bækry algorithm is used in multiple process solution.  
 2. It solves the problem of critical section for n processes.  
 3. Each process requesting entry to critical section is given a number token such that the number on the token is larger than the maximum number issued earlier.  
 4. This algorithm was developed for a distributed environment.  
 5. The algorithm permits processes to enter the critical section in the order of their taken numbers.  
 6. The Bækry algorithm cannot guarantee that two processes do not receive the same number.

- Answer**  
**Critical section :** Refer Q. 2.7, Page 2-7B, Unit-2.  
**Deekker's algorithm :**  
 1. Bækry algorithm is used in multiple process solution.  
 2. It solves the problem of critical section for n processes.  
 3. Each process requesting entry to critical section is given a number token such that the number on the token is larger than the maximum number issued earlier.  
 4. This algorithm was developed for a distributed environment.  
 5. The algorithm permits processes to enter the critical section in the order of their taken numbers.  
 6. The Bækry algorithm cannot guarantee that two processes do not receive the same number.

- Answer**  
**Critical section :** Refer Q. 2.7, Page 2-7B, Unit-2.  
**Deekker's algorithm :**  
 1. Bækry algorithm is used in multiple process solution.  
 2. It solves the problem of critical section for n processes.  
 3. Each process requesting entry to critical section is given a number token such that the number on the token is larger than the maximum number issued earlier.  
 4. This algorithm was developed for a distributed environment.  
 5. The algorithm permits processes to enter the critical section in the order of their taken numbers.  
 6. The Bækry algorithm cannot guarantee that two processes do not receive the same number.

- Answer**  
**Critical section :** Refer Q. 2.7, Page 2-7B, Unit-2.  
**Deekker's algorithm :**  
 1. Bækry algorithm is used in multiple process solution.  
 2. It solves the problem of critical section for n processes.  
 3. Each process requesting entry to critical section is given a number token such that the number on the token is larger than the maximum number issued earlier.  
 4. This algorithm was developed for a distributed environment.  
 5. The algorithm permits processes to enter the critical section in the order of their taken numbers.  
 6. The Bækry algorithm cannot guarantee that two processes do not receive the same number.



2-20 B (CSIT-Sem-4)

2-21 B (CSIT-Sem-4)

**Q. 11** State the readers/writers problem with readers having priority. Give solution of the problem using semaphores.**Answer** Readers/writers problem : Refer Q. 2.18, Page 2-17B, Unit-2.**Q. 11** Readers/writers problem with readers having priority and its solution :

- Here priority means, no reader should wait if the share is currently opened for reading. Three variables are used : mutex, wrt, readcnt to implement solution
- semaphore mutex, wrt; // semaphore mutex is used to ensure mutual exclusion when readent is updated i.e. when any reader enters or exit from the critical section and semaphore wrt is used by both readers and writers
- int readent; // readent tells the number of processes performing read in the critical section. initially 0

**Functions for semaphore :**

- wait() decrements the semaphore value.
- signal() increments the semaphore value.

**Writer process :**

- Writer requests the entry to critical section.
- If allowed i.e. wait() gives a true value, it enters and performs the write.
- If not allowed, it keeps on waiting.
- It exits the critical section.

**Reader processes :**

- Reader requests the entry to critical section.
  - If allowed :
    - It increments the count of number of readers inside the critical section. If this reader is the first reader entering, it locks the wrt semaphore to restrict the entry of writers if any reader is inside.
    - If then, signals mutex as any other reader is allowed to enter while others are already reading.
    - After performing reading, it exits the critical section. When exiting, it checks if no more reader is inside, it signals the semaphore 'wrt' as now, writer can enter the critical section.
  - If not allowed, it keeps on waiting.
- Thus, the mutex 'wrt' is queued on both readers and writers in a manner such that preference is given to readers if writers are also there. Thus, no reader is waiting simply because a writer has requested to enter the critical section.

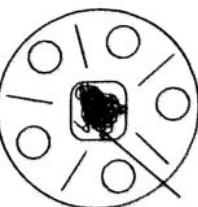
**Q. 12** Describe classical problem of concurrency.**Answer****Classical problem of concurrency are :**

- Bounded buffer producer consumer problem : Refer Q. 2.17, Page 2-16B, Unit-2.
- Readers/writers problem : Refer Q. 2.18, Page 2-17B, Unit-2.

**Q. 12** Dining philosophers problem :

- The Dining philosophers problems is a classic synchronization problem.
- There is a dining room containing a circular table with five chairs. At each chair is a plate, and between each plate is a single chopstick.
- In the middle of the table is a bowl of spaghetti.

Spaghetti

**Fig. 2.22.1.**

- Near the room are five philosophers who spend most of their time thinking, but who occasionally get hungry and need to eat so they can think some more.
- In order to eat, a philosopher must sit at the table, pick up the two chopsticks to the left and right of a plate, then serve and eat the spaghetti on the plate.
- Thus, each philosopher is represented by the following pseudocode :

```

process Pi]
  while true do
    THINK;
    PICKUP(CHOPSTICKi, CHOPSTICKi+1 mod 5);
    EAT;
    PUTDOWN(CHOPSTICKi, CHOPSTICKi+1 mod 5);
}

```

- A philosopher may THINK indefinitely. Every philosopher who EATs will eventually finish.
- Philosophers may PICKUP and PUTDOWN their chopsticks in either order, or nondeterministically, but these are atomic actions, and, of course, two philosophers cannot use a single CHOPSTICK at the same time.
- The problem is to design a protocol to satisfy the liveness condition: any philosopher who tries to EAT, eventually does.

**Operating Systems****2-23 B (CSIT-Sem-4)**

- 4. Sleeping barber problem :**  
A barbershop is designed so that there is a private room that contains the barber chair and an adjoining waiting room with a sliding door that contains  $N$  chairs.

1. If the barber is busy, the door to the private room is closed and arriving customers sit in one of the available chairs.
2. If a customer enters the shop and all chairs are occupied, the customer leaves without a haircut. If there are no customers to be served, the barber goes to sleep in the barber chair with the door to the waiting room open.
3. If the barber is asleep, the customer wakes the barber and has a haircut. Fig. 2.22.2 shows an implementation that uses semaphores.

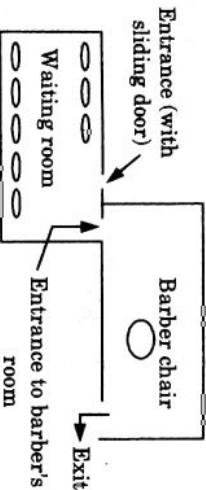


Fig. 2.22.2. Barbershop.

**Que 2.23. Write a short note on interprocess communication.**

**AKTU 2011-12, Marks 10**

**OR**

**Define message passing and shared memory interprocess communication.**

**AKTU 2013-14, Marks 05**

**Answer**

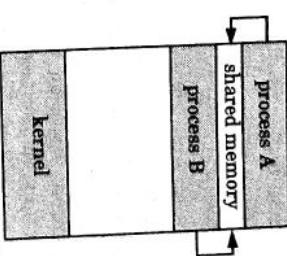
1. Interprocess communication (IPC) means the mechanism used by processes to communicate with each other while they are running.
2. IPC allows processes to synchronize their action without sharing the same address space.
3. Messages provide a basic communication capability between two processes.
4. IPC is particularly useful in a distributed environment where the communicating processes may reside on different computers connected with a network.
5. Interprocess communication is best provided by the two fundamental models :
  - a. **Shared memory systems :**
    - i. Interprocess communication using shared memory requires communicating processes to establish a region of shared memory.

Fig. 2.23.1. Shared memory.

**b. Message passing systems :**

- i. Message passing provides a mechanism to allow processes to communicate and to synchronize their actions without sharing the same address space.
- ii. It is particularly useful in a distributed environment, where the communicating processes may reside on different computers connected by a network.
- iii. A message passing facility provides at least two operations : send (message) and receive (message).
- iv. Messages sent by a process can be either fixed or variable in size.
- v. If processes  $P$  and  $Q$  want to communicate, they must send messages to and receive messages from each other : a communication link must exist between them.
- vi. Here are several methods for logically implementing a link and the send () or receive () operations :
  1. Direct or indirect communication
  2. Synchronous or asynchronous communication
  3. Automatic or explicit buffering

- ii. Typically, a shared memory region resides in the address space of the process creating the shared memory segment.
- iii. Other processes that wish to communicate using this shared memory segment must attach it to their address space.
- iv. The operating system tries to prevent one process from accessing another process's memory.
- v. Shared memory requires that two or more processes agree to remove this restriction.
- vi. They can then exchange information by reading and writing data in the shared areas.
- vii. The form of the data and the location are determined by these processes and are not under the operating system's control.
- viii. The processes are also responsible for ensuring that they are not writing to the same location simultaneously.



## Operating Systems

## 2-25 B (CSIT-Seam-4)

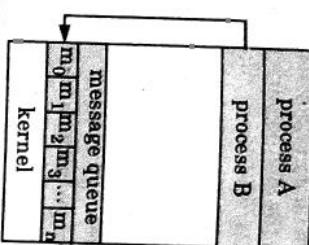


Fig. 2.23.2. Message passing.

**Que 2.24.** Discuss message passing systems. Explain how message passing can be used to solve buffer producer consumer problem with infinite buffer.

**AKTU 2012-13, Marks 10**

**Answer**

**Message passing system :** Refer Q. 2.23, Page 2-22B, Unit-2.

**Solution of producer consumer problem with infinite buffer:**

- Let us first consider the case in which the buffer is infinite.
- Fig. 2.24.1 shows two pointers, in and out, which are used respectively indicating the next space for producers to put a newly created product and the next product that is available for consumers to consume.

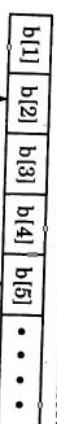


Fig. 2.24.1.

Out

In

- Instead of two pointers, in and out, an integer variable, n is used to indicate the number of products available for consumption and a set of functions are provided for convenience: produce(), consume(), append(), and take().

- The semaphore s is used to enforce the mutual exclusion; the semaphore delay is used to force the consumer to wait if the buffer is empty.
- ```

void producer(void)
{
    while (TRUE) {
        void consumer(void)
        {
            while (TRUE) {
                recv (producer, item);
                produce item
                send (consumer, item);
            }
        }
    }
}

```

**Answer**

**AKTU 2014-15, Marks 10**

We use 3 semaphores. Semaphore 'customers' counts waiting customers; semaphore 'barbers' is the number of idle barbers (0 or 1); and mutex is used for 'mutual' exclusion.

- A shared data variable 'customers' also counts waiting customers. It is a copy of 'customers' but we need it here because we cannot access the value of semaphores directly.
- We also need semaphore 'cutting' which ensures that the barber would not cut another customer's hair before the previous customer leaves.
- shared data  
semaphore customers = 0;  
semaphore barbers = 0;  
semaphore cutting = 0;
- semaphore mutex = 1;  
int customers1 = 0;  
void barber() {  
 while(true) {  
 wait(customers);  
 // sleep when there are no waiting  
 customers  
 wait(mutex);  
 // mutex for accessing customers1  
 customers1 = customers1 - 1;  
 signal(mutex);  
 signal(barbers);  
 cut\_hair();  
 }  
}
- void customer() {  
 wait(mutex); // mutex for accessing customers1  
 if (customers1 < n) {  
 customers1 = customers1 + 1;  
 signal(customers);  
 signal(mutex);  
 wait(barbers);  
 }  
}

**Que 2.25.** A barber shop consists of a waiting room with  $n$  chairs and a barber room with one barber chair. If there are no customers to be served, the barber goes to sleep. If a customer enters the barber shop and all chairs are occupied, then the customer leaves the shop. If the barber is busy but chairs are available, then the customer sits in one of the free chairs. If the barber is asleep, the customer wakes up the barber. Write an algorithm for the above synchronization problem using semaphores.

```
//wait for available barbers
get_haircut();
else ( // do nothing (leave) when all chairs
    signal(mutex);
)

```

are used.

```
) cut_hair()
waiting(cutting);
```

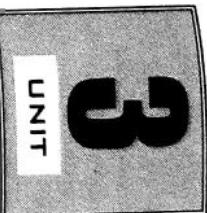
```
) get_haircut()
get hair cut for some time;
signal(cutting);
```

### VERY IMPORTANT QUESTIONS

*Following questions are very important. These questions may be asked in your SESSIONALS as well as UNIVERSITY EXAMINATION.*

- Q. 1. Discuss process model and principle of concurrency.  
Refer Q. 2.1 and Q. 2.3.
- Q. 2. Write short note on producer consumer problem along with its solution using semaphore.  
Refer Q. 2.4 and Q. 2.16.
- Q. 3. What is mutual exclusion ? What are the four conditions of mutual exclusion ?  
Refer Q. 2.5.
- Q. 4. Write and explain Peterson solution to the critical section problem.  
Refer Q. 2.11.
- Q. 5. Give a solution to reader/writer problem using semaphores.  
Refer Q. 2.19.
- Q. 6. Discuss classical problem of concurrency.  
Refer Q. 2.22.
- Q. 7. Write short note on interprocess communication.  
Refer Q. 2.23.

©©©



## CPU Scheduling

A. Concept Outline : Part-1 ..... (3-2B to 3-12B)  
B. Long and Medium Answer Type Questions ..... 3-2B

Part-2 ..... (3-12B to 3-44B)

- CPU Scheduling : Scheduling Concepts
- Performance Criteria
- Process States
- Process Transition Diagram
- Schedulers
- Process Control Block (PCB)
- Process Address Space
- Process Identification Information

- Threads and their Management
  - Scheduling Algorithms
  - Multiprocessor Scheduling
  - Deadlock : System Model
  - Deadlock Characterization
  - Prevention
  - Avoidance and Detection
  - Recovery from Deadlock
- A. Concept Outline : Part-2 ..... 3-12B  
B. Long and Medium Answer-Type Questions ..... 3-13B

**PART-1**

**CPU Scheduling :** Scheduling Concepts, Performance Criteria,

**Process States, Process Transition Diagram, Scheduling,**

**Process Control Block (PCB), Process Address Space,**

**Process Identification Information.**

**CONCEPT OUTLINE : PART-1**

- CPU scheduling is the management of CPU resources.
- **Performance scheduling criteria:**
  - i. CPU utilization
  - ii. Throughput
  - iii. Waiting time
  - iv. Turnaround time
  - v. Response time
- **Various states of process :**
  - i. New
  - ii. Running
  - iii. Waiting
  - iv. Ready
  - v. Terminated
- **Types of scheduler:**
  - i. Long term scheduler
  - ii. Short term scheduler
  - iii. Mid term scheduler

**Questions-Answers****Long Answer Type and Medium Answer Type Questions**

**Que 3.1.** What is CPU scheduling ? Give the criteria for scheduling.

Discuss the performance criteria for CPU scheduling.

OR

**AKTU 2015-16, Marks 10**

**Answer**

1. CPU scheduling is the management of CPU resources.
  2. The scheduling mechanism is the part of the process manager that handles the removal of the running process from the CPU and the selection of another process on the basis of a particular strategy.
  3. CPU scheduling is the basis of multi-programmed operating systems.
  4. The scheduler is responsible for multiplexing processes on the CPU.
  5. CPU scheduling is used to increase CPU utilization.
- Scheduling criteria :** The scheduling policy determines the importance of each of the criteria. Some commonly used criteria are :
1. **CPU utilization :**
    - a. CPU utilization is the average function of time, during which the processor is busy.
    - b. The load on the system affects the level of utilization that can be achieved.
    - c. CPU utilization may range from 0 % to 100 %. On large and expensive system i.e., time shared system, CPU utilization may be the primary consideration.
  2. **Throughput :**
    - a. Throughput refers to the amount of work completed in a unit of time.
    - b. The number of processes the system can execute in a period of time.
    - c. The higher the number, the more work is done by the system.
  3. **Waiting time :**
    - a. The average period of time a process spends waiting.
    - b. Waiting time may be expressed as turnaround time less the actual execution time.
  4. **Turnaround time :**
    - a. The interval from the time of submission of a process to the time of completion is the turnaround time.
    - b. Turnaround time is the sum of the periods spent waiting to get into memory, waiting into the ready queue, executing on the CPU and doing I/O.
  5. **Response time :**
    - a. Response time is the time from the submission of a request until the first response is produced.
  6. **Priority :**
    - a. Give preferential treatment to processes with higher priorities.

**7. Balanced utilization :**

- a. Utilization of memory, I/O devices and other system resources are also considered.
- b. Not only CPU utilization considered for performance.

**8. Fairness:**

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

**Que 3.2.** Discuss the objectives of CPU scheduling.

**Answer**

**Objectives of CPU scheduling:**

1. **Efficiency:** Keep the CPU busy all the time.
2. **Maximize throughput:** Service the largest possible number of jobs in a given amount of time; minimize the amount of time users must wait for their results.
3. **Minimize response time :** Interactive users should see good performance.
4. **Minimize overhead :** Do not waste too many resources. Keep scheduling time and context switch time at a minimum.
5. **Maximize resource use :** Favour processes that will use under-utilized resources.
6. **Avoid indefinite postponement :** Every process should get a chance to run eventually.
7. **Enforce priorities :** If the scheduler allows a process to be assigned a priority, it should be meaningful and enforced.

**Que 3.3.** What is the difference between pre-emptive and non pre-emptive scheduling ? State, why strict non pre-emptive scheduling is unlikely to be used in computer centre ? Explain the operation of multi-level scheduling.

**Answer**

Difference:

| S.No. | Pre-emptive scheduling                                                                                        | Non pre-emptive scheduling                                                                                              |
|-------|---------------------------------------------------------------------------------------------------------------|-------------------------------------------------------------------------------------------------------------------------|
| 1.    | Processor can be pre-empted to execute a different process in the middle of execution of any current process. | Once processor starts to execute a process it must finish it before executing the other. It cannot be paused in middle. |

|    |                                                                                                    |                                                                                                             |
|----|----------------------------------------------------------------------------------------------------|-------------------------------------------------------------------------------------------------------------|
| 2. | CPU utilization is more compared to non pre-emptive scheduling.                                    | CPU utilization is less compared to pre-emptive scheduling.                                                 |
| 3. | Waiting time and response time is less.                                                            | Waiting time and response time is more.                                                                     |
| 4. | If a high priority process frequently arrives in the ready queue, low priority process may starve. | If a process with long burst time is running CPU, then another process with less CPU burst time may starve. |
| 5. | Pre-emptive scheduling is flexible.                                                                | Non pre-emptive scheduling is rigid.                                                                        |
| 6. | For example, SRTF, Priority, Round-Robin, etc.                                                     | For example, FCFS, SJF, Priority, etc.                                                                      |

**Non pre-emptive scheduling is unlikely to be used in computer centre :**

1. Once the CPU has been allocated to a process, the process keeps the CPU until it releases the CPU either by terminating or by switching to the waiting state.
2. In a general purpose computer system, users share the CPU and care about system responsiveness.
3. If the system uses non pre-emptive scheduling, some users may sit before the monitor for several hours without doing anything other than waiting for the set of processes in front of them in the system queue to finish.
4. So, strictly non pre-emptive scheduling is unlikely to be used in a general purpose computer system.

**Operation of multilevel scheduling:**

1. A multilevel queue scheduling algorithm partitions the ready queue into several separate queues (Fig. 3.3.1).
2. The processes are permanently assigned to one queue, generally based on some property of the process, such as memory size, process priority, or process type.
3. Each queue has its own scheduling algorithm.
4. For example, separate queues might be used for foreground and background processes.
5. The foreground queue might be scheduled by an RR algorithm, while the background queue is scheduled by an FCFS algorithm.

highest priority  
system processes

interactive processes

interactive editing processes

batch processes

student processes

lowest priority

**Fig. 3.3.1. Multilevel queue scheduling.**

6. There must be scheduling among the queues, which is commonly implemented as fixed-priority pre-emptive scheduling.  
 7. For example, the foreground queue may have absolute priority over the background queue.

- Que 3.4.** Explain the objectives and implementation of short term scheduling and long term scheduling.

- Answer**
- Objectives and implementation of short term scheduling :**

1. The main objective is increasing system performance in accordance with the chosen set of criteria.
2. It is the change of ready state to running state of the process.
3. It is also called CPU scheduler.
4. CPU scheduler selects from among the processes that are ready to execute and allocates the CPU to one of them.

- Objectives and implementation of long term scheduling :**

1. The primary objective is to provide a balanced mix of jobs, such as I/O bound and processor bound.
2. It also controls the degree of multi-programming.

3. If the degree of multi-programming is stable, then the average rate of process creation must be equal to the average departure rate of processes leaving the system.

4. Long term scheduler determines which programs are admitted to the system for processing.

5. It is also called job scheduler,
6. Job scheduler selects processes from the queue and loads them into memory for execution. Process loads into the memory for CPU scheduler.

- Que 3.5.** List out the various states of process.

OR

- Draw the process state diagram and describe the various process states.

OR

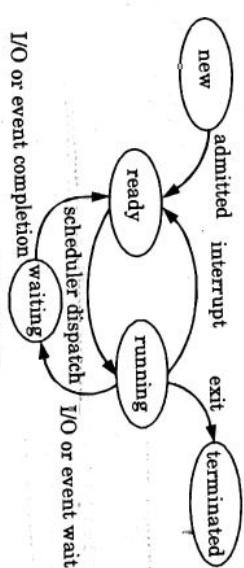
- What are the various process states? Depict process state diagram. What do you understand by context switching and various processes involved in it.

- Draw and explain the process state transition diagram.

**AKTU 2011-12, Marks 05**

- Answer**
- Various process states are :
1. New : The process is being created.
  2. Running : Instructions are being executed.
  3. Waiting : The process is waiting for some event to occur (such as an I/O completion or reception of a signal).
  4. Ready : The process is waiting to be assigned to a processor.
  5. Terminated : The process has finished execution.

- Process state diagram :**



**Fig. 3.5.1.**

- Context switching :**

1. A context switching 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.
2. Any operating system that allows for multitasking relies heavily on the use of context switching to allow different processes to run at the same time.
3. A context switch occurs when a computer's CPU switches from one process or thread to a different process or thread.

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

**Process involved in context switching are :**

The steps in a full process switch are :

1. Save the context of the processor, including program counter and other registers.
2. Update the process control block of the process that is currently in the running state. This includes changing the state of the process to one of the other states (Ready; Blocked; Ready/Suspend or Exit). Other relevant fields must also be updated, including the reason for leaving the running state and accounting information.
3. Move the process control block of this process to the appropriate queue (Ready; Blocked on Event i; Ready/Suspend).
4. Select another process for execution.
5. Update the process control block of the process selected. This includes changing the state of this process to running.
6. Update memory management data structures. This may be required, depending on how address translation is managed.
7. Restore the context of the processor to that which existed at the time the selected process was last switched out of the running state, by loading in the previous values of the program counter and other registers.

**Que 3.6.** Define the different states of a process with diagram.  
**Explain the need of process suspension.**

**AKTU 2013-14, Marks 05**

**Answer**

**States of a process :** Refer Q. 3.5, Page 3-TB, Unit-3.

**Need of process suspension are :**

The process was placed in the suspended state by itself, or OS, a parent process for the purpose of preventing its execution. There are following needs of process suspension:

1. Parent process request : A process may be suspended in order of parent process request in order to modify the process.
2. Swapping : If the process is ready and there is no place in the main memory, the process get suspended.
3. Other OS reason : When one process in main memory which was blocked and there is another process ready for execution, but waiting in secondary memory, then the process in main memory is suspended.

**Que 3.7** What is process control block ? Also list out the information it contains.

OR

**What is the process control block ? Explain all its components.**

**Answer**

**Process control block :**

1. Process control block is a data structure used to store the information about the processes.
2. The information of the process is used by CPU at the runtime.

**The following are the various information that is contained by process control block :**

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. Input / output devices associated with process

**Components of process control block :**

|   |                                   |
|---|-----------------------------------|
| 1 | Process Id                        |
| 2 | Program counter                   |
| 3 | Register information              |
| 4 | Scheduling information            |
| 5 | Memory related information        |
| 6 | Accounting information            |
| 7 | Status information related to I/O |

**Fig 3.7.1. Structure of process control block**

P V. S. M. A. S

The following are the various components that are associated with the process control block (PCB) :

1. **Process ID :**
  - a. In computer system, there are various processes running simultaneously and each process has its unique Id.
  - b. This Id helps system in scheduling the processes.
  - c. This Id is provided by the process control block.
  - d. In other words, it is an identification number that uniquely identifies the processes of computer system.

**2. Process state :**

- a. As we know that the process state of any process can be new, running, waiting, executing, blocked, suspended, terminated, process.
- b. Process control block is used to define the process state of any process.
- c. In other words, process control block refers the states of the processes.

**3. Program counter :**

- a. Program counter is used to point to the address of the next instruction to be executed in any process.
- b. This is also managed by the process control block.

**4. Register information :**

- a. It comprises of the various registers, such as index and stack that are associated with the process.
- b. This information is also managed by the process control block.

**5. Scheduling information :**

- a. Scheduling information is used to set the priority of different processes.
- b. This is very useful information which is set by the process control block.
- c. The priority of primary feature of RAM is higher than other secondary features.
- d. Scheduling information is very useful in managing any computer system.

**6. Memory related information :**

- a. This section of the process control block comprises of page and segment tables.
- b. It also stores the data contained in base and limit registers.

**7. Accounting information :**

- a. This section of process control block stores the details relate to central processing unit (CPU) utilization and execution time of a process.

**8. Status information related to input / output :**

- a. This section of process control block stores the details pertaining to resource utilization and file opened during the process execution.

**Que 3.8** Differentiate between long term, short term and mid term schedulers.

OR

With a diagram, explain the different states of a process. Differentiate between long term and short term schedulers.

**Answer**

Different states of process : Refer Q. 2.2, Page 2-3B, Unit-2.

**Difference between various schedulers:**

| S.No. | Long term                                                              | Short term                                               | Mid term                                                                    |
|-------|------------------------------------------------------------------------|----------------------------------------------------------|-----------------------------------------------------------------------------|
| 1.    | It is job scheduler.                                                   | It is CPU scheduler.                                     | It is process swapping scheduler.                                           |
| 2.    | Speed is less than short term scheduler.                               | Speed is very fast.                                      | Speed is in between both.                                                   |
| 3.    | It controls degree of multi-programming.                               | Less control over degree of multi-programming.           | Reduced the degree of multi-programming.                                    |
| 4.    | Absent or minimal in time sharing system.                              | Minimal in time sharing system.                          | Time sharing system use mid term scheduler.                                 |
| 5.    | It selects process from pool and loads them into memory for execution. | It select among the processes that are ready to execute. | Process can be reintroduced into memory and its execution can be continued. |

**Que 3.9.** Describe process address space.**Answer**

- 1. Process address space means a space that is allocated in memory for a process.

- 2. Address space is a space in computer memory.

- 3. Every process has an address space.

- 4. 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.

**Que 3.10.** Write a short note on process identification information.

**Answer**

- 1. Process registration involves recording all information necessary to identify a registered process and to differentiate it from other processes

in the system. This information is called process identification information.

- 2 This information is recorded right after the process is created or born.
- 3 Usually, every process is also given a unique identifier, called a process ID, in order to make future references easier and unambiguous.
- 4 Process identification information is usually static, which means it does not change as time goes by and as the process moves from one state of life to another, for that purpose, the system will record process state information.
- 5 Process identification information use following numeric identifiers:

- a. **Unique process identifier (PID)**: PID provide index id to the process.
- b. **User identifier (UID)**: UID identifies the user who is responsible for the job.

## PART-2

*Threads and their Management, Scheduling Algorithms, Multiprocessor Scheduling, Deadlock : System Model, Deadlock Characterization, Prevention, Avoidance and Detection, Recovery from Deadlock.*

### CONCEPT OUTLINE : PART-2

- A thread is a flow of execution through the process code, with its own program counter, system registers and stack.

- **Types of thread :**

- **Scheduling algorithm :**
  - i. User level thread
  - ii. Kernel level thread
- i. FCFS
- ii. SJF
- iii. Priority scheduling
- iv. Round Robin scheduling
- v. Multilevel queue scheduling
- vi. Multilevel feedback queue scheduling
- **Necessary conditions for deadlock :**
  - i. Mutual exclusion
  - ii. Hold and wait
  - iii. Circular wait
  - iv. No pre-emption

### Ques 3.11 How threads are implemented ?

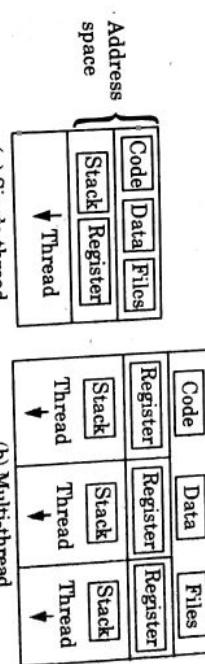
#### Questions-Answers

#### Long Answer Type and Medium Answer Type Questions

### Ques 3.11. What is thread ? Discuss its advantages.

#### Answer

1. A thread is a flow of execution through the process code, with its own program counter, system registers and stack.
2. Threads represent a software approach to improving performance of operating system by reducing the overhead of process switching.
3. Each thread belongs to exactly one process and no thread can exist outside a process. Each thread represents a separate flow of control.
4. They also provide a suitable foundation for parallel execution of applications on shared memory multi-processors.



**Fig. 3.11.1. Single and multi-thread processes.**

#### 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.
5. The benefits of multi-threading can be greatly increased in a multi-processor architecture.

To implement a threads package, there are the following three ways :

1. **Threads implementation in user space :**
  - a. Threads could be implemented at the user level.

- b. A user-level thread is known only to the user or actually the process that has created it.
- c. These tasks include but are not restricted to : CPU assignment, device assignment, state transition, etc.

- d. To be clearer, the operating system will not know that a process has created any threads and hence cannot help managing them.
- e. What the operating system does, in respect to user-level threads, is to provide a set of basic routines for creation, manipulation, and destruction of threads.
- f. These can be used by the user whenever needed, without any managerial responsibility as the part of the operating system.

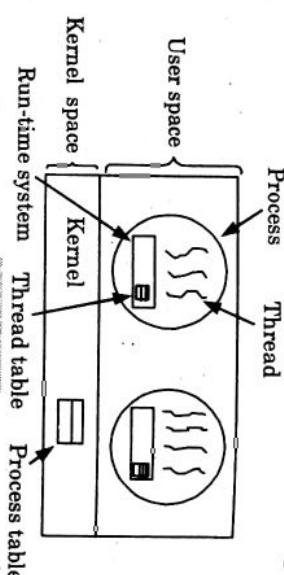


Fig. 3.12.1.

**2. Threads implementation in kernel:**

- a. In this method of implementing the threads package entirely in the kernel, no any run-time system is need in each as illustrated in the Fig. 3.12.2.

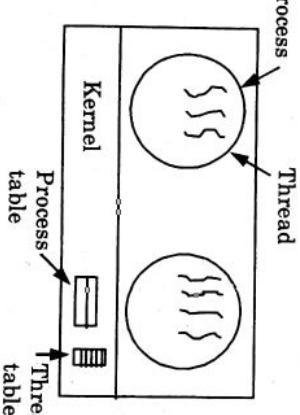


Fig. 3.12.2.

**Answer**  
Difference between user level and kernel level thread :

| S.No. | User level threads                                                  | Kernel level threads                                              |
|-------|---------------------------------------------------------------------|-------------------------------------------------------------------|
| 1.    | User level threads are <u>faster</u> to create and manage.          | Kernel level threads are slower to create and manage.             |
| 2.    | Implemented by a thread library at the user level.                  | Operating system support directly to kernel threads.              |
| 3.    | User level thread can run on any operating system.                  | Kernel level threads are <u>specific</u> to the operating system. |
| 4.    | Support provided at the user level called user level thread.        | Support may be provided by kernel is called kernel level thread.  |
| 5.    | Multi-thread application cannot take advantage of multi-processing. | Kernel routines themselves can be multi-threaded.                 |

- b. In this, there is no any thread table in each process. But to keep track of all the threads in the system, the kernel has the thread table.
- c. Whenever a thread wants to create a new thread or destroy an existing thread, then it makes a kernel call, which does the creation or destruction just by updating the kernel thread table.

- d. The thread table of the kernel holds each registers, state, and some other useful information of the thread.
- e. Here the information is the same as with the user level threads.
- f. This information is a subset of the information that traditional kernels maintain about each of their single-threaded processes, that is, the process state.
- g. In addition to these, to keep track of processes, the kernel also maintains the traditional process table.

**3. Hybrid implementation :**

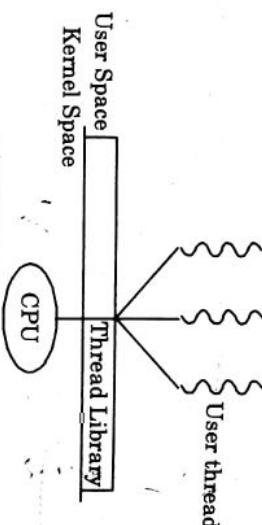
- a. The third type of thread implementation is hybrid implementation that is the implementation of a combination of user level and kernel level threads.
- b. The managerial responsibility of this kind of thread is performed partially by the user who has created the thread and partially by the operating system.

**Que 3.13.** Differentiate between user level and kernel level thread.

**AKTU 2012-13, Marks 2.5**

**Answer****Types of thread are :****1. User level thread :**

- In a user level thread, all of the work of thread management is done by the application.
- The thread library contains code for creating and destroying threads, for passing message and data between threads, in scheduling thread execution and for saving and restoring thread contexts.
- The application begins with a single thread and begins running that thread.



**Fig. 3.14.1. User level thread.**

**Advantages of user level thread :**

- Thread switching does not require kernel mode privileges.
- User level thread can run on any operating system.
- Scheduling can be application specific.
- User level threads are fast to create and manage.

**Disadvantages of user level thread :**

- In a typical operating system, most system calls are blocking.
- Multi-threaded application cannot take advantage of multi-processing.

**2. Kernel level thread :**

- In kernel level thread, thread management is done by the kernel. There is no thread management code in the application area.
- Kernel threads are supported directly by the operating system.
- All of the threads within an application are supported within a single process.
- Scheduling by the kernel is done on a thread basis.
- The kernel performs thread creation, scheduling and management in kernel space.

**Advantages of kernel level thread :**

- Kernel can simultaneously schedule multiple threads from the same process on multiple processes.
- If one thread in a process is blocked, the kernel can schedule another thread of the same process.
- Kernel routines themselves can multithread.

**Disadvantages of kernel level thread :**

- Kernel threads are generally slower to create and manage than the user threads.
- Transfer of control from one thread to another within same process requires a mode switch to the kernel.

**Que 3.15.** What is a thread ? How thread is different from a process ? What resources are used when a thread is created ?

**AKTU 2012-15, Marks 10**

| Difference between thread and process: |                                                                                                  |                                                                                                                   |
|----------------------------------------|--------------------------------------------------------------------------------------------------|-------------------------------------------------------------------------------------------------------------------|
| S.No.                                  | Thread                                                                                           | Process                                                                                                           |
| 1.                                     | Thread is called <u>light weight</u> process.                                                    | Process is called <u>heavy weight</u> process.                                                                    |
| 2.                                     | Thread switching does not need to call an operating system and cause an interrupt to the kernel. | Process switching needs interface with operating system.                                                          |
| 3.                                     | All threads can share same set of open files, child processes.                                   | In multiple process implementation each process executes the same code but has its own memory and file resources. |
| 4.                                     | While one server thread is blocked and waiting, second thread in the same task could run.        | If one server process is blocked no other server process can execute until the first process unblocked.           |
| 5.                                     | One thread can read, write or even completely wipe out another threads stack.                    | In multiple process each process operates independently of the other.                                             |

**Resources used when thread is created :** When a thread is created, the thread does not require any new resource to execute, the thread shares the resources like memory of the process to which they belong to.

**Que 3.16.** Differentiate between user thread and kernel thread  
What is thread cancellation ?

**AKTU 2013-14, Marks 05**

**Answer**

**User thread vs Kernel thread :** Refer Q. 3.13, Page 3-15B, Unit-3.

**Thread cancellation :**

1. Thread cancellation is the task of terminating a thread before it has completed.
2. For example, if multiple threads are concurrently searching through a database and one thread returns the result, the remaining threads might be canceled.
3. Another situation might occur when a user presses a button on a web browser that stops a web page from loading any further.
4. Often, a web page is loaded using several threads and each image is loaded in a separate thread.
5. When a user presses the stop button on the browser, all threads loading the page are canceled.

**Que 3.17.** Discuss First Come First Serve (FCFS) scheduling algorithm. Give its advantages and disadvantages.

**Answer**

1. The simplest scheduling algorithm is the First Come First Served (FCFS) algorithm, i.e., the process, which requests the CPU first is allocated the CPU first.
2. The FCFS algorithm is simply realized with a FIFO queue. It functions as follows :

- a. When a process enters the ready queue, its Process Control Block (PCB) is linked onto the tail of the queue.
- b. As soon as the CPU is free, it is allocated to the process at the head of the ready queue.
- c. The running process is then removed from the ready queue.

**Advantages of First Come First Served (FCFS) algorithm :**

1. Better for long processes.
2. Simple method (i.e., minimum overhead on processor).
3. No starvation.

**Disadvantages of First Come First Served (FCFS) algorithm :**

1. Convoy effect occurs. Even very small process should wait for its turn to come to utilize the CPU.

2. Short process behind long process results in lower CPU utilization.
3. Throughput is not emphasized.

**Que 3.18.** Discuss Shortest Job First (SJF) scheduling algorithm.  
Also, write its advantages and disadvantages.

**Answer**

1. SJF algorithm associates with each process the length of the process's next CPU burst.
2. When the CPU is available, it is assigned to the process that has the smallest next CPU burst.
3. If the next CPU bursts of two processes are the same, FCFS scheduling is used to break the tie.
4. The appropriate term for this scheduling method would be the shortest-next-CPU-burst algorithm, because scheduling depends on the length of the next CPU burst of a process, rather than its total length.
5. The SJF algorithm can be either preemptive or non-preemptive.
6. The choice arises when a new process arrives at the ready queue while a previous process is still executing.
7. The next CPU burst of the newly arrived process may be shorter than what is left of the currently executing process.
8. A preemptive SJF algorithm will preempt the currently executing process, whereas a non-preemptive SJF algorithm will allow the currently running process to finish its CPU burst.
9. Preemptive SJF scheduling is sometimes called shortest remaining time first scheduling.

**Advantages of Shortest Job First (SJF) algorithm :**

1. It gives superior turnaround time performance to shortest process next because a short job is given immediate preference to a running longer job.
  2. Throughput is high.
- Disadvantages of Shortest Job First (SJF) algorithm :**
1. Elapsed time (i.e., execution-completed-time) must be recorded, it results an additional overhead on the processor.
  2. Starvation may be possible for the longer processes.

**Que 3.19.** Discuss priority scheduling algorithm. What are its advantages and disadvantages ?

1. Priority scheduling is a non pre-emptive algorithm and one of the most common scheduling algorithms in batch systems.

2. Each process is assigned a priority. Process with highest priority is to be executed first and so on.
  3. Processes with same priority are executed on FCFS basis.
  4. Priority can be decided based on memory requirements, time requirements or any other resource requirements.
- Advantages of priority scheduling algorithm :**
1. The priority of a process can be selected based on memory requirement, time requirement or user preference.
  2. For example, a high end game will have better graphics that means the process which updates the screen in a game will have higher priority so as to achieve better graphics performance.
  3. Priority scheduling provides a good mechanism where the relative importance of each process may be precisely defined.
- Disadvantages of priority scheduling algorithm :**
1. If high priority processes use up a lot of CPU time, lower priority processes may starve and be postponed indefinitely.
  2. The situation where a process never gets scheduled to run is called starvation.
  3. Another problem with priority scheduling is deciding which process gets which priority level assigned to it.

**Ques 3.20** Discuss Round Robin scheduling algorithm. Give its advantages and disadvantages.

**Answer**

- L* The Round Robin (RR) scheduling algorithm is designed especially for time sharing systems.
2. It is similar to FCFS scheduling, but pre-emption is added to enable the system to switch between processes.
  3. The ready queue is treated as a circular queue.
  4. The CPU scheduler goes around the ready queue, allocating the CPU to each process for a time interval of up to 1 time quantum.
  5. A small unit of time is called a time quantum or time slice.
  6. A time quantum is generally from 10 to 100 milliseconds in length.
  7. To implement RR scheduling, we again treat the ready queue as a FIFO queue of processes.
  8. New processes are added to the tail of the ready queue.
  9. The CPU scheduler picks the first process from the ready queue, sets a timer to interrupt after 1 time quantum, and dispatches the process.

- Advantages of Round Robin (RR) algorithm :**
1. Round-Robin is effective in a general-purpose, time sharing system or transaction processing system.
  2. Fair treatment for all the processes.
  3. Overhead on processor is low.
  4. Good response time for short processes.
- Disadvantages of Round Robin (RR) algorithm :**
1. Care must be taken in choosing quantum value.
  2. Processing overhead is there in handling clock interrupt.
  3. Throughput is low if time quantum is too small.

**Que 3.21** What is CPU scheduling? Discuss the multilevel feedback queue scheduling.

OR

Explain the functioning of multilevel feedback queue scheduling.

**AKTU 2013-14, Marks 10**

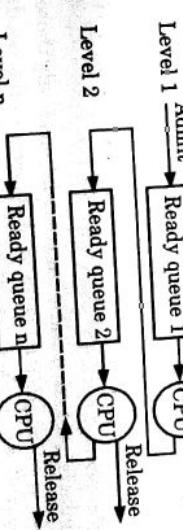
**Answer**

CPU scheduling : Refer Q. 3.1, Page 3-2B, Unit-3.

**Multilevel feedback queue scheduling:**

1. Multilevel Feedback Queue (MFQ) scheduling algorithm overcomes the problem of multilevel queue scheduling algorithm.
2. MFQ means to separate processes with different CPU burst time. If a process uses too much CPU time, it will be moved to a lower priority queue.

3. MFQ allows a process to move between the queues.
4. MFQ implements two or more scheduling queues.



**Que 3.21.1 Multilevel feedback queue**

5. For example, each process may start at the top level queue. If the process is completed within a given time slice, it departs the system.
6. Processes that need more than one time slice may be reassigned by the operating system to a lower priority queue, which gets a lower percentage of the processor time.

Multilevel feedback queue scheduler is defined by the following parameters

1. The number of queues.
2. Scheduling algorithm for each queue.
3. Method used to determine when to denote a process to a lower priority queue.
4. Method used to determine when to upgrade a process to a higher priority queue.
5. Method used to determine which queue a process will enter when the process needs service.

**Ques 3.22:** Explain the following scheduling algorithms :

- i. Multilevel feedback queue scheduling
- ii. Multiprocessor scheduling

**AKTU 2014-15, Marks 10**

### Answer

i. **Multilevel feedback queue scheduling :** Refer Q. 3.21, Page 3-21B Unit-3

ii. **Multiprocessor scheduling :**

1. On a multiprocessor, scheduling is two dimensional. The scheduler has to decide which process to run and which CPU to run it on. This extra dimension greatly complicates scheduling on multiprocessors.
2. Another complicating factor is that in some systems, all the processes are unrelated whereas in others they come in groups.
3. An example of the former situation is a timesharing system in which independent users start up independent processes. The processes are unrelated and each one can be scheduled without regard to the other ones.
4. Large systems often consist of some number of header files containing macros, type definitions, and variable declarations that are used by the actual code files.
5. When a header file is changed, all the code files that include it must be recompiled. The program make is commonly used to manage development.
6. When make is invoked, it starts the compilation of only those code files that must be recompiled on account of changes to the header or code files. Object files that are still valid are not regenerated.
7. The original version of make did its work sequentially, but newer versions designed for multiprocessors can start up all the compilations at once.

**Ques 3.23:** Consider a variant of Round Robin scheduling algorithm where the entries in the ready queue are pointers to the

processes. What would be the effect of putting two pointers to the same process in the ready queue ? What would be advantages and disadvantages of this scheme ?

**AKTU 2012-13, Marks 05**

### Answer

1. Process appears twice in the ready queue and is scheduled twice as often as other processes.
2. In effect, that process will have increased its priority since by getting time more often it is receiving preferential treatment. It will double the time given to that process.
3. The advantage of this scheme is that more important jobs could be given more time. It will provide higher priority with minimal modification to scheduler.
4. The disadvantage of this scheme is that the shorter job will suffer because important jobs will execute with more time. There will be overhead for managing pointers. It may also increase overhead if same process runs back-to-back.

**Ques 3.24:** Consider the set of the processes given in the table and the following scheduling algorithms :

- i. Round Robin (Quantum = 1)
- ii. Round Robin (Quantum = 2)
- iii. Shortest Remaining Job First

| Process Id | Arrival Time | Execution Time |
|------------|--------------|----------------|
| A          | 0            | 4              |
| B          | 2            | 7              |
| C          | 3            | 3              |
| D          | 3.5          | 3              |
| E          | 4            | 5              |

If there is tie within the processes, the tie is broken in the favour of the oldest process. Draw the Gantt chart and find the average waiting time, response time and turnaround time for the algorithms. Comment on your result. Which one is better and why?

**AKTU 2013-14, Marks 10**

### Answer

i. Round Robin (Quantum = 1)

|   |   |   |   |   |   |   |   |   |   |    |    |    |    |    |    |
|---|---|---|---|---|---|---|---|---|---|----|----|----|----|----|----|
| A | A | B | C | D | E | A | B | C | D | E  | B  | E  | B  | E  | B  |
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |

16 17 18 19 20 21 22 23

Waiting time for process,

$$A = 0 + 4 + 4 = 8$$

$$B = 2 + 4 + 4 + 3 + 1 + 1 + 1 = 16$$

$$C = 3 + 4 + 4 = 11$$

$$D = 4 + 4 + 4 = 12$$

$$E = 5 + 4 + 4 + 1 + 1 + 1 = 16$$

$$\text{Average waiting time} = \frac{8 + 16 + 11 + 12 + 16}{5} = 12.6$$

Turnaround time for process,

$$A = 12$$

$$B = 21$$

$$C = 11$$

$$D = 11.5$$

$$E = 18$$

Average turnaround time,

$$= \frac{12 + 21 + 11 + 11.5 + 18}{5} = 14.7$$

Response time for process,

$$A = 0$$

$$B = 0$$

$$C = 0$$

$$D = 4 - 3.5 = 0.5$$

$$E = 5 - 1 = 4$$

Average response time,

$$= \frac{0.5 + 4}{5} = 0.9$$

ii. Round Robin (Quantum = 2)

| A  | B | C | D | E | A  | B  | C  | D  | E  | B  | E  | B  |
|----|---|---|---|---|----|----|----|----|----|----|----|----|
| 0  | 2 | 4 | 6 | 8 | 10 | 12 | 14 | 15 | 16 | 18 | 20 | 21 |
| 22 |   |   |   |   |    |    |    |    |    |    |    |    |

Waiting time for process,

$$A = 8$$

$$B = 2 + 8 + 6 + 2 = 18$$

$$C = 4 + 8 = 12$$

$$D = 8 + 8 = 16$$

$$E = 8 + 8 + 2 = 18$$

Average waiting time,

$$= \frac{8 + 18 + 12 + 16 + 18}{5} = 14.4$$

Turnaround time for process,

$$A = 12$$

$$B = 20$$

$$C = 11$$

$$D = 12.5$$

$$E = 17$$

Average turnaround time,

$$= \frac{12 + 20 + 11 + 12.5 + 17}{5} = 14.5$$

Response time for process,

$$A = 0$$

$$B = 0$$

$$C = 4 - 3 = 1$$

$$D = 6 - 3.5 = 2.5$$

$$E = 8 - 4 = 4$$

Average response time,

$$= \frac{7.5}{5} = 1.5$$

iii. Shortest Remaining Job First

| A | A | A | C | D | E  | B  |
|---|---|---|---|---|----|----|
| 0 | 2 | 3 | 4 | 7 | 10 | 15 |

Waiting time for process,

$$A = 0$$

$$B = 15$$

$$C = 4$$

$$D = 7$$

$$E = 10$$

Average waiting time,

$$= \frac{0 + 15 + 4 + 7 + 10}{5} = 7.2$$

Turnaround time for process,

$$A = 4$$

$$B = 20$$

$$C = 4$$

$$D = 6.5$$

$$E = 6$$

Average turnaround time,

$$= \frac{4 + 20 + 4 + 6.5 + 6}{5} = 8.1$$

Response time for process,

$$A = 0$$

$$B = 15 - 2 = 13$$

$$C = 4 - 3 = 1$$

$$D = 7 - 3.5 = 3.5$$

$$E = 10 - 4 = 6$$

$$\text{Average response time} = \frac{23.5}{5} = 4.7$$

SJF gives the minimum average waiting time as compared to RR algorithm. So, it is the best scheduling algorithm.

**Que 3.25.** Consider the processes, CPU burst time and arrival time given below:

| Processes      | CPU burst time | Arrival time |
|----------------|----------------|--------------|
| P <sub>1</sub> | 8              | 0            |
| P <sub>2</sub> | 4              | 1            |
| P <sub>3</sub> | 9              | 2            |
| P <sub>4</sub> | 5              | 3            |

Draw the Gantt chart and calculate the following by using SJF CPU scheduling algorithm,

- Average waiting time
- Average turnaround time

**Answer**

**AKTU 2015-16, Marks 10**

Gantt Chart:

| P <sub>1</sub> | P <sub>2</sub> | P <sub>4</sub> | P <sub>1</sub> | P <sub>3</sub> |
|----------------|----------------|----------------|----------------|----------------|
| P <sub>1</sub> | 0              | 8              |                |                |
| P <sub>2</sub> | 1              | 4              |                |                |
| P <sub>3</sub> | 2              | 9              |                |                |
| P <sub>4</sub> | 3              | 5              |                |                |

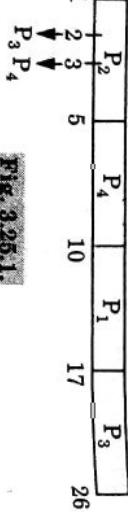


Fig. 3.25.1.

Waiting time,

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

$$P_2 = 1 - 1 = 0$$

$$P_3 = 17 - 2 = 15$$

$$P_4 = 5 - 3 = 2$$

$$\text{Average waiting time} = \frac{9 + 0 + 15 + 2}{4}$$

$$= \frac{26}{4}$$

$$= 6.5$$

Turnaround time (TAT),

$$P_1 = 17 - 0 = 17$$

$$P_2 = 5 - 1 = 4$$

$$P_3 = 26 - 2 = 24$$

$$P_4 = 10 - 3 = 7$$

$$\text{Average turnaround time} = \frac{17 + 4 + 24 + 7}{4}$$

$$= \frac{52}{4}$$

$$= 13$$

**Que 3.26.** List various performance criteria for scheduling algorithms. Five processes A, B, C, D, E require CPU burst of 3, 5, 2, 5 and 5 units respectively. Their arrival times in the system are 0, 1, 3, 9 and 12 respectively. Draw Gantt Chart and compute the average turnaround time and average waiting time of these processes for the Shortest Job First (SJF) and Shortest Remaining Time First (SRTF) scheduling algorithms.

**AKTU 2011-12, Marks 10**

**Answer**

Performance criteria for scheduling algorithm : Refer Q. 3.1,

Page 3-2B, Unit-3.

Numerical :

SJF:

| A | C | B | D  | E  |
|---|---|---|----|----|
| 0 | 1 | 5 | 10 | 15 |

Turnaround Time

A = 3

B = 9

C = 2

D = 6

1

**S → S**

Average turnaround time

 $(3 + 9 + 2 + 6 + 8)/5 = 5.6 \text{ ms}$ 

Average waiting time

 $(0 + 4 + 0 + 1 + 3)/5 = 1.6 \text{ ms}$ **SRTF:**

| A                       | C | B | D  | E                                        |
|-------------------------|---|---|----|------------------------------------------|
| 0                       | 1 | 5 | 10 | 15                                       |
| T turnaround Time       |   |   |    | 20                                       |
| A = 3                   |   |   |    | Waiting Time                             |
| B = 9                   |   |   |    | 0                                        |
| C = 2                   |   |   |    | 4                                        |
| D = 6                   |   |   |    | 0                                        |
| E = 8                   |   |   |    | 1                                        |
| Average turnaround time |   |   |    | $(3 + 9 + 2 + 6 + 8)/5 = 5.6 \text{ ms}$ |
| Average waiting time    |   |   |    | $(0 + 4 + 0 + 1 + 3)/5 = 1.6 \text{ ms}$ |

| A                       | C | B | D  | E                                        |
|-------------------------|---|---|----|------------------------------------------|
| 0                       | 1 | 5 | 10 | 15                                       |
| T turnaround Time       |   |   |    | 20                                       |
| A = 3                   |   |   |    | Waiting Time                             |
| B = 9                   |   |   |    | 0                                        |
| C = 2                   |   |   |    | 4                                        |
| D = 6                   |   |   |    | 0                                        |
| E = 8                   |   |   |    | 1                                        |
| Average turnaround time |   |   |    | $(3 + 9 + 2 + 6 + 8)/5 = 5.6 \text{ ms}$ |
| Average waiting time    |   |   |    | $(0 + 4 + 0 + 1 + 3)/5 = 1.6 \text{ ms}$ |

**Q-3.27** What is deadlock? What are the necessary conditions for deadlock?**Q-3.28** What are the necessary conditions to hold a deadlock in a system?**OR****Q-3.29** What are the necessary conditions for deadlock with examples?**AKTU 2013-14, Marks 06****Answer**

What is a deadlock? Discuss the necessary conditions for deadlock with examples.

**Answer**

1. A deadlock is a situation where a group of processes are permanently blocked as a result of each process having acquired a subset of the resources needed for its completion and waiting for release of the remaining resource held by other in the same group, thus making it impossible for any of the processes to proceed.

2. Resource managers and other operating system, processes can be involved in a deadlock situation.

A deadlock situation can arise if the following four conditions holds simultaneously in a system :

1. **Mutual exclusion :** At least one resource must be held in a non-shareable mode; that is, only one process at a time can use the resource. If another process requests that resource, the requesting process must be delayed until the resource has been released.

2. **Hold and wait :** A process must be holding at least one resource and waiting to acquire additional resources that are currently being held by other processes.

3. **No pre-emption :** Resources cannot be pre-empted; i.e., a resource can be released only voluntarily by the process holding it, after that process has completed its task.

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

**Ques 3.28** How deadlock are prevented?

OR

Define and explain the deadlock prevention.

OR

What are the approaches that can be used for prevention of deadlock?

**AKTU 2013-14, Marks 05****Answer**

Deadlock prevention is an approach which ensures that system will never enter in deadlock state.

These are following approaches of deadlock prevention :

1. **Mutual exclusion :**

- a. Mutual exclusion condition must hold for non-shareable resources.
- b. If access to a resource requires mutual exclusion, then mutual exclusion must be supported by the operating system.

2. **Hold and wait :**

- a. The hold and wait condition can be eliminated by forcing a process to release all resources held by it, whenever it requests a resource that is not available.
- b. For example, process copies data from a floppy disk to a hard disk, sort a disk file and then prints the results to a printer.

c. If all the resources must be requested at the beginning of the process, then the process must initially request the floppy disk, hard disk and a printer.

- d. It will hold the printer for its entire execution, even though it needs the printer only at the end.

3. **No pre-emption :**

- a. If a process holding certain resources is denied a further request. That process must release its original resources and if necessary request them again, together with additional resource.
- b. If a process requests a resource that is currently held by another process, the operating system may preempt the second process and require it to release its resources.
- c. Pre-emption is possible for certain types of resources, such as CPU and main memory.

4. **Circular wait :**

- a. One way to prevent the circular wait condition is by linear ordering of different types of system resources.

- b. In this, system resources are divided into different classes.  
 c. If a process has been allocated resources of type  $R$ , then it may subsequently request only those resource types following  $R$  in the ordering.

**Ques 3.29.** What is deadlock avoidance ? Define all states of a system.

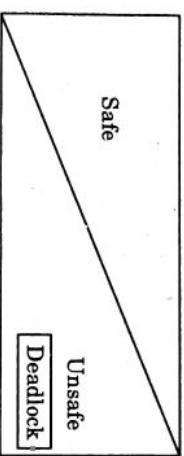
**Answer**

- Deadlock avoidance allows the three necessary conditions but makes judicious choices to assure that the deadlock point is never reached.
- Deadlock avoidance therefore allows more concurrency than prevention does.
- Deadlock avoidance requires additional information about how resources are to be requested.
- With deadlock avoidance, a decision is made dynamically whether the current resource allocation request could, if granted, potentially lead to a deadlock.

Two approaches are used to avoid the deadlock :

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

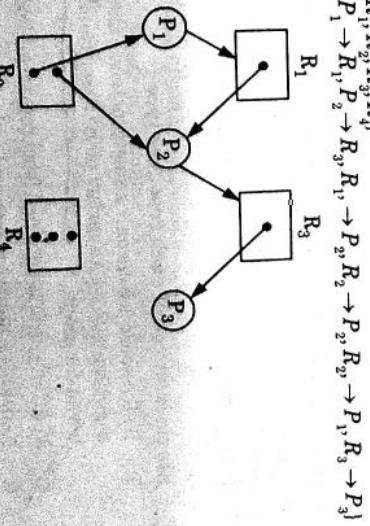
System can be in one of the following states :



**Fig. 3.29.1.** Relationship between safe, unsafe and a deadlock state.

- Safe state :** Such a state occurs when the system can allocate resources to each process (up to its maximum) in some order and avoid a deadlock. This state will be characterized by a safe sequence.
- 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, then it is in an unsafe state.
- Deadlock state :** If the system has some circular wait condition existing for some processes, then it is in deadlock.

**Ques 3.30.** Describe resource-allocation graph.



**Fig. 3.30.1.** Resource allocation graph.

**Answer**

- Resource-allocation graph is used to describe the deadlock. It is also called system resource allocation graph.
- Graph consists of a set of vertices ( $V$ ) and set of edges ( $E$ ).
- All the active processes in the system denoted by  $P = [P_1, P_2, \dots, P_n]$  and set consisting of all resource type in the system is denoted by  $R = [R_1, R_2, R_3, \dots, R_m]$ .
- Request edge is an edge from process to resource and denoted by  $P_i \rightarrow R_j$ .
- An assignment edge is an edge from resource to process and denoted by  $R_j \rightarrow P_i$ .
- Holding of resource by process is denoted by assignment edge.
- Requesting of resource by process is denoted by request edge.
- For representing process and resource in the resource allocation graph is shown by square and circle. Each process is represented by circle and resource by square. Dot within the square represents the number of instance.
- Consider the following system which consists of three processes i.e.,  $P_1, P_2$  and  $P_3$  and four resources i.e.,  $R_1, R_2, R_3$  and  $R_4$ . Resource  $R_1$  and  $R_3$  have one instance,  $R_2$  has two instances and  $R_4$  has three instances.
  - Process  $P_1$  is holding an instance of resource type  $R_2$  and is waiting for an instance of resource type  $R_1$ .
  - Process  $P_2$  is holding an instance of  $R_1$  and  $R_2$  and waiting for an instance of resource type  $R_3$ .
  - Process  $P_3$  is holding an instance of  $R_3$ .
- $P = \{P_1, P_2, P_3\}$   
 $R = \{R_1, R_2, R_3, R_4\}$   
 $E = (P_1 \rightarrow R_1, P_2 \rightarrow R_3, R_1 \rightarrow P_2, R_2 \rightarrow P_3, R_2 \rightarrow P_1, R_3 \rightarrow P_3)$

**Que 3.31.** Define deadlock detection. Write an algorithm for deadlock detection.

**Answer**

1. Deadlock detection is the process of actually determining that a deadlock exists and identifying the processes and resources involved in the deadlock.
2. The deadlock detection algorithm is invoked when the allocation cannot be granted immediately and this condition occurs frequently.
3. The basic idea is to check allocation against resource availability for all possible allocation sequences to determine if the system is in deadlocked state.

**Deadlock detection algorithm :**

**Step 1:** Let work and finish be vector of length  $m$  and  $n$  respectively.

Initialize

Work = Available

If Allocation<sub>i</sub> ≠ 0, then Finish[i] = false

Otherwise, Finish[i] = true For  $i = 1, 2, \dots, n$

**Step 2:** Find an index  $i$  such that both

Finish[i] = false

Request<sub>i</sub> ≤ Work

**Step 3:** If no such  $i$  exists, go to step 4.

**Step 4:** Work = Work + Allocation<sub>i</sub>

Finish[i] = true

Go to step 2

**Step 5:** If  $[i] = \text{false}$ , for some  $i, 1 \leq i \leq n$  then system is in deadlock state.

Moreover, if Finish[i] = false, then  $P_i$  is deadlocked.

**Que 3.32.** What are the two ways in which deadlock can be detected?

**Answer**

Following are the ways in which deadlock can be detected :

i. **Single instance of each resource type :**

1. If all resources have only a single instance, then deadlock detection algorithm that uses a variant of the resource-allocation graph, called await-for graph.
2. Wait-for graph is obtained from resource-allocation graph.
3. Nodes of resource are removed and collapsing the appropriate edge i.e., assignment and request edge.

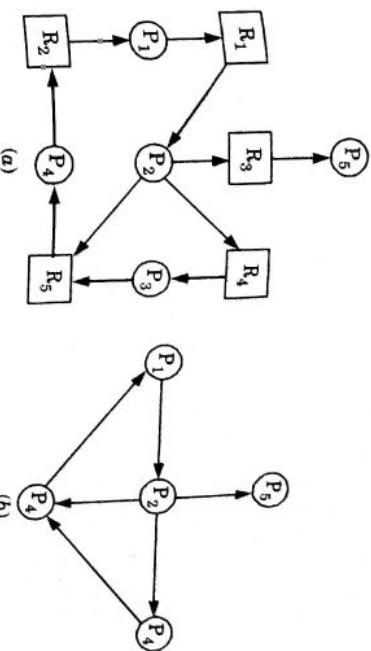


Fig. 3.32.1. Resource-allocation graph with corresponding wait-for graph.

4. The assumptions made in the wait-for graph are as follows :

a. An edge from  $P_i$  to  $P_j$  in a wait-for graph implies that process  $P_i$  is waiting for process  $P_j$  to release a resource that  $P_i$  needs.

b. An edge  $P_i \rightarrow P_j$  exists in a wait-for graph if and only if the corresponding resource-allocation graph contains two edges  $P_i \rightarrow R_q$  and  $R_q \rightarrow P_j$  for some resource  $R_q$ .

5. If WFG contains cycle then the system is deadlocked otherwise is in safe state.

ii. **Several instances of a resource type :**

1. Unmark all active processes from Allocation, Max and Available in accordance with the system state.

2. Find an unmarked process  $i$  such that

If found, mark process  $i$ , update Available  
Available := Available + Allocation

Max := Available + Allocation

and repeat this step.

If no process is found, then go to next step.

If all processes are marked, the system is not deadlocked.

3. If all processes are marked, the system is not deadlocked.

**Que 3.33.** "A safe state is not a deadlock state but a deadlock state is an unsafe state". Explain. Discuss the different methods for handling deadlocks.

**Answer**

1. A safe state is a state in which there is at least one order in which all the processes can be run to completion without resulting in a deadlock.
2. A sequence of processes  $\langle P_1, P_2, \dots, P_n \rangle$  is a safe sequence for the current allocation state if, for each  $P_i$  the resources that  $P_i$  can still

request can be satisfied by the currently available resources plus the resources held by all the  $P_j$  with  $j < i$ .

3. A safe state is not a deadlock state. Conversely, a deadlock state is an unsafe state. Not all unsafe states are deadlocks. An unsafe state may lead to a deadlock.
4. As long as the state is safe, an operating system can avoid unsafe state.
5. In an unsafe state, the operating system cannot prevent processes from requesting resource such that a deadlock occurs.

Methods for handling deadlock are:

1. Deadlock prevention : Refer Q. 3.28, Page 3-29B, Unit-3.
2. Deadlock avoidance : Refer Q. 3.29, Page 3-30B, Unit-3.
3. Deadlock detection : Refer Q. 3.31, Page 3-32B, Unit-3.

**Que 3.34** Define deadlock. List four necessary conditions for occurrence of deadlock. A system contains 6 units of resource, and processes that use the resource. What is the maximum value of  $n$  for which the system will be deadlock free if the maximum requirement of each process is 3 ?

**AKTU 2012-13, Marks 6**

**Answer**  
Deadlock and its four necessary conditions : Refer Q. 3.27, Page 3-28B, Unit-3.

#### Maximum value of $n$ :

The maximum value of  $n$  for which the system is guaranteed to be deadlock free is 2. Two processes can never lead to deadlock as the peak time demand of ( $3 + 3 = 6$ ) resources can be satisfied. But 3 processes can lead to deadlock if each process holds 2 resources and then demand on more.

**Que 3.35.** Write an algorithm for detection of deadlock in a system having several instances of multiple resource types.

**AKTU 2012-13, Marks 10**

#### Answer

The algorithm uses several time-varying data structures which are as follows:

1. Available : A vector of length  $m$  indicates the number of available resources of each type.
2. Allocation : An  $n \times m$  matrix defines the number of resources of each type currently allocated to each process.

**Request :** An  $n \times m$  matrix indicates the current request of each process. If it  $\text{Request}[i][j]$  equals  $k$ , then process  $P_i$  is requesting  $k$  more instances of resource type  $R_j$ .

#### Algorithm :

1. Let Work and Finish be vectors of length  $m$  and  $n$ , respectively.
2. Initialize Work = Available.
3. For  $i = 0, 1, \dots, n - 1$ , if  $\text{Allocation}_i \neq 0$ , then  $\text{Finish}[i] = \text{false}$ ; otherwise,  $\text{Finish}[i] = \text{true}$ .
4. Find an index  $i$  such that both  $\text{Request}_i \leq \text{Work}$
5. If no such  $i$  exists, go to step (4).
6. Work = Work + Allocation <sub>$i$</sub>
7. Finish[ $i$ ] = true
8. Go to step (2).
9. If  $\text{Finish}[i] == \text{false}$  for some  $i$ ,  $0 \leq i < n$  then the system is in a deadlock state.

Moreover if  $\text{Finish}[i] == \text{false}$  then process  $P_i$  is deadlock. This algorithm requires an order of  $m \times n^2$  operations to detect whether the system is in deadlock state.

**Que 3.36.** Write the Banker's algorithm and how it can be used to avoid can be used to avoid deadlock.

OR

Describe Banker's algorithm for safe allocation.

**AKTU 2016-17, Marks 7.5**

OR

Write and explain Banker's algorithm for avoidance of deadlock.

**AKTU 2011-12, Marks 10**

#### Answer

#### Banker's algorithm :

The banker's algorithm is a resource allocation and deadlock avoidance algorithm that tests for safety by simulating the allocation for pre-determined maximum possible amounts of all resources, then makes an "s-state" check to test for possible activities, before deciding whether allocation should be allowed to continue.

Following data structures are used to implement the Banker's algorithm : Let ' $n$ ' be the number of processes in the system and ' $m$ ' be the number of resources types.

- 1. Available :** It is a 1D array of size ' $m$ ' indicating the number of available resources of each type.

- 2. Max :** It is a 2D array of size ' $n*m$ ' that defines the maximum demand of each process in a system.

- 3. Allocation :** It is a 2D array of size ' $n*m$ ' that defines the number of resources of each type currently allocated to each process.

- 4. Need :** It is a 2D array of size ' $n*m$ ' that indicates the remaining resource need of each process.

Need [ $i, j$ ] =  $k$  means process  $P_i$  currently allocated ' $k$ ' instances of resource type  $R_j$

$$\text{Need } [i, j] = \text{Max } [i, j] - \text{Allocation } [i, j]$$

Allocation <sub>$i$</sub>  specifies the resources currently allocated to process  $P_i$  and Need <sub>$i$</sub>  specifies the additional resources that process  $P_i$  may still request to complete its task.

Banker's algorithm is used to avoid deadlock by following algorithms :

**a. Safety algorithm :**

1. Let work and finish be vector of length  $m$  and  $n$  respectively Initialize Work = Available and Finish [ $i$ ] = False for  $i = 1, 2, 3, 4, \dots, n$ .

2. Find an  $i$  such that both

- a. Finish [ $i$ ] = False
- b. Need [ $i$ ]  $\leq$  Work

If no such  $i$  exist, goto step 4.

3. Work = Work + Allocation <sub>$i$</sub>

Finish [ $i$ ] = True

Goto step 2

4. If Finish [ $i$ ] = True for all  $i$ , then the system is in a safe state.

- b. Resource request algorithm :** Let Request <sub>$i$</sub>  be the request array for process  $P_i$ . Request <sub>$i$</sub> [ $j$ ] =  $k$  means process  $P_i$  wants  $k$  instances of resource type  $R_j$ . When a request for resources is made by process  $P_i$ , the following actions are taken :

1. If Request <sub>$i$</sub>   $\leq$  Need <sub>$i$</sub> ,

Goto step 2; otherwise, raise an error condition, since the process has exceeded its maximum claim.

2. If Request <sub>$i$</sub>   $\leq$  Available

- Que 3.37.** Describe Banker's algorithm for deadlock avoidance. Consider a system with three processes and three resources. The snapshot of a system at time  $t_0$  is given below :

| Processes | Allocation |   |   | Max |   |   | Available |   |    |
|-----------|------------|---|---|-----|---|---|-----------|---|----|
|           | A          | B | C | A   | B | C | A         | B | C  |
| $P_0$     | 2          | 2 | 3 | 3   | 6 | 8 | 7         | 7 | 10 |
| $P_1$     | 2          | 0 | 3 | 4   | 3 | 3 |           |   |    |
| $P_2$     | 1          | 2 | 4 | 3   | 4 | 4 |           |   |    |

- i. Is the current allocation in safe state ?  
ii. Would the following requests be granted in the current state :

- a. Process  $P_2$  requests (1, 0, 0)  
b. Process  $P_1$  requests (1, 0, 0)

**AKTU 2014-15, Marks 10**

**Answer**

**Banker's algorithm :** Refer Q. 3.36, Page 3-35B, Unit-3.

- i. First find the available resources in the system  
Available = Number of instance - sum of allocation

| Process | A | B | C |
|---------|---|---|---|
| $P_0$   | 2 | 2 | 3 |
| $P_1$   | 2 | 0 | 3 |
| $P_2$   | 1 | 2 | 4 |

$$\text{Available} = (7\ 7\ 10) - (5\ 4\ 10)$$

$$\text{Available resources} = (2\ 3\ 0)$$

Goto step 3; otherwise,  $P_i$  must wait, since the resources are not available.

Have the system pretend to have allocated the requested resources to process  $P_i$  by modifying the state as follows :

$$\text{Available} = \text{Available} - \text{Request}_i$$

$$\text{Allocation}_i = \text{Allocation}_i + \text{Request}_i$$

$$\text{Need}_i = \text{Need}_i - \text{Request}_i$$

| Process | Need |   |   |  |
|---------|------|---|---|--|
| $P_0$   | 1    | 4 | 5 |  |
| $P_1$   | 2    | 3 | 0 |  |
| $P_2$   | 2    | 2 | 0 |  |

Safe sequence  $<P_2, P_1, P_0>$ .

The system is in safe state.

- ii. a. Process  $P_2$  request  $(1, 0, 0)$ , this request is less than need. Need for process  $P_2$  is  $(2, 2, 0)$ , available resource is  $(2, 3, 0)$  and request is  $(1, 0, 0)$ . After allocating  $(1, 0, 0)$  to process  $P_2$  the need becomes as follows :

| Process | Need |   |   |  |
|---------|------|---|---|--|
| $P_0$   | 1    | 4 | 5 |  |
| $P_1$   | 2    | 3 | 0 |  |
| $P_2$   | 1    | 2 | 0 |  |

And available resource is  $(1, 3, 0)$ . We see that system is in safe state with safe sequence  $<P_2, P_1, P_0>$ . Hence request of  $P_2(1, 0, 0)$  will be granted.

- b. Process  $P_1$  request  $(1, 0, 0)$ , this request is less than need. Need for process  $P_1$  is  $(2, 3, 0)$ . Available resource is  $(1, 3, 0)$  and request is  $(1, 0, 0)$ . Request  $<\text{Available}, (1, 0, 0) < (1, 3, 0)$

After allocating  $(1, 0, 0)$  to process  $P_1$  the need becomes as follows :

| Process | Need |   |   |  |
|---------|------|---|---|--|
| $P_0$   | 1    | 4 | 5 |  |
| $P_1$   | 2    | 3 | 0 |  |
| $P_2$   | 1    | 2 | 0 |  |

And available resource is  $(0, 3, 0)$ . Need of any process is never satisfied after granting the  $P_1$  request  $(1, 0, 0)$ . So, the system will be blocked. Therefore, request of  $P_1(1, 0, 0)$  cannot be granted.

Que 3.Ms. Consider the following snapshot of the system :

|       | Allocation |   |   |   | Max |   |   |   | Available |   |   |   |
|-------|------------|---|---|---|-----|---|---|---|-----------|---|---|---|
|       | A          | B | C | D | A   | B | C | D | A         | B | C | D |
| $P_0$ | 0          | 0 | 1 | 2 | 0   | 0 | 1 | 2 | 1         | 5 | 2 | 0 |
| $P_1$ | 1          | 0 | 0 | 0 | 1   | 7 | 5 | 0 |           |   |   |   |
| $P_2$ | 1          | 3 | 5 | 4 | 2   | 3 | 5 | 6 |           |   |   |   |
| $P_3$ | 0          | 6 | 3 | 2 | 0   | 6 | 5 | 2 |           |   |   |   |
| $P_4$ | 0          | 0 | 1 | 4 | 0   | 6 | 5 | 6 |           |   |   |   |

Answer the following questions using the banker's algorithm : i. What is the content of the matrix need ?

ii. Is the system in a safe state ? If yes then find the safe sequence.

- iii. If a request from process  $P_1$  arrives for  $(0, 4, 2, 0)$  can the request be granted immediately ?

AKTU 2015-16, Marks 10

i. The need matrix can be calculated according to Banker's algorithm.

Need<sub>d</sub> = Max<sub>r</sub> - Allocation<sub>r</sub>  
Need matrix

|       | A | B | C | D |
|-------|---|---|---|---|
| $P_0$ | 0 | 0 | 0 | 0 |
| $P_1$ | 0 | 7 | 5 | 0 |
| $P_2$ | 1 | 0 | 0 | 2 |
| $P_3$ | 0 | 0 | 2 | 0 |
| $P_4$ | 0 | 6 | 4 | 2 |

ii. Yes, system is in safe state. This can be seen as below :

Since, we have available =  $(1, 5, 2, 0)$  which can be allocated to  $P_0$  or  $P_2$  to satisfy the need or request

According to safety algorithm,

A      B      C      D  
Work =    1      5      2      0 (Available)  
          A      B      C      D      A      B      C      D

$$\text{Need } [P_0] = \begin{matrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{matrix} < \begin{matrix} 1 & 5 & 2 & 0 \\ 0 & 0 & 1 & 2 \end{matrix} \quad (\text{Available})$$

Hence, process  $P_0$  will release the allocated resources. Hence,

$$\begin{aligned} \text{Work} &= \text{Work} + \text{allocation} \\ &= \begin{matrix} A & B & C & D & A & B & C & D \\ 1 & 5 & 2 & 0 & + & 0 & 0 & 1 & 2 \end{matrix} \\ &= \begin{matrix} A & B & C & D & A & B & C & D \\ 1 & 5 & 3 & 2 & & & & \end{matrix} \quad (\text{Available}) \end{aligned}$$

### 3-40 B (CSIT-Sem-4)

#### CPU Scheduling

This can satisfy the request of process  $P_2$ . Since

$$\text{Need } [P_2] = \begin{bmatrix} 1 & 0 & 0 & 2 \\ 1 & 5 & 3 & 2 \end{bmatrix} < \begin{bmatrix} 1 & 5 & 3 & 2 \end{bmatrix}$$

Hence, process  $P_2$  will also release the resource. Hence

$$\begin{aligned} \text{Work} &= \text{Work} + \text{Allocation} \\ &= \begin{bmatrix} A & B & C & D \\ 1 & 5 & 3 & 2 \end{bmatrix} + \begin{bmatrix} A & B & C & D \\ 1 & 3 & 5 & 4 \end{bmatrix} \\ &= \begin{bmatrix} 2 & 8 & 8 & 6 \end{bmatrix} \text{ (Available)} \end{aligned}$$

This (Available) can satisfy the request process  $P_3$ .

Hence, resources will be allocated to it

$$\begin{aligned} \text{Work} &= \text{Work} + \text{Allocation} \\ &= \begin{bmatrix} A & B & C & D \\ 2 & 8 & 8 & 6 \end{bmatrix} + \begin{bmatrix} A & B & C & D \\ 0 & 6 & 3 & 2 \end{bmatrix} \\ &= \begin{bmatrix} A & B & C & D \\ 2 & 14 & 11 & 8 \end{bmatrix} \text{ (Available)} \end{aligned}$$

This (Available) resource, can be allocated to  
Process  $P_4$ . Since

$$\text{Need } [P_4] = \begin{bmatrix} A & B & C & D \\ 0 & 6 & 4 & 2 \end{bmatrix} < \begin{bmatrix} A & B & C & D \\ 2 & 14 & 11 & 8 \end{bmatrix}$$

Hence, after completion processes  $P_4$  will also release the resource.

$\text{Work} = \text{Work} + \text{Allocation}$

$$\begin{aligned} &= \begin{bmatrix} A & B & C & D \\ 2 & 14 & 11 & 8 \end{bmatrix} + \begin{bmatrix} A & B & C & D \\ 0 & 0 & 1 & 4 \end{bmatrix} \\ &= \begin{bmatrix} A & B & C & D \\ 2 & 14 & 12 & 12 \end{bmatrix} \end{aligned}$$

These resource, can be allocated to process  $P_1$

$$\text{Need } [P_1] = \begin{bmatrix} A & B & C & D \\ 0 & 7 & 5 & 0 \end{bmatrix} < \begin{bmatrix} A & B & C & D \\ 2 & 14 & 12 & 12 \end{bmatrix}$$

Hence safe sequence will be  $\langle P_0, P_2, P_3, P_4, P_1 \rangle$

iii. Since process  $P_1$  requests for the resources (0, 4, 2, 0)

Now, the matrices will be

|       | Allocation |   |   |   | Max |   |   |   | Available |   |   |   |
|-------|------------|---|---|---|-----|---|---|---|-----------|---|---|---|
|       | A          | B | C | D | A   | B | C | D | A         | B | C | D |
| $P_0$ | 0          | 0 | 1 | 2 | 0   | 0 | 1 | 2 | 1         | 1 | 0 | 0 |
| $P_1$ | 1          | 4 | 2 | 0 | 1   | 7 | 5 | 0 |           |   |   |   |
| $P_2$ | 1          | 3 | 5 | 4 | 2   | 3 | 5 | 6 |           |   |   |   |
| $P_3$ | 0          | 6 | 3 | 2 | 0   | 6 | 5 | 2 |           |   |   |   |
| $P_4$ | 0          | 0 | 1 | 4 | 0   | 6 | 5 | 6 |           |   |   |   |

Here, Available

$$\begin{aligned} &= \begin{bmatrix} A & B & C & D \\ 1 & 5 & 2 & 0 \end{bmatrix} - \begin{bmatrix} A & B & C & D \\ 0 & 4 & 2 & 0 \end{bmatrix} = \begin{bmatrix} 1 & 1 & 0 & 0 \end{bmatrix} \end{aligned}$$

Now the need matrix can be represented as

Need<sub>i</sub> = Max<sub>i</sub> - Allocation<sub>i</sub>

$$\begin{bmatrix} A & B & C & D \\ 0 & 0 & 0 & 0 \\ 0 & 3 & 3 & 0 \end{bmatrix}$$

#### Operating Systems

### 3-41 B (CSIT-Sem-4)

$$\sum_{i=1}^n \text{Max}_i < m+n$$

$$\text{Max}_i \geq 1 \text{ for all } i$$

$$\text{Proof: } \text{Need}_i = \text{Max}_i - \text{Allocation}_i$$

iii. If there exists a deadlock state then:

$$\sum_{i=1}^n \text{Allocation}_i \leq m$$

$$\sum_{i=1}^n \text{Max}_i > m+n$$

Use (i) to get:  $\sum \text{Need}_i + \sum \text{Allocation}_i = \sum \text{Max}_i < m+n$

Use (ii) to get:  $\sum \text{Need}_i + m < m+n$

Rewrite to get:  $\sum_{i=1}^n \text{Need}_i < n$

This implies that there exists a process  $P_i$  such that  $\text{Need}_i = 0$ . Since  $\text{Max}_i \geq 1$  it follows that  $P_i$  has at least one resource that it can release. Hence, the system cannot be in a deadlock state.

**Que 340.** Describe the Banker's algorithm for safe allocation. Consider a system with five processes and three resource types and at time  $t_0$  the following snapshot of the system has been taken:

| Process Id     | Allocated                                    | Maximum                                      | Available                                    |
|----------------|----------------------------------------------|----------------------------------------------|----------------------------------------------|
|                | R <sub>1</sub> R <sub>2</sub> R <sub>3</sub> | R <sub>1</sub> R <sub>2</sub> R <sub>3</sub> | R <sub>1</sub> R <sub>2</sub> R <sub>3</sub> |
| P <sub>1</sub> | 1 1 2                                        | 4 3 3                                        | 3 1 0                                        |
| P <sub>2</sub> | 2 1 2                                        | 3 2 2                                        | 0 6 4                                        |
| P <sub>3</sub> | 4 0 1                                        | 9 0 2                                        |                                              |
| P <sub>4</sub> | 0 2 0                                        | 7 5 3                                        |                                              |
| P <sub>5</sub> | 1 1 2                                        | 11 2 3                                       |                                              |

**Que 3-42.** Discuss the techniques to recover from deadlock.**Answer**

There are two options for recovering from a deadlock :

1. **Process termination** : To eliminate deadlocks by aborting a process, we use one of two methods.
  - a. **Abort all deadlocked processes** : This method clearly will break the deadlock cycle, but the deadlocked processes may have computed for a long time, and the results of these partial computations must be discarded and probably will have to be recomputed later.
  - b. **Abort one process at a time until the deadlock cycle is eliminated** : This method incurs considerable overhead, since after each process is aborted, a deadlock-detection algorithm must be invoked to determine whether any processes are still deadlocked.
2. **Resource pre-emption** : To eliminate deadlocks using resource pre-emption, we successively preempt some resources from processes and give these resources to other processes until the deadlock cycle is broken.
  - a. **Selecting a victim** :
    - i. Which resources and which processes are to be preempted? As in process termination, we must determine the order of pre-emption to minimize cost.
    - ii. Cost factors may include such parameters as the number of resources a deadlocked process is holding and the amount of time the process has thus far consumed during its execution.
  - b. **Rollback** :
    - i. If we preempt a resource from a process, then it cannot continue with its normal execution.
    - ii. It is missing some needed resource.
    - iii. We must roll back the process to some safe state and restart it from that state.
    - iv. Since, in general, it is difficult to determine what a safe state is, the simplest solution is a total rollback, abort the process and then restart it.
    - v. Although it is more effective to roll back the process only as far as necessary to break the deadlock, this method requires the system to keep more information about the state of all running processes.
  - c. **Kill the process**.

**VERY IMPORTANT QUESTIONS**  
*Following questions are very important. These questions may be asked in your SESSIONALS as well as UNIVERSITY EXAMINATION.*

- Q. 1. Explain performance criteria of CPU scheduling.  
ANS Refer Q. 3.1.
- Q. 2. What is CPU scheduling? Write the difference between pre-emptive and non pre-emptive scheduling.  
ANS Refer Q. 3.1 and Q. 3.3.
- Q. 3. Explain process transition diagram.  
ANS Refer Q. 3.5.
- Q. 4. Discuss Process Control Block (PCB).  
ANS Refer Q. 3.7.
- Q. 5. Write short notes on process identification information.  
ANS Refer Q. 3.10.
- Q. 6. Write short notes on following :
  - i. FCFS scheduling
  - ii. SJF scheduling
  - iii. Round robin scheduling
- Q. 7. Write the difference between following :
  1. Thread and process
  2. User level thread and kernel level thread
  3. Long term, short term and mid term scheduler
- Q. 8. Explain Banker's algorithm for avoidance of deadlock.  
ANS Refer Q. 3.36.
- Q. 9. Explain the following :
  1. Deadlock prevention
  2. Deadlock avoidance
  3. Deadlock detection

1. Refer Q. 3.28.
2. Refer Q. 3.29.
3. Refer Q. 3.31.

# 4

## UNIT

### Memory Management

**Part-1** ..... (4-2B to 4-2B)

- **Memory Management : Basic Bare Machine**
- **Resident Monitor**
- **Multiprogramming with Fixed Partitions**
- **Multiprogramming with Variable Partitions**
- **Protection Schemes**
- **Paging**

- **Segmentation**
- **Paged Segmentation**

A. Concept Outline : Part-1 ..... 4-2B

B. Long and Medium Answer Type Questions ..... 4-2B

**Part-2** ..... (4-2B to 4-41B)

- **Virtual Memory Concepts**
- **Demand Paging**
- **Performance of Demand Paging**
- **Page Replacement Algorithms**
- **Thrashing**
- **Cache Memory Organization**
- **Locality of Reference**

A. Concept Outline : Part-2 ..... 4-25B  
 B. Long and Medium Answer Type Questions ..... 4-25B

#### PART-1

**Memory Management : Basic Bare Machine, Resident Monitor, Multiprogramming with Fixed Partitions, Multiprogramming with Variable Partitions, Protection Schemes, Paging, Segmentation, Paged Segmentation.**

#### CONCEPT OUTLINE : PART-1

- 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.
- Paging is a memory management scheme that permits the physical address space of a process to be non contiguous.
- Segmentation is a memory management scheme that supports the programmer view of memory.

#### Questions-Answers

#### Long Answer Type and Medium Answer Type Questions

**Que 4.1.** Explain memory management and its requirements.

#### Answer

1. 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.
2. The overall resource utilization and other performance criteria of a computer system are largely affected by the performance of the memory management module.

Five requirements of memory management are :

1. **Relocation :**
  - a. Programmers do not know where the program will be placed in memory when it is executed.
  - b. While the program is executing, it may be swapped to disk and returned to main memory at different location (relocated).
  - c. Memory references must be translated in the code to actual memory address.

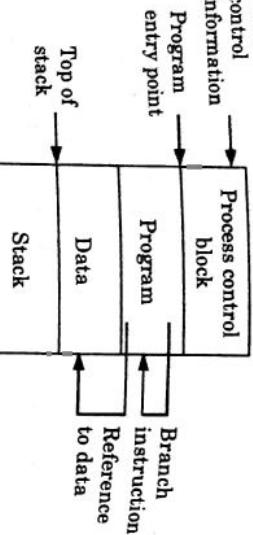


Fig. 4.1.1.

**2. Sharing:**

- a. Any protection mechanism must have the flexibility to allow several processes to access the same portion of main memory.
- b. For example if the number of processes are executing the same program then it is advantageous to allow each process to access the same copy of the program rather than its own separate copy.

**3. Protection :**

- a. Every process should protect against unwanted interference by other processes.
- b. Satisfaction of the relocation requirement increases the difficulty of satisfying the protection requirement.
- c. It is not possible to check the absolute address at compile time.
- d. Most of the programming languages allow the dynamic calculation of address at runtime.

**4. Logical organization :**

- a. Main memory is organized as a linear or one-dimensional address space that consists of sequence of bytes or words.
- b. Secondary memory at its physical level is similarly organized.
- c. Most of the programs are organized into modules.

**5. Physical organization :**

- a. The computer memory is organized as main memory and secondary memory.
- b. The main memory provides fast access but high cost.
- c. It is slow volatile. However the secondary memory is slower, cheaper and non-volatile.
- d. As a result, secondary memory is used for long term storage with large capacity.

**What do you mean by address binding?****Answer**

1. Computer memory uses both logical addresses and physical addresses.
2. Address binding allocates a physical memory location to a logical pointer by associating a physical address to a logical address, which is also known as a virtual address.
3. 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.

**Ques 4.3. Write a short note on :**

- i. Dynamic loading
- ii. Dynamic linking

**Answer****i. Dynamic loading :**

1. In dynamic loading, a routine of a program is not loaded until it is called by the program.
2. All routines are kept on disk in a re-locatable load format. The main program is loaded into memory and is executed.
3. Other routines methods or modules are loaded on request.
4. Dynamic loading makes better memory space utilization and unused routines are never loaded.
5. It is useful when large amounts of code are needed to handle infrequently occurring cases.
6. No special support from the operating system is required implemented through program design.

**ii. Dynamic linking :**

1. 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.
2. Operating system can link system level libraries to a program.
3. 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.
4. In static linking, libraries linked at compile time, so program code size becomes bigger whereas in dynamic linking libraries linked at execution time so program code size remains smaller.

**Que 4.4.** Write the short note on

- i. Overlays
- ii. Swapping

**Answer****i. Overlays :**

1. Overlays is used to keep in memory only those instructions and data that are needed at any given time.
2. When other instructions are needed, they are loaded into space that was occupied previously by instructions that are no longer needed.
3. Overlays are implemented by user; no special support needed from operating system, programming design of overlay structure is complex.

**ii. Swapping :**

1. A process must be in memory to be executed.
2. A process, however, can be swapped temporarily out of memory to a backing store and then brought back into memory for continued execution.
3. 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.
4. Sometimes swap out process also known as roll-out and swap-in process known as roll-in.

**Que 4.5.** Discuss logical versus physical address space.**Answer**

1. Logical address space is set of all logical addresses 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 memory address register.
- Compile time and load time address binding methods generate same logical and physical addresses.
- Execution time address binding scheme results in different logical and physical addresses.
- Generally logical address is referred as a virtual address.

**Que 4.6.** 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 ?

**AKTU 2013-14, Marks 05****Answer**

1. The logical address is split into two parts : the page address and then the offset.
2. The page address must be able to reference 8 (or  $2^3$ ) distinct pages. This requires 3 address bits.
3. The offset must be able to reference 1,024 (or  $2^{10}$ ) distinct words on each page. This requires 10 address bits. Therefore, 13 bits are required for the logical address.
4. The physical address is also split into two parts : the frame address and then the offset.
5. The number of frames is 32 (or  $2^5$ ), so that will require 5 bits.
6. The size of the frame is same as the size of the page, therefore this will require 10 bits.
7. Therefore, 15 bits are required for the physical address.

**Que 4.7.** Define the terms :

- i. Bare machine
- ii. Resident monitor

**Answer****i. Bare machine :**

1. This is the simplest form of memory management.
2. It used by hardware diagnostics, by system boot code, real time dedicated systems.

3. User can have complete control. Commensurably, the operating system has none.
4. No protection, no utilities, no overhead.

**ii. Resident monitor :**

1. It usually in low memory where interrupt vectors are placed.
2. It must check each memory reference against fence (fixed or variable) in hardware or register. If user generated address < fence, then illegal.
3. User program starts at fence -> fixed for duration of execution. Then user code has fence address built in. But only works for static-sized monitor.
4. If monitor can change in size, start user at high end and move back, use fence as base register that requires address binding at execution time.
5. Add base register to every generated user address. Isolate user from physical address space using logical address space.

**Ques 4.8.] Briefly discuss holes in memory partitioning.**

**Answer**

1. In the variable-partition scheme, the operating system keeps a table indicating which parts of memory are available and which are occupied.
2. Initially, all memory is available for user processes and is considered one large block of available memory known as hole.
3. When a part of memory is occupied by a process and left rest of the memory also known as hole.
4. A hole can be created when a process is completed and leaves the memory.

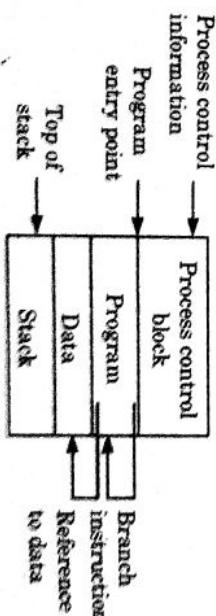


Fig. 4.8.1.

**Ques 4.9.] Discuss about fixed/static partitioning ?**

**Answer**

**Fixed partitioning:**

- Partition main memory into a set of non-overlapping memory regions called partitions.
1. Fixed partitions can be of equal or unequal sizes.
  2. Leftover space in partition space in partition, after program assignment, is called internal fragmentation.

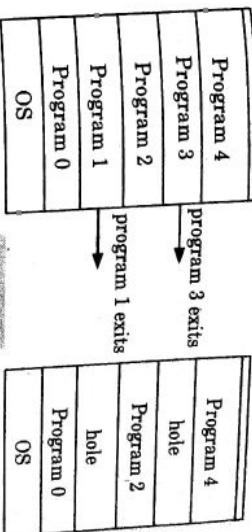


Fig. 4.9.1.

**Placement algorithm with partitions :**

**a. Equal-size partitions :**

- i. In this algorithm, if there is an available partition, a process can be loaded into that partition.
- ii. Because all partitions are of equal size, it does not matter which partition is used.
- iii. If all partitions are occupied by blocked processes, choose one process to swap out to make room for the new process.

**b. Unequal-size partitions, use of multiple queues :**

- i. This algorithm assigns each process to the smallest partition within which it will fit.
- ii. A queue exists for each partition size.
- iii. It tries to minimize internal fragmentation.
- iv. Problem with this algorithm is that some queues might be empty while some might be loaded.

**c. Unequal-size partitions, use of a single queue :**

- i. In this algorithm, when it's time to load a process into memory, the smallest available partition that will hold the process is selected.
- ii. It increases the level of multiprogramming at the expense of internal fragmentation.

**Ques 4.10.] Describe variable/dynamic partitioning.**

**Answer****Dynamic partitioning :**

1. Dynamic partition leads to external fragmentation where portion of memory outside the variable size partition are fragmented.
2. Solution to external fragmentation problem is compaction i.e., to move the free space together.
3. It is a time consuming process and complete overhead.
4. Processes must be dynamically relocatable for moving them around in memory.
5. To avoid compaction we can make intelligent assignment of process to memory so as to reduce the external fragmentation with the help of best fit, first fit and next fit. First fit is usually best and fastest.
  - a. First fit may lead to holes in the front end of memory. More the fragments in the front end more scanning for first fit.
  - b. Next fit may lead to more external fragmentation leaving the need for compaction.
  - c. Best fits leads to many smaller memory holes leading to more frequent compaction.

**Que 4.11. Describe memory allocation.****Answer**

1. One of the simplest methods for allocating memory is to divide memory into several fixed-sized partitions.
2. Each partition may contain exactly one process.
3. Thus, the degree of multiprogramming is bound by the number of partitions.
4. In this multiple-partition method, when a partition is free, a process is selected from the input queue and is loaded into the free partition. When the process terminates, the partition becomes available for another process.
5. In the variable-partition scheme, the operating system keeps a table indicating which parts of memory are available and which are occupied.
6. Initially, all memory is available for user processes and is considered one large block of available memory, a hole.
7. Memory contains a set of holes of various sizes.
8. At any given time, then, we have a list of available block sizes and an input queue.
9. The operating system can order the input queue according to a scheduling algorithm.

**Que 4.12. Discuss dynamic memory allocations techniques.**  
OR  
**Discuss first fit, best fit and worst fit memory allocation techniques.****Answer**

1. First fit, best fit, and worst fit are the most common strategies used to select a free hole from the set of available holes.
2. First fit and best fit are better than worst fit in terms of speed and storage utilization respectively.

**Dynamic memory allocation techniques :**

1. **First fit :** Allocate the first hole that is big enough. Searching can start either at the beginning of the set of holes or at the location where the previous first fit search ended. We can stop searching as soon as we find a free hole that is large enough.
2. **Best fit :** Allocate the smallest hole that is big enough. We must search the entire list, unless the list is ordered by size. This strategy produces the smallest leftover hole.
3. **Worst fit :** Allocate the largest hole. Again, we must search the entire list, unless it is sorted by size. This strategy produces the largest leftover hole, which may be more useful than the smaller leftover hole from a best fit approach.

**Memory Management**

10. Memory is allocated to processes until finally the memory requirements of the next process cannot be satisfied that is, no available block of memory or hole is large enough to hold that process.
11. The operating system can then wait until a large enough block is available, or it can skip down the input queue to see whether the smaller memory requirements of some other process can be met.
12. The memory blocks available comprise a set of holes of various sizes scattered throughout memory.
13. When a process arrives and needs memory, the system searches the set for a hole that is large enough for this process.
14. If the hole is too large, it is split into two parts. One part is allocated to the arriving process; the other is returned to the set of holes.
15. When a process terminates, it releases its block of memory, which is then placed back in the set of holes.
16. If the new hole is adjacent to other holes, these adjacent holes are merged to form one larger hole.
17. At this point, the system may need to check whether there are processes waiting for memory and whether this newly freed and recombined memory could satisfy the demands of any of these waiting processes.

**Que 4.13.** Explain the difference between internal and external fragmentation.

**AKTU 2013-14, Marks 06**

**Answer**

| S.No. | Basis for comparison | Internal fragmentation                                                                                                                                                              | External fragmentation                                                                                               |
|-------|----------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|----------------------------------------------------------------------------------------------------------------------|
| 1.    | Basic                | It occurs when fixed sized memory blocks are allocated to the processes.                                                                                                            | It occurs when variable size memory space is allocated to the processes dynamically.                                 |
| 2.    | Occurrence           | When the memory assigned to the process is slightly larger than the memory requested by the process, this creates free space in the allocated block causing internal fragmentation. | When the process is removed from the memory, it creates the free space in the memory causing external fragmentation. |
| 3.    | Solution             | The memory must be partitioned into variable sized blocks and assign the best fit block to the process.                                                                             | Compaction, paging and segmentation.                                                                                 |

**Que 4.14.** What are the different techniques to remove fragmentation in case of multiprogramming with fixed partitions and variables partitions ?

**AKTU 2015-16, Marks 10**

| Process number | Memory requested |
|----------------|------------------|
| P1             | 212 K            |
| P2             | 417 K            |
| P3             | 112 K            |
| P4             | 426 K            |

First fit :

| Memory partition | Process number | Memory requested | Status | Internal fragmentation |
|------------------|----------------|------------------|--------|------------------------|
| 100 K            | —              | —                | Free   | —                      |
| 500 K            | P1             | 212 K            | Busy   | 288 K                  |
| 200 K            | P3             | 112 K            | Busy   | 88 K                   |
| 300 K            | —              | —                | Free   | —                      |
| 600 K            | P2             | 417 K            | Busy   | 183 K                  |

Total Available : 1700 K Total used : 741 K

Process P4 has no memory partition allocated to it.

Two memory partition of size 100 K and 300 K are free which cannot be allocated to P4 because it has size more than these memory partitions.

Best fit :

| Memory partition | Process number | Memory requested | Status | Internal fragmentation |
|------------------|----------------|------------------|--------|------------------------|
| 100 K            | —              | —                | Free   | —                      |
| 500 K            | P2             | 417 K            | Busy   | 88 K                   |
| 200 K            | P3             | 112 K            | Busy   | 88 K                   |
| 300 K            | P1             | 212 K            | Busy   | 174 K                  |
| 600 K            | P4             | 426 K            | Busy   | —                      |

Total Available : 1700 K Total used : 1167 K

Only one memory partition is free but all the processes are allocated the memory partitions.

Worst fit :

| Memory Partition | Process number | Memory requested | Status | Internal fragmentation |
|------------------|----------------|------------------|--------|------------------------|
| 100 K            | —              | —                | Free   | —                      |
| 500 K            | P2             | 417 K            | Busy   | 88 K                   |
| 200 K            | —              | —                | Free   | —                      |
| 300 K            | P3             | 112 K            | Busy   | 188 K                  |
| 600 K            | P1             | 212 K            | Busy   | 388 K                  |

Total Available : 1700 K Total used : 741 K

**Fixed partitions :** Refer Q. 4.9, Page 4-7B, Unit-4.

**Variable partitions :** Refer Q. 4.10, Page 4-8B, Unit-4.

**Que 4.15.** Given memory partitions of 100 K, 500 K, 200 K, 300 K and 600 K (in order). How would each of the first fit, best fit and worst fit algorithms place processes of 212 K, 417 K, 112 K and 426 K (in order) ? Which algorithm makes the most efficient use of memory ?

Process P4 has no memory partition allocated to it. Two memory partitions of 100 K and 200 K are free but it cannot be allocated to P4 because its size is more than these memory partitions.

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

**Ques 4.16.** What do you mean by paging? How page table can be implemented?

### Answer

#### Paging :

1. Paging is a memory management scheme that permits the physical address space of a process to be non-contiguous.
2. Paging avoids external fragmentation and the need for compaction.
3. The basic method for implementing paging involves breaking physical memory into fixed-sized blocks called frames and breaking logical memory into blocks of the same size called pages.
4. When a process is to be executed, its pages are loaded into any available memory frames from their source (a file system or the backing store).
5. The backing store is divided into fixed-sized blocks that are the same size as the memory frames or clusters of multiple frames.

#### Page table implementation :

1. The hardware support for paging is illustrated in Fig. 4.16.1.

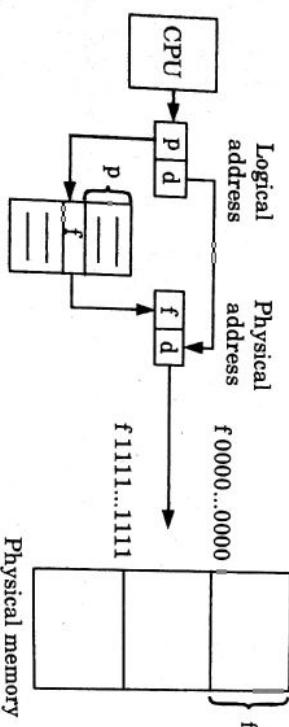


Fig. 4.16.1. Paging hardware.

Paging : Refer Q. 4.16, Page 4-13B, Unit-4.

#### Translation of logical address to physical address :

1. Every address generated by the CPU is divided into two parts : a page number ( $p$ ) and a page offset ( $d$ ).
2. The page number is used as an index into a page table. The page table contains the base address of each page in physical memory.
3. This base address is combined with the page offset to define the physical memory address that is sent to the memory unit.
4. The page model of memory is shown in Fig. 4.16.2.



Fig. 4.16.2. Paging model of logical and physical memory.

1. If the size of the logical address space is  $2^m$ , and a page size is  $2^n$  bytes, then the high-order  $m - n$  bits of a logical address designate the page number, and the  $n$  low-order bits designate the page offset.
2. Thus, the logical address is :



Where  $p$  is an index into the page table and  $d$  is the displacement within the page.

**Ques 4.17.** What is paging? Describe how logical address is translated to physical address in a paged system. Further give reasons as to why page sizes are always kept in powers of 2.

AKTU 2011-12, Marks 10

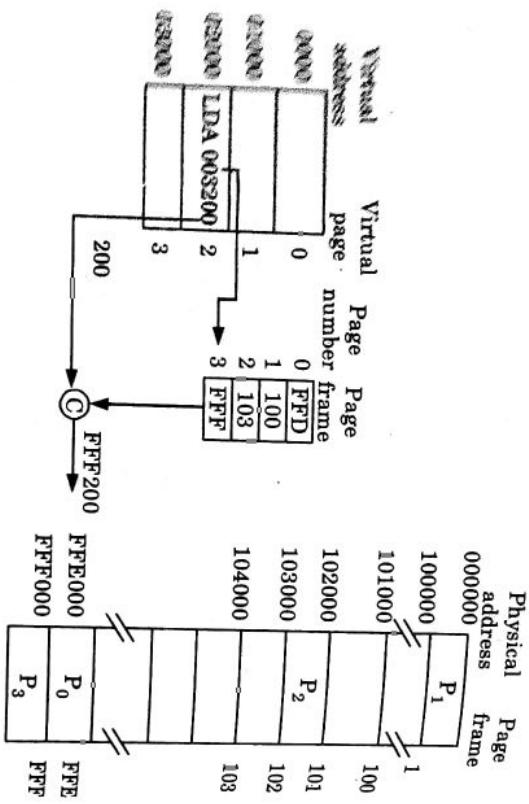
### Answer

#### Paging :

1. The logic of the address translation process in paged system is shown in the following Fig. 4.17.1.
2. For example, the virtual address is 03200H.
3. This virtual address is split by hardware into the page number and offset within that page.
4. High order 12 bit is used as page number i.e. 003H and lower order 12 bit is used for the offset i.e., (200H).

Page number is used to index the page table and to obtain the corresponding physical frame number (FFFF).

This value is then concatenated with the offset to produce the physical address (FFF200H) which is used to reference the target item in memory.



**Fig. 4.17.1.** Virtual address to physical address **paging**.

7. The operating system keeps track of the status of each page frame by means of a physical memory map that may be structured as a static table.

8. The logical address space may be the same size as the physical address space or it may be of smaller.  
9. If logical address space is small, it prevents one process from monopolizing all of the memory.  
10. If the logical address space is larger than the physical address space, all page could not be resident in physical memory.

**Reason :**

- The size of page is typically a power of 2, varying between 512 bytes and 16 MB per page, depending on the computer architecture.
- Page size is always a power of 2 because all addresses are binary and are divided into page or page frame number and offset can be determined by looking at particular bits in the address; no mathematical calculations are required.
- The physical address can be generated by concatenating the offset to the frame number.

Under what circumstances do page faults occur ? Describe the actions taken by the operating system when a page fault occurs ?

OR

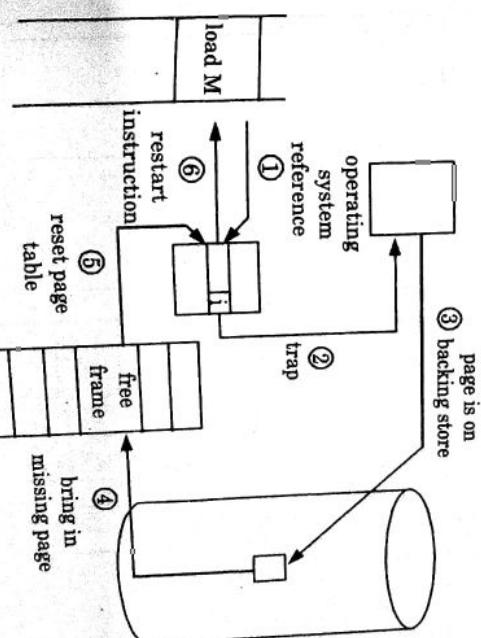
When do page faults occur ? Describe in detail the actions taken by the operating system when a page fault occurs.

**AKTU 2012-13, Marks 05**

- Answer**
- A page fault occurs when an access to a page that has not been brought into main memory takes place.
  - The operating system verifies the memory access, aborting the program if it is invalid.
  - If it is valid, a free frame is located and I/O is requested to read the needed page into the free frame.
  - Upon completion of I/O, the process table and page table are updated and the instruction is restarted.

**Steps in handling a page fault by operating system :**

- We check an internal table (usually kept with the process control block) for the process to determine whether the reference is a valid or an invalid memory access.

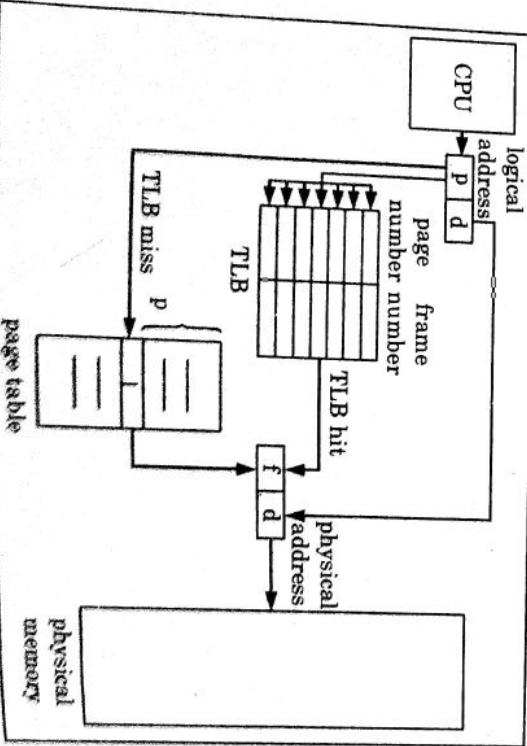


**Fig. 4.18.1.** Step in handling a page fault.

2. If the reference is invalid, we terminate the process. If it is valid, but we have not yet brought-in that page, we now page it in.
3. We find a free frame (by taking one from the free frame list).
4. We schedule a disk operation to read the desired page into the newly allocated frame.
5. When the disk read is complete, we modify the internal table kept with the process and the page table to indicate that the page is now in memory.
6. We restart the instruction that was interrupted by the trap. The process can now access the page as though it had always been in memory.

**Que 4.19** Discuss the paging hardware with TLB.**Answer**

1. Translation look aside buffer (TLB) is a special, small, fast-lookup hardware cache.
2. The TLB is associative, high-speed memory.
3. Each entry in the TLB consists of two parts: a key (or tag) and a value.
4. When the associative memory is presented with an item, the item is compared with all keys simultaneously.
5. If the item is found, the corresponding value field is returned.
6. Typically, the number of entries in a TLB is small, often numbering between 64 and 1,024.
7. The TLB is used with page tables in the following way :

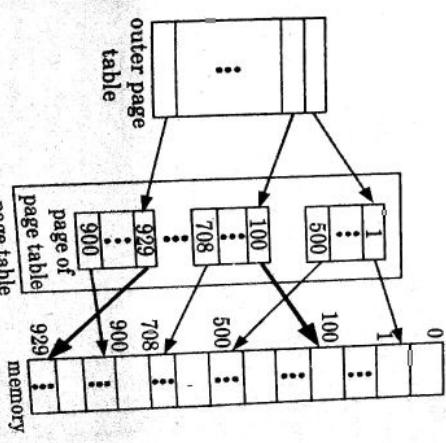
**Fig. 4.19.1.**

8. The TLB contains only a few of the page-table entries. When a logical address is generated by the CPU, its page number is presented to the TLB.
9. If the page number is found (known as 'TLB hit'), its frame number is immediately available and is used to access memory.
10. If the page number is not in the TLB (known as a 'TLB miss') a memory reference to the page table must be made.
11. When the frame number is obtained, we can use it to access memory.
12. In addition, we add the page number and frame number to the TLB, so that they will be found quickly on the next reference.
13. If the TLB is already full of entries, the operating system must select one for replacement.
14. Replacement policies range from least recently used (LRU) to random.

**Que 4.20** Discuss various structures of page table.**Answer**

Some of the most common structures of page tables are:

1. **Hierarchical page table:**
  - a. Under hierarchical page table the logical address space is divided into multiple page tables.
  - b. A simple technique used is a two-level page table.
  - c. Two-level paging as shown in Fig. 4.20.1.

**Fig. 4.20.1. Two-level paging scheme.**

- d. A logical address (on 32-bit machine with 4K page size) in two-level is divided into following two parts:

- A page number consisting of 20 bits.
- A page offset consisting of 12 bits.

e. Now since the page table is paged, the page number is further divided into following two parts :

- A 10-bit page number
- A 10-bit page offset

f. Thus, a logical address is as follows :

| page number | page offset |
|-------------|-------------|
| $p_1$       | $p_2$       |
| 10          | 10 12       |

where  $p_1$  is an index into the outer page table and  $p_2$  is the displacement within the page of the outer page table. The address-translation scheme for a two-level 32-bit paging architecture.

logical address

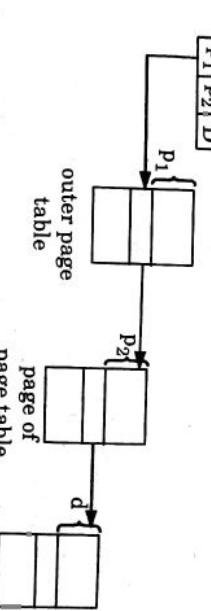


Fig. 4.20.2.

## 2. Hashed page table:

- In hashed page tables address spaces are greater than (>) 32 bits.
- The virtual page number is hashed into a page table.
- This page table contains a chain of elements hashing to the same locations.
- Virtual page numbers are compared in this chain searching for a match. If a match is found, the corresponding physical frame is extracted.

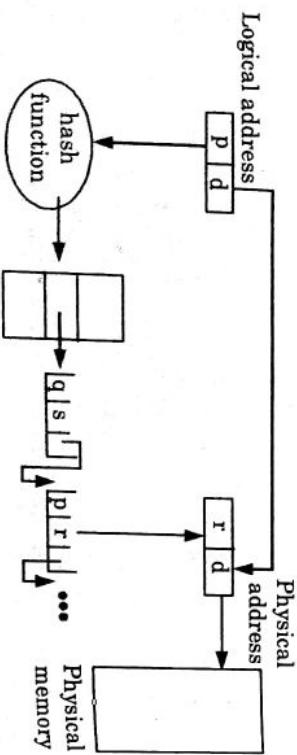


Fig. 4.20.3.

## 4-20 B (CS/IT-Sem-4)

### Inverted page table :

- In paging, page tables are quite large, one for each process and thus consume large amounts of memory.
- One solution is the use of inverted page table that has one entry for each frame of physical memory.
- Each entry contains virtual address of page stored in that real memory location with information about process that owns the page.
- The working of inverted page table is shown Fig. 4.20.4.

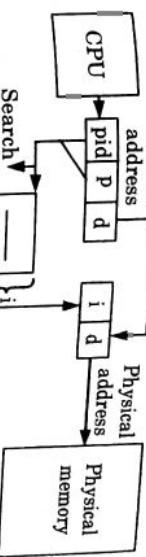


Fig. 4.20.4.

- Each logical address consists of triplet as : <process-id, page number, offset>

- When a memory references occurs, <part of logical address consisting of pair ><process-id, page number > is presented to memory table and it is then searched for a match.

- If a match is found say at entry  $i$ , then the physical address  $<i, \text{offset}>$  is generated. If no match is found, error is indicated.

**Ques 4.21.** What do you mean by segmentation? Give the hardware implementation of segmentation.

OR

Explain segmentation with diagram.

AKTU 2014-15, Marks 10

### Answer

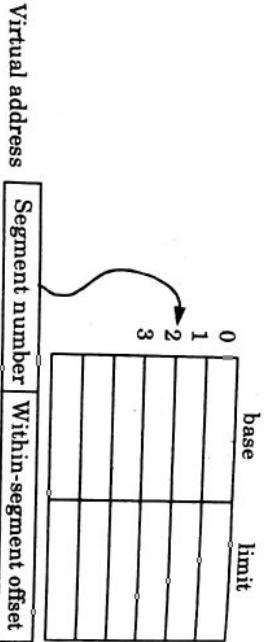
#### Segmentation :

- 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 addresses 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.

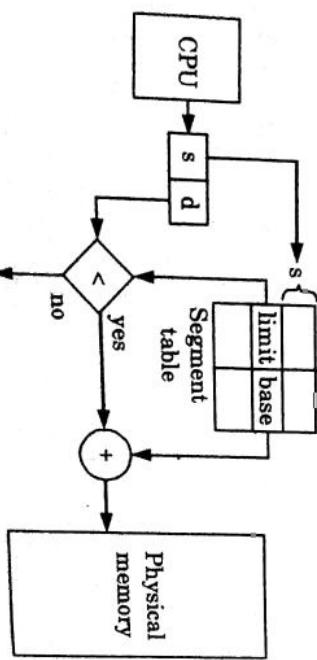
6. For simplicity of implementation, segments are numbered and referred to by a segment number, rather than by a segment name. Thus, a logical address consists of a two tuple :
- $$<\text{segment-number, offset}>$$
7. Segment number indexes a process segment table.
8. Segment table has segment base and segment limit.
9. Each entry in segment table contains the physical address where segment resides in memory, whereas segment limit specifies the length of segment.
10. Segment base contains the physical address where segment resides in memory, whereas segment limit specifies the length of segment.

**Hardware implementation of segmentation :**

- Segment table is an indexed array of base and limit pairs.
- In this segment table the virtual address are partitioned into two parts and the position of each partition defines maximum number of segments and their size.



- Other information which is also present in each index entry are :
  - Presence bit** : Whether this segment exists and has a valid base and limit.
  - Protection bit** : To allow read/write/execute, etc., for this segment.
  - Cachable bit** : To allow data in this segment to be cached or not.
- The use of segment table is shown in Fig. 4.21.1.

**Fig. 4.21.1. Segmentation hardware.**

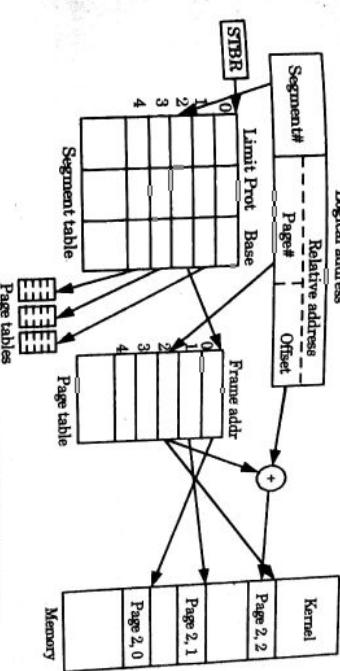
- The segment number is used as an index into segment table.
6. The offset  $d$  of logical address must be between 0 and segment limit.
7. If it is not, we trap to OS (logical addressing beyond end of segment).
- If the offset is legal it is added to segment base to produce the physical address.
8. If the offset is legal it is added to segment base to produce the physical address.

**Ques 4.22** Explain the address translation scheme in segmented-paging with the help of a diagram. OR

Discuss the paged segmentation scheme of memory management and explain how logical address is translated to physical address in such a scheme.

**AKTU 2012-13 Marks 10**

1. Fig. 4.22.1 presents the address translation mechanism under the paged segmentation scheme.

**Fig. 4.22.1. Address translation in paged segmentation.**

- Each segment is treated as a single, small linear address space within the container logical address space proper, and thus mini address space is paged.
- The address translation in the mini address space is done following the pure paging scheme.
- A page table is associated with the mini address space, which helps in the address translation task.
- As in the pure segmentation scheme, segment related information is structured into segment descriptors that are stored in a segment table.
- Unlike pure segmentation, each segment descriptor contains the physical base address of the segment's page table.

### Operating Systems

#### 4-23 B (CSIR-I-Sem-4)

- Que 4.23** How is segmentation different from paging? Discuss the address translation scheme in segmented-paging with the help of a diagram.
- OR**
- Write the difference between paging and segmentation.
- Answer**
- AKTU 2013-14, Marks 05**
- Difference between segmentation and paging :**
- | S.No. | Segmentation                                                              | Paging                                                    |
|-------|---------------------------------------------------------------------------|-----------------------------------------------------------|
| 1.    | Program is divided into variable size segments.                           | Program is divided into fixed size pages.                 |
| 2.    | User (or compiler) is responsible for dividing the program into segments. | Division into pages is performed by the operating system. |
| 3.    | Segmentation is slower than paging.                                       | Paging is faster than segmentation.                       |
| 4.    | Segmentation is visible to the user.                                      | Paging is invisible to the user.                          |
- Que 4.24** Why are segmentation and paging sometimes combined into one scheme?
- Answer**
- AKTU 2014-15, Marks 05**
- Address translation in segmented-paging :** Refer Q. 4.22, Page 4-22B, Unit-4.
- Que 4.24** On a system using paging and segmentation the virtual address space consists of upto 16 segments where each segment can be upto 216 bytes long. The hardware pages each segment into 512 byte pages. How many bits in the virtual address specify the following ?
- i. Segment number
  - ii. Page number
  - iii. Offset within page
  - iv. Entire virtual address
- Answer**
- i. Segment number : Virtual address space consists of upto 16 segments So,  $16 = 2^4$   
 $\therefore$  4 bits are needed to specify segment number.
- ii. Page number : Hardware pages each segment into 512 bytes pages.  
 $512 = 2^9$  byte pages  
Size of segment is  $2^{16}$  bytes.  
 $2^{16} / 2^9 = 2^{16-9} = 2^7 = 7$  pages  
 $\therefore$  7 bits are required to specify the page number.
- iii. Offset within page : For  $2^9$  byte page, 9 bits are needed.
- iv. Entire virtual address : Segment number + Page number + offset  
 $= 4 + 7 + 9 = 20$

### Memory Management

#### 4-24 B (CSIR-I-Sem-4)

|    |                                                                   |                                                                      |
|----|-------------------------------------------------------------------|----------------------------------------------------------------------|
| 5. | Segmentation eliminates internal fragmentation.                   | Paging suffers from internal fragmentation.                          |
| 6. | Segmentation suffers from external fragmentation.                 | There is no external fragmentation.                                  |
| 7. | Processor uses page number, offset to calculate absolute address. | Processor uses segment number, offset to calculate absolute address. |
| 8. | OS maintain a list of free holes in main memory.                  | OS must maintain a free frame list.                                  |

**ANSWER**

1. Segmentation and paging are often combined in order to improve upon each other.
2. Segmented paging is helpful when the page table becomes very large. A large contiguous section of the page table that is unused can be collapsed into a single-segment table entry with a page-table address of zero.
3. Paged segmentation handles the case of having very long segments that require a lot of time for allocation.
4. By paging the segments, we reduce wasted memory due to external fragmentation as well as simplify the allocation.

**PART-2**

*Virtual Memory Concepts, Demand Paging, Performance of Demand Paging, Page Replacement Algorithms, Thrashing, Cache Memory Organization, Locality of Reference.*

**CONCEPT OUTLINE : PART-2**

- Virtual memory is a technique that allows the execution of processes which are not completely available in memory.
- The process of loading the page into memory on demand (whenever the page fault occurs) is known as demand paging.
- Page replacement algorithms.
  - i. FIFO
  - ii. LRU
  - iii. Optimal
- Locality of reference is a term for the phenomenon in which the same values or related storage locations are frequently accessed, depending on the memory access pattern.

**Ques 4.27.** Explain the process of demand paging.

**Answer**

The process of loading the page into memory on demand (whenever page fault occurs) is known as demand paging. The process includes the following steps:

1. If CPU tries to refer a page that is currently not available in the main memory, it generates an interrupt indicating memory access fault.
2. The OS puts the interrupted process in a blocking state. For the execution to proceed the OS must bring the required page into the memory.

**Answer****Virtual memory :**

1. Virtual memory is a memory management capability of an OS that uses hardware and software to allow a computer to compensate for physical memory shortages by temporarily transferring data from Random Access Memory (RAM) to disk storage.
2. Virtual address space is increased using active memory in RAM and inactive memory in Hard Disk Drives (HDDs) to form contiguous addresses that hold both the application and its data.
3. A system using virtual memory can load larger programs or multiple programs running at the same time, allowing each one to operate as if it has infinite memory and without having to purchase more RAM.
4. Virtual memory is a facility that allows program to address memory from local point of view, without regard to the amount of main memory.
5. Virtual memory is used to give programmers the illusion that they have a very large memory at their disposal, even though the computer actually has a relatively small main memory.
6. A virtual memory system provides a mechanism for translating program generated addresses into correct main memory locations.
7. This is done dynamically, while programs are being executed in the CPU.

**Advantages of virtual memory :**

1. Allows us to fit many large programs into a relatively small RAM.
2. Only part of a program needs to be loaded into memory.
3. Memory can be used efficiently because a section of program loaded only when it is needed in CPU.
4. Virtual memory allows sharing of code and data, unlimited amounts of multi-programming.

**Ques 4.27.** Explain the process of demand paging.

**Answer**

The process of loading the page into memory on demand (whenever page fault occurs) is known as demand paging. The process includes the following steps:

1. If CPU tries to refer a page that is currently not available in the main memory, it generates an interrupt indicating memory access fault.
2. The OS puts the interrupted process in a blocking state. For the execution to proceed the OS must bring the required page into the memory.

**Ques 4.28.** What is virtual memory ?

OR

Explain the concept of virtual memory and also discuss its advantages.

**Questions-Answers****Long Answer Type and Medium Answer Type Questions**

3. The OS will search for the required page in the logical address space.  
 4. The required page will be brought from logical address space to physical address space. The page replacement algorithms are used for the decision making of replacing the page in physical address space.

5. The page table will be updated accordingly.  
 6. The signal will be sent to the CPU to continue the program execution and it will place the process back into ready state.

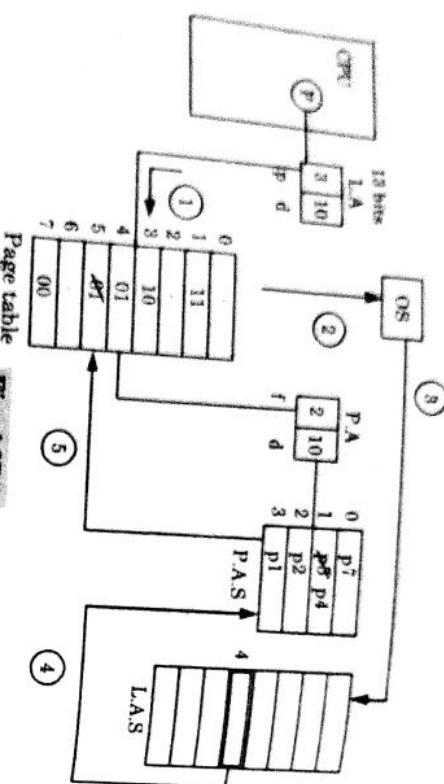


Fig. 4.27.1.

**Ques 4.28.** What are the advantages and disadvantages of demand paging?

**Answer**

**Advantages of demand paging:**

1. Large virtual memory.
2. More efficient use of memory.
3. Unconstrained multiprogramming. There is no limit on degree of multiprogramming.

**Demand paging:**

1. Number of tables and amount of processor overhead for handling page interrupts are greater than in the case of the simple paged management techniques.
2. Due to the lack of an explicit constraint on a job's address space size of amount of multiprogramming, it is necessary to develop approaches to prevent thrashing.

**Ques 4.29.**

Discuss performance of demand paging.

**Answer**

Demand paging can have a significant effect on the performance of a computer system.

- Let us calculate effective access time for a demand paged memory. Let  $p$  be the probability of a page fault ( $0 \leq p \leq 1$ ).  
 Effective access time =  $(1 - p) * m_a + p * \text{page fault time}$   
 where,

$$m_a = \text{memory access time}$$

1. Effective access time is directly proportional to the page fault rate.  
 2. It is important to keep the page-fault rate low in a demand paging system. Otherwise, the effective access time increases, slowing process execution dramatically.

**Ques 4.30.** Consider a demand paged system. Page tables are held in registers. It takes 8 milliseconds to service a page fault if an empty page is available or the replaced page is not modified, and 20 milliseconds if the replaced page is modified. Memory access time is 100 nanoseconds. Assume that the page to be replaced is modified 70 percent of the time. What is the maximum acceptable page fault rate for an effective access time of not more than 200 nanoseconds?

**AKTU 2012-13, Marks: 0**

**Answer**

From the given conditions, we get

$$0.2 \mu\text{s} = (1 - p) \times 0.1 \mu\text{s} + (0.3p) \times 8 \text{ msec} + (0.7P) \times 20 \text{ msec}$$

$$\text{or } 1 = -p + 2400P + 14000p$$

$$[\because 100 \text{ ns} = 0.1 \mu\text{sec} \text{ and } 200 \text{ ns} = 0.2 \mu\text{sec}]$$

$$\text{or } 1 = 16399p$$

$$\text{or } p = 0.00006$$

**Ques 4.31.** Consider a demand paged system. Page tables are stored in main memory which has an access time of 100 nanoseconds. The TLB can hold 8 page table entries and has an access time of 10 nanoseconds. During the execution of a process, it is found that 80% of the time, a required page table entry exists in TLB and only 2% of the references cause page faults. The average time to service page fault is 2 milliseconds. Compute the effective memory access time.

**AKTU 2011-12, Marks: 10**

**Ans:-**

TLB lookup takes 10 nanoseconds  
Effective access time = Page fault time  $\times p + m_a \times (1 - p)$

$$\begin{aligned} &\approx (10 \times 10^{-9} + 100 \times 10^{-9}) \times 0.8 \\ &+ (10 \times 10^{-9} + 100 \times 10^{-9}) \\ &+ 2 \times 10^{-9} \times 0.2 \\ &= (88 + 4000022) \times 10^{-9} \\ &= 400110 \times 10^{-9} \\ &= 400.110 \text{ nsec} \end{aligned}$$

**Ques:-**

**Discuss page replacement algorithms with suitable examples.**

**Ans:-**

The basic page replacement algorithms are as follows :

- First In First Out (FIFO) algorithm :

- A FIFO replacement algorithm is a First In First Out

- 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. When a page is brought into memory, we insert it at the tail of the queue.

For example :

reference string

|   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 7 | 0 | 1 | 2 | 0 | 3 | 0 | 4 | 2 | 3 | 0 | 3 | 2 | 1 | 2 | 0 | 1 | 7 | 0 | 1 |
| 7 | 7 | 7 | 2 | 2 | 2 | 4 | 4 | 4 | 0 | 0 | 0 | 2 | 2 | 2 | 2 | 2 | 7 | 0 | 1 |
| 0 | 0 | 0 | 0 | 2 | 3 | 3 | 3 | 2 | 2 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 1 |
| 1 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 2 | 2 | 1 |
| 0 | 0 | 0 | 0 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 2 | 2 | 1 |

page frames

| page frames | 7 | 0 | 1 | 2 | 0 | 3 | 0 | 4 | 2 | 3 | 0 | 3 | 2 | 1 | 2 | 0 | 1 | 7 | 0 | 1 |
|-------------|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 7           | 7 | 7 | 2 | 2 | 2 | 4 | 4 | 4 | 0 | 0 | 0 | 2 | 2 | 2 | 2 | 2 | 7 | 0 | 1 |   |
| 0           | 0 | 0 | 0 | 2 | 3 | 3 | 3 | 2 | 2 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 1 |   |
| 1           | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 2 | 2 | 1 |   |
| 0           | 0 | 0 | 0 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 2 | 2 | 1 |   |

Reference string, our three frames are initially empty. The first three references (7, 0, 1) cause page faults and are brought into these empty frames.

- The next reference (2) replaces page 7, because page 7 was brought in first. Since 0 is the next reference and 0 is already in memory, we have no fault for this reference.

- The first reference to 3 results in replacement of page 0, since it is now first in line.

**Ques:-**

**Memory Management**

Because of this replacement, the next reference, to 0, will fault. Every time a fault occurs, we show which pages are in our three frames.

**Optimal 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 and will never suffer from Belady's anomaly.
- This algorithm is impossible to implement because it would require the operating system to have perfect knowledge of future events.

for example :

reference string

|   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 7 | 0 | 1 | 2 | 0 | 3 | 0 | 4 | 2 | 3 | 0 | 3 | 2 | 1 | 2 | 0 | 1 | 7 | 0 | 1 |
| 7 | 7 | 7 | 2 | 2 | 2 | 4 | 4 | 4 | 0 | 0 | 0 | 2 | 2 | 2 | 2 | 2 | 7 | 0 | 1 |
| 0 | 0 | 0 | 0 | 2 | 3 | 3 | 3 | 2 | 2 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 1 |
| 1 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 2 | 2 | 1 |
| 0 | 0 | 0 | 0 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 2 | 2 | 1 |

page frames

- The optimal page replacement algorithm would yield nine page faults.
- The first three references cause faults that fill the three empty frames until reference 18, whereas page 0 will be used at 5, and page 1 at 14.
- The reference to page 3 replaces page 1, as page 1 will be the last of the three pages in memory to be referenced again.

- With only nine page faults, optimal replacement is much better than a FIFO algorithm, which results in fifteen faults.

**3. LRU page replacement:**

- If the optimal algorithm is not feasible, perhaps an approximation of the optimal algorithm is possible.
- The key distinction between the FIFO and OPT algorithms (other than looking backward versus forward in time) is that the FIFO algorithm uses the time when a page was brought into memory, whereas the OPT algorithm uses the time when a page is to be used.
- If we use the recent past as an approximation of the near future, then we can replace the page that has not been used for the longest period of time.

- This approach is the least recently used (LRU) algorithm.

For example :

| reference string |   |   |   |   |   |   |
|------------------|---|---|---|---|---|---|
| 7                | 0 | 1 | 2 | 0 | 3 | 0 |
| 4                | 2 | 3 | 0 | 3 | 2 | 1 |
| 2                | 1 | 0 | 0 | 0 | 0 | 1 |
| 3                | 2 | 2 | 3 | 3 | 0 | 0 |
| 2                | 1 | 0 | 1 | 1 | 0 | 1 |
| 0                | 1 | 1 | 0 | 0 | 1 | 0 |

i. The LRU algorithm produces twelve faults.

ii. When the reference to page 4 occurs, however, LRU replacement sees that, of the three frames in memory, page 2 was used least recently.

iii. Thus, the LRU algorithm replaces page 2, not knowing that page 2 is about to be used.

iv. When it then faults for page 2, the LRU algorithm replaces page 3, since it is now the least recently used of the three pages in memory.

#### Ques 4.33.

Consider the following reference string for:

i. FIFO  
ii. LRU page replacement algorithm ?

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

#### Answer

##### i. FIFO

3 Frame

4 Frame

|   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 2 | 3 | 4 | 2 | 1 | 5 | 6 | 2 | 1 | 2 | 3 | 7 | 6 | 3 | 2 | 1 | 2 | 3 | 6 |
| 4 | 4 | 6 | 6 | 6 | 3 | 3 | 3 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 1 | 1 | 1 | 1 | 1 |
| 3 | 5 | 5 | 5 | 5 | 1 | 1 | 1 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 3 | 3 | 3 | 3 | 3 |
| 3 | 5 | 5 | 5 | 5 | 1 | 1 | 1 | 6 | 6 | 6 | 6 | 6 | 6 | 6 | 4 | 4 | 4 | 4 | 4 |

|   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 2 | 3 | 4 | 2 | 1 | 5 | 6 | 2 | 1 | 2 | 3 | 7 | 6 | 3 | 2 | 1 | 2 | 3 | 6 |
| 5 | 5 | 5 | 5 | 5 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 4 | 4 | 4 | 4 | 4 |
| 2 | 6 | 6 | 6 | 6 | 7 | 7 | 7 | 7 | 7 | 7 | 7 | 7 | 7 | 7 | 5 | 5 | 5 | 5 | 5 |
| 3 | 3 | 3 | 3 | 3 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 6 | 6 | 6 | 6 | 6 |

##### ii. LRU

3 Frame

4 Frame

|   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |             |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|-------------|
| 1 | 2 | 3 | 4 | 2 | 1 | 5 | 6 | 2 | 1 | 2 | 3 | 7 | 6 | 3 | 2 | 1 | 2 | 3 | 6           |
| 1 | 1 | 4 | 4 | 5 | 5 | 1 | 7 | 7 | 7 | 2 | 2 | 2 | 2 | 2 | 1 | 2 | 3 | 6 | Page Faults |
| 2 | 2 | 2 | 2 | 2 | 6 | 6 | 6 | 6 | 6 | 6 | 6 | 6 | 6 | 6 | 3 | 3 | 3 | 3 | 3           |
| 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 2 | 2 | 2 | 2 | 2           |

#### 4.31 B (CSE71-Sem-4)

| Memory Management                       |  |  |  |  |  |  |  |  |             |
|-----------------------------------------|--|--|--|--|--|--|--|--|-------------|
| 4 Frame                                 |  |  |  |  |  |  |  |  | 1           |
| 1 2 3 4 2 1 3 6 2 1 2 3 7 6 3 2 1 2 3 6 |  |  |  |  |  |  |  |  | Page Faults |
| 1 2 3 4 2 1 3 6 2 1 2 3 7 6 3 2 1 2 3 6 |  |  |  |  |  |  |  |  | 1 0         |
| 1 2 3 4 2 1 3 6 2 1 2 3 7 6 3 2 1 2 3 6 |  |  |  |  |  |  |  |  | 1 0         |

#### Ques 4.34.

How many page faults would occur for the following reference string for four page frames using LRU and FIFO algorithms ?  
1,2,3,4,5,6,3,4,1,6,7,8,7,8,9,7,8,9,5,4,5,4,2

#### AKTU 2014-16, Marks 10

#### Answer

i. LRU

|   |   |   |   |   |       |   |   |   |   |     |   |   |   |   |   |   |   |   |   |
|---|---|---|---|---|-------|---|---|---|---|-----|---|---|---|---|---|---|---|---|---|
| 1 | 2 | 3 | 4 | 5 | 5,3,4 | 1 | 6 | 7 | 8 | 7,8 | 9 | 7 | 8 | 9 |   |   |   |   |   |
| 1 | 2 | 2 | 2 | 2 | 2     | 2 | 2 | 2 | 2 | 2   | 2 | 2 | 2 | 2 | 3 | 3 | 3 | 3 | 3 |
| 3 | 3 | 3 | 3 | 3 | 3     | 3 | 3 | 3 | 3 | 3   | 3 | 3 | 3 | 3 | 4 | 4 | 4 | 4 | 4 |
| 3 | 3 | 3 | 3 | 3 | 3     | 3 | 3 | 3 | 3 | 3   | 3 | 3 | 3 | 3 | 4 | 4 | 4 | 4 | 4 |

Total number of page fault will be 13.

ii. FIFO

|   |   |   |   |   |       |   |   |   |   |     |   |   |   |   |   |   |   |   |   |
|---|---|---|---|---|-------|---|---|---|---|-----|---|---|---|---|---|---|---|---|---|
| 1 | 2 | 3 | 4 | 5 | 5,3,4 | 1 | 6 | 7 | 8 | 7,8 | 9 | 7 | 8 | 9 |   |   |   |   |   |
| 1 | 2 | 2 | 2 | 2 | 2     | 2 | 2 | 2 | 2 | 2   | 2 | 2 | 2 | 2 | 3 | 3 | 3 | 3 | 3 |
| 3 | 3 | 3 | 3 | 3 | 3     | 3 | 3 | 3 | 3 | 3   | 3 | 3 | 3 | 3 | 4 | 4 | 4 | 4 | 4 |
| 3 | 3 | 3 | 3 | 3 | 3     | 3 | 3 | 3 | 3 | 3   | 3 | 3 | 3 | 3 | 4 | 4 | 4 | 4 | 4 |

Total number of page fault will be 13.

Ques 4.35. Consider the following reference string 1234215621

- i. First in First out ?  
ii. LRU page replacement algorithm ?  
Assuming three and four frames (initially empty) in each case.

**ARHU 2011-12, Marks 10**

i. First In first out (page frame = 3):

| Page  | 1  | 2  | 3  | 4  | 5  | 6  | 2  | 1  | 2  | 3  | 7  | 6  | 3  | 2  | 1  | 2  | 3  | 6 |
|-------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|---|
| Frame | 0  | 3  | 1  | 4  | 4  | 4  | 6  | 6  | 6  | 3  | 3  | 3  | 2  | 2  | 2  | 2  | 6  |   |
| 1     | -  | 2  | 2  | 2  | 1  | 1  | 2  | 2  | 2  | 7  | 7  | 7  | 1  | 1  | 1  | 1  | 1  |   |
| 2     | -  | 3  | 3  | 3  | 3  | 5  | 5  | 5  | 1  | 1  | 1  | 6  | 6  | 6  | 3  | 3  | 3  |   |
| Page  | PP | PF |   |
| Fault | PF |   |

Number of page faults = 16  
First In First out (page frame = 4):

| Page  | 1  | 2  | 3  | 4  | 2  | 1  | 5  | 6  | 2  | 1  | 2  | 3  | 7  | 6  | 3  | 2  | 1  | 2  | 3  | 6  |
|-------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
| Frame | 0  | 1  | 1  | 1  | 1  | 1  | 1  | 5  | 5  | 5  | 5  | 5  | 3  | 3  | 3  | 3  | 1  | 1  | 1  | 1  |
| 1     | -  | 2  | 2  | 2  | 2  | 2  | 2  | 6  | 6  | 6  | 6  | 6  | 6  | 7  | 7  | 7  | 7  | 3  | 3  | 3  |
| 2     | -  | 3  | 3  | 3  | 3  | 3  | 3  | 2  | 2  | 2  | 2  | 2  | 6  | 6  | 6  | 6  | 6  | 6  | 6  | 6  |
| 3     | -  | -  | 4  | 4  | 4  | 4  | 4  | 4  | 1  | 1  | 1  | 1  | 1  | 1  | 2  | 2  | 2  | 2  | 2  | 2  |
| Page  | PP | PF |
| Fault | PF |

Number of page faults = 16

Number of page faults = 10  
First In First out (page frame = 4):

| Page  | 1  | 2  | 3  | 4  | 2  | 1  | 5  | 4  | 3  | 1  | 4  | 3  | 7  | 6  | 5  | 4  | 3  | 2  | 1  | 4  |
|-------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
| Frame | 7  | 7  | 7  | 2  | 2  | 4  | 4  | 4  | 0  | 1  | 1  | 1  | 1  | 4  | 4  | 4  | 0  | 3  | 3  | 0  |
| 1     | -  | 2  | 2  | 2  | 2  | 2  | 6  | 6  | 6  | 3  | 3  | 3  | 3  | 3  | 3  | 3  | 0  | 0  | 0  | 1  |
| 2     | -  | 3  | 3  | 3  | 3  | 3  | 2  | 2  | 2  | 6  | 6  | 6  | 6  | 6  | 6  | 6  | 6  | 6  | 6  | 6  |
| Page  | PP | PP | PP | PP | PP | PP | PF |
| Fault | PF |

Discuss the LRU page replacement policy. Prove that this policy do not suffer from Belady's anomaly. How many page faults would occur with LRU policy for the following reference string :

1, 2, 3, 4, 1, 5, 3, 4, 1, 6, 7, 8, 7, 8, 9, 8, 7, 8, 9, 5, 4, 5, 4, 2  
Assume that there are four frames which are initially empty.

**ARHU 2011-12, Marks 10**

#### Answer

##### LRU page replacement:

- If the optimal algorithm is not feasible, perhaps an approximation of the optimal algorithm is possible.
- The key distinction between the FIFO and OPT algorithms is that the FIFO algorithm uses the time when a page was brought into memory, whereas the OPT algorithm uses the time when a page is to be used.
- If we use the recent past as an approximation of the near future, then we can replace the page that has not been used for the longest period of time.
- This approach is the least recently used (LRU) algorithm.

Consider the reference string :

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

|   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 7 | 7 | 7 | 2 | 2 | 4 | 4 | 4 | 0 | 1 | 1 | 1 | 1 | 4 | 4 | 4 | 0 | 3 | 3 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 2 | 2 | 2 | 2 | 2 | 2 | 1 |

page frames

5. The LRU algorithm produces twelve faults.

6. The LRU policy is often used as a page replacement algorithm and is considered to be good.

7. The major problem is how to implement LRU replacement.

8. An LRU page replacement algorithm may require substantial hardware assistance.

Number of page faults = 14

- The problem is to determine an order for the frames defined by the time of last use.

**Proof:**

1. LRU replacement does not suffer from Belady's anomaly.
2. It belongs to a class of page replacement algorithms that can never exhibit Belady's anomaly.
3. A stack algorithm is an algorithm for which it can be shown that the set of pages in memory for  $n$  frames is always a subset of the set of pages in memory with  $n+1$  frames.
4. For LRU replacement, the set of pages in memory would be the  $n$  most recently referenced pages.

5. If the number of frames is increased, these  $n$  pages will still be the most recently referenced and so will still be in memory.
- Numerical:**
1. 2, 3, 3, 4, 1, 5, 3, 4, 1, 6, 7, 8, 7, 8, 9, 5, 4, 5, 4, 2

|    |    |    |    |    |    |    |    |    |    |    |    |    |
|----|----|----|----|----|----|----|----|----|----|----|----|----|
| 1  | 2  | 3  | 3  | 4  | 1  | 5  | 3  | 4  | 1  | 6  | 7  | 8  |
| 1  | 2  | 2  | 1  | 1  | 1  | 1  | 1  | 1  | 1  | 1  | 1  | 1  |
| 2  | 2  | 2  | 2  | 2  | 2  | 2  | 2  | 2  | 2  | 2  | 2  | 2  |
| 3  | 3  | 3  | 3  | 3  | 3  | 3  | 3  | 3  | 3  | 3  | 3  | 3  |
| 4  | 4  | 4  | 4  | 4  | 4  | 4  | 4  | 4  | 4  | 4  | 4  | 4  |
| 5  | 5  | 5  | 5  | 5  | 5  | 5  | 5  | 5  | 5  | 5  | 5  | 5  |
| 6  | 6  | 6  | 6  | 6  | 6  | 6  | 6  | 6  | 6  | 6  | 6  | 6  |
| 7  | 7  | 7  | 7  | 7  | 7  | 7  | 7  | 7  | 7  | 7  | 7  | 7  |
| 8  | 8  | 8  | 8  | 8  | 8  | 8  | 8  | 8  | 8  | 8  | 8  | 8  |
| 9  | 9  | 9  | 9  | 9  | 9  | 9  | 9  | 9  | 9  | 9  | 9  | 9  |
| 10 | 10 | 10 | 10 | 10 | 10 | 10 | 10 | 10 | 10 | 10 | 10 | 10 |

Total number of page faults = 12.

**Ques 4.37. What is thrashing? State the cause of thrashing and discuss its solution.**

**AKTU 2015-16, Marks 10****Answer****Thrashing:**

1. Thrashing occurs when a system spends more time in processing page faults rather than executing transactions.
2. While processing page faults it is necessary to be in order to appreciate the benefits of virtual memory, thrashing has a negative effect on the system.
3. As the page fault rate increases, more transactions need processing from the paging device.
4. The queue at the paging device increases, resulting in increased service time for a page fault.

**4.38 B (CS/IT-Sem-4)****Memory Management****Fig. 4.37.1.**

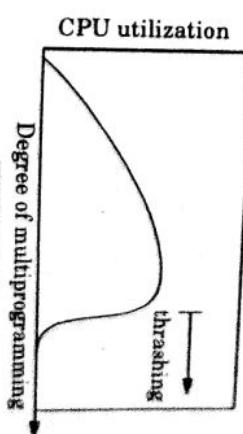
- While the transactions in the system are waiting for the paging device, CPU utilization, system throughput and system response time decrease, resulting in below optimal performance of a system.

Thrashing becomes a greater threat as the degree of multiprogramming of the system increases.

Therefore, to increase CPU utilization multiprogramming should be reduced.

**Cause of thrashing:**

1. A system thrashes if physical memory is too small to hold the working sets of all the processes running.
2. The upshot of thrashing is that pages always spend their time waiting for the backing store to fetch their pages.

**Fig. 4.37.2.**

Thrashing is a discrete phenomenon.

4. Usually, there is a phase transition from no thrashing to thrashing.
5. Scheduler brought in new process whenever CPU utilization goes down.
6. Eventually the size of the working sets becomes larger than physical memory, and processes start to page.
7. The CPU utilization drops even further, so more processes come in, and the system starts to thrash.
8. Thrashing is busy swapping pages in and out or doing more paging than executing.
9. A process is thrashing if it is spending more time in paging than executing.

**Solution / Elimination :** The data cache pre-fetching and single assignment data move principle enable elimination of cache thrashing.

**Ques 4.38.** What is the cause of thrashing? Once it detects thrashing, what can system do to eliminate this problem?

**What is the cause of thrashing? OR What steps are taken by the system to eliminate this problem?**

**AKTU 2016-17, Marks 10**

**Causes of thrashing :** Refer Q. 4.37, Page 4-35B, Unit-4.  
**Detection of thrashing :** The system can detect thrashing by evaluation the level of CPU utilization as compared to the level of multiprogramming.

**Elimination of thrashing :** Refer Q. 4.37, Page 4-35B, Unit-4.

**Ques 4.39.** Write short note on cache memory.

**Answer**

- Cache memory is a small-sized type of volatile computer memory that provides high-speed data access to a processor and stores frequently used computer programs, applications and data.
- It stores and retains data only until a computer is powered up.
- Cache memory provides faster data storage and access by storing an instance of programs and data routinely accessed by the processor.
- Thus, when a processor requests data that already has an instance in the cache memory, it does not need to go to the main memory or the hard disk to fetch the data.
- Cache memory can be primary or secondary cache memory, where primary cache memory is directly integrated or closest to the processor. In addition to hardware-based cache, cache memory also can be a disk cache, where a reserved portion on a disk stores and provide access to frequently accessed data/applications from the disk.

**Ques 4.40.** Discuss the design issues in cache design.

**Answer**

The primary elements having strong influence on cache design are:

i. **Cache size :** There are two important factor in deciding cache size at the time of its design.

- Size should be small enough so that its cost is very close to the main memory.

Size should be large enough so that most of the memory references should be available in the cache so that average access time will be close to cache alone.

#### Block size :

- Block size of cache is an important factor in cache performances.
- Larger block size could store the desired word as well as some adjacent word also. As a result hit ratio will increase as well as some A small block size result in frequent replacement automatically after it is fetched increasing the overhead of data stored.
- As the block size increases from smaller to larger, the hit ratio will increase first and on increasing the block size further the hit ratio began to decrease.

**Associativity :** It is a basic tradeoff between parallel searching versus constraints on which addresses can be stored.

**Write strategy :** Its main concern is related to when do we write cache contents to main memory.

**Ques 4.41.** Describe the different mapping techniques for cache memory.

**OR**

**What is meant by cache mapping? What are different types of mapping? Discuss different mapping techniques with examples.**

**Answer**

#### Cache mapping :

- Cache mapping is the method by which the contents of main memory are brought into the cache and referenced by the CPU. The mapping method used directly affects the performance of the entire computer system.
- Mapping is a process to discuss possible methods for specifying where memory blocks are placed in the cache. Mapping function dictates how the cache is organized.

#### Types of mapping :

##### 1. Direct mapping :

- The direct mapping technique is simple and inexpensive to implement.
- When the CPU wants to access data from memory, it places an address. The index field of CPU address is used to access address.
- The tag field of CPU address is compared with the associated tag in the word read from the cache.
- If the tag-bits of CPU address are matched with the tag-bits of cache, then there is a hit and the required data word is read from cache.

3. If there is no match, then there is a miss and the required data word is entered in main memory. It is then transferred from main memory to cache memory with the new tag.

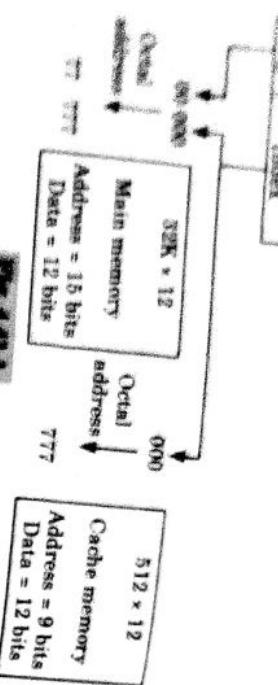


Fig. 4.1.1

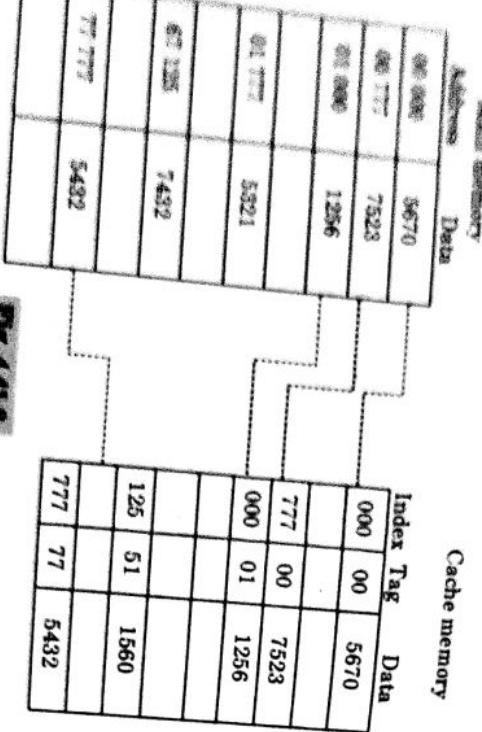


Fig. 4.1.2

- i. Associative mapping:**
1. An associative mapping uses an associative memory.
  2. This memory is being accessed using its contents.
  3. Each line of cache memory will accommodate the address (main memory) and the contents of that address from the main memory.
  4. That is why this memory is also called Content Addressable Memory (CAM). It allows each block of main memory to be stored in the cache.

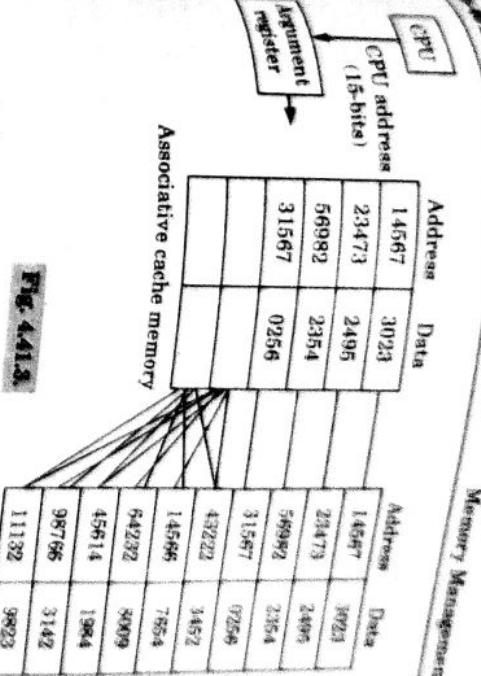


Fig. 4.1.3

#### ii. Set associative mapping:

1. That is the easy control of the direct mapping cache and the more flexible mapping of the fully associative cache.
2. In set associative mapping, each cache location can have more than one pair of tag + data items.
3. That is more than one pair of tag and data are residing at the same location of cache memory. If one cache location is holding two pair of tag + data items, that is called 2-way set associative mapping.

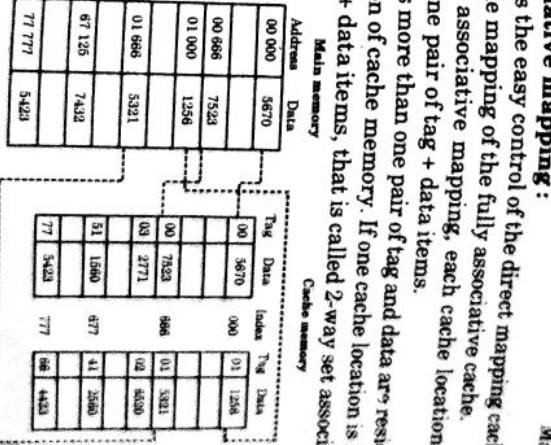


Fig. 4.1.4

**Ques. Write short notes on locality of reference?**

Locality of reference is a term for the phenomenon in which the same values or related storage locations are frequently accessed, depending on the memory access pattern.

There are two basic types of reference locality:

1. Temporal locality.

2. Spatial locality.

**a. Temporal locality :**

1. The temporal locality means that a recently executed instruction is likely to be executed again very soon.
2. The temporal aspect of the locality reference suggests that whenever information of instruction and data is first needed, this information should be brought into cache where it will hope fully remain until it is needed again.

**b. Spatial locality :**

1. The spatial locality means that instructions stored nearby to the recently executed instructions are also likely to be executed soon.
2. The spatial aspect suggests that instead of bringing just one item from the main memory to the cache, it is wise to bring several items that reside at adjacent addresses as well.

**VERY IMPORTANT QUESTIONS**

*Following questions are very important. These questions may be asked in your SESSIONALS as well as UNIVERSITY EXAMINATION.*



**Q.5. What do you mean by paging and segmentation ?**  
Refer Q. 4.16 and 4.21.

**Q.6. Differentiate between paging and segmentation.**  
Refer Q. 4.23.

**Q.7. Describe demand paging. What are its advantages and disadvantages ?**  
Refer Q. 4.27 and Q. 4.28.

**Q.8. Discuss the page replacement algorithms.**  
Refer Q. 4.32.

**Q.9. What is thrashing ? State the cause of thrashing and its solution.**  
Refer Q. 4.37.

- Q.1. Describe memory management and its requirement.**  
Refer Q. 4.1.
- Q.2. Write a short note on**
  - i. Dynamic loading and dynamic linking
  - ii. Overlaps and swapping
  - iii. Base machine and resident monitor
- Q.3. Explain the following terms :**
  - i. Fixed / static partitioning
  - ii. Variable / dynamic partitioning
  - iii. Refer Q. 4.9.
  - iv. Refer Q. 4.10.
  - v. Refer Q. 4.13.
- Q.4. Differentiate between internal fragmentation and external fragmentation.**  
Refer Q. 4.13.

# 5

## I/O Management and Disk Scheduling

**UNIT**

### I/O Management and Disk Scheduling

#### CONCEPT OUTLINE : PART-1

- Part-1 ..... (5-2B to 5-19B)
  - I/O Management and Disk Scheduling : I/O Devices
  - I/O Subsystem
  - I/O Buffering
  - Disk Storage and Disk Scheduling
  - RAID
  
- A. Concept Outline : Part-1 ..... (5-2B to 5-19B)
  - I/O Long and Medium Answer Type Questions ..... 5-2B
  
- Part-2 ..... (5-19B to 5-36B)
  - File System : File Concept
  - File Organization and Access Mechanism
  - File Directories, and File Sharing
  - File System Implementation Issues
  - File System Protection and Security

- A. Concept Outline : Part-2 ..... (5-2B to 5-19B)
  - I/O Long and Medium Answer Type Questions ..... 5-20B

#### PART-1

**I/O Management and Disk Scheduling : I/O Devices, and I/O Subsystem, I/O Buffering, Disk Storage and Disk Scheduling, RAID.**

- I/O devices are the collection of interfaces that different functional units of an information processing system use to communicate with each other.
- Types of I/O buffering schemes :
  - i. Single buffer
  - ii. Double buffer
  - iii. Circular buffer
- Various disk scheduling algorithms :
  - i. FCFS scheduling
  - ii. SSTF scheduling
  - iii. SCAN scheduling
  - iv. C-SCAN scheduling
  - v. LOOK scheduling

#### Questions-Answers

#### Long Answer Type and Medium Answer Type Questions

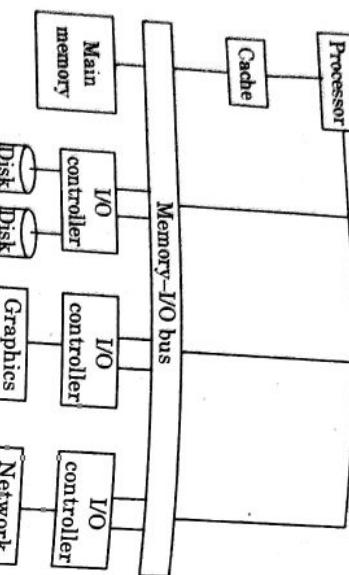
- Ques 5.1.** Explain the terms I/O devices. What are the categories of I/O devices?

#### Answer

1. Input/output, or I/O, is the collection of interfaces that different functional units (sub-systems) of an information processing system use to communicate with each other, or the signals (information) sent through those interfaces.
2. Inputs are the signals received by the unit, and outputs are the signals sent from it.
3. The term can also be used as part of an action; to "do I/O" is to perform an input or output operation.
4. I/O devices are used by a person (or other system) to communicate with a computer.

**Q4B (CS/IT-Sem-4)****Interrupt driven I/O :**

- For instance, keyboard and mouse are considered input devices of a computer and monitors and printers are considered output devices of a computer.
- Typical devices for communication between computers are for both input and output, such as modems and network cards.

**Fig. 5.11 Input/Output (I/O) Device.****Categories of I/O devices are:****1. Human readable : It is suitable for communicating with the computer user.**

For example: Printers, video display terminals, keyboard etc.

**2. Machine readable : It is suitable for communicating with electronic equipment.**

For example: Disk and tape drives, sensors, controllers and actuators.

**3. Communication : It is suitable for communicating with remote devices.**  
For example: Digital line drivers and modems.**Ques 5.2** What are the techniques for performing I/O ?**Describe the organization of I/O function.****Answer****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.

**Ques 5.3** What is buffer in devices ?**OR**Write short note on I/O buffering.**AKTU 2013-14, 2015-16, Marks 05****Answer****1. A buffer is a memory area that stores data being transferred between**

- two devices or between a device and an application.

- Buffering is done to cope with a speed mismatch between the producer and consumer of a data stream.

- A buffer is created in main memory to accumulate the bytes received from the modem.

4. When an entire buffer of data has arrived, the buffer can be written to disk in a single operation.
5. Since the disk write is not instantaneous and the modem still needs a place to store additional incoming data, two buffers are used.
6. After the modem fills the first buffer, the disk write is requested.
7. The modem then starts to fill the second buffer while the first buffer is written to disk.

By the time the modem has filled the second buffer, the disk write from the first one should have completed, so the modem can switch back to the first buffer while the disk writes the second one.

This double buffering decouples the producer of data from the consumer, thus relaxing timing requirements between them.

8. Another use of buffering is to provide adaptations for devices that have different data transfer sizes.

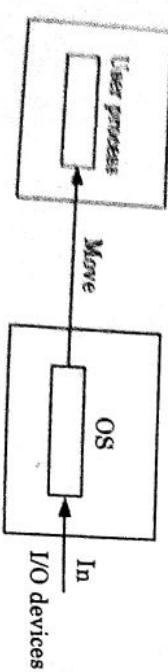
### Ques. What are the types of I/O buffering schemes?

#### Type of I/O buffering schemes:

- i. **Single buffer**: Operating system assigns a buffer in the system memory of main memory.

#### Block oriented device:

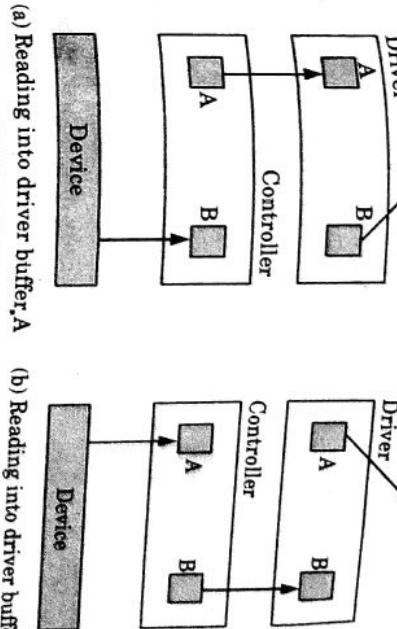
- i. Input transfers are made to the system buffer.
- ii. After transferring, the process moves the block into user space and requests for another block.
- iii. User process can be processing one block of data while the next block is being read in.
- iv. OS is able to swap the process out.
- v. OS must keep track of the assignment of system buffers to user processes.



**Fig. 5.4.1. Single buffering.**

#### 2. Double buffer:

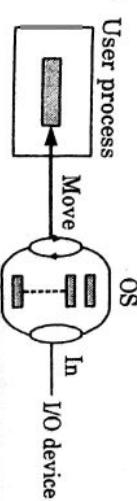
- There are two buffers in the system.
- One buffer is for the driver or controller to store data while waiting for it to be retrieved by higher level of the hierarchy.
- Other buffer is to store data from the lower level module.



**Fig. 5.4.2. Double buffering in the driver.**

#### 3. Circular buffer:

- When more than two buffers are used, the collection of buffers is itself referred to as a circular buffer.
- In this, the producer cannot pass the consumer because it would overwrite buffers before they had been consumed.
- The producer can only fill up to buffer  $j - 1$  while data in buffer  $j$  is waiting to be consumed.



**Fig. 5.4.3. Circular buffering.**

#### OR

What do you mean by caching, spooling and error handling? Explain in detail.

**AKU 2016-17, Marks 15**

**Answer**  
Kernels provide many services related to I/O. Several services such as scheduling, buffering, caching, spooling, device reservation, and error

- i. Double buffering is also called buffer swapping.
- ii. Double buffering improvement comes at the cost of increased complexity.
- iii. Double buffering may be inadequate if the process performs rapid bursts of I/O.
- iv. bursts of I/O.
- v. bursts of I/O.
- vi. bursts of I/O.

handling are provided by the kernel's I/O subsystem and build on the hardware and device driver infrastructure.

#### I/O scheduling:

- To schedule a set of I/O requests means to determine a good order in which to execute them.

- Scheduling can improve overall system performance, can share device access fairly among processes, and can reduce the average waiting time for I/O to complete.

- One way in which the I/O subsystem improves the efficiency of the computer is by scheduling I/O operations. Another way is by using storage space in main memory or on disk via techniques called buffering, caching, and spooling.

**Buffering:** Refer Q. 5.3, Page 5-4B, Unit-5.

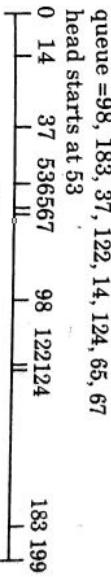
#### Caching :

- A cache is a region of fast memory that holds copies of data. Access to the cached copy is more efficient than access to the original.
- For instance, the instruction of the currently running process are stored on disk, copied in physical memory and copied again in the CPU's secondary and primary caches.
- The difference between a buffer and a cache is that a buffer may hold the only existing copy of a data item, whereas a cache, by definition, holds a copy on faster storage of an item that resides elsewhere.

#### Spooling:

- A spool is a buffer that holds output for 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 mixed together.
- The operating system solves this problem by intercepting all output to the printer.
- Each application's output is spooled to a separate disk file.
- When an application finishes printing, the spooling system queues the corresponding spool file for output to the printing, the spooling system copies the queued spool files to the printer one at a time.

**Fig. 5.6.1. FCFS disk scheduling**



**Answer**  
Various disk scheduling algorithms are :

#### i. FCFS scheduling:

- The simplest form of disk scheduling is the First-Come First-Served (FCFS) algorithm.

queue = 98, 183, 37, 122, 14, 124, 65, 67

head starts at 53

**AKTU 2012-13, Marks 10**  
Write short note on disk scheduling policies.  
**OR**

**AKTU 2014-15, Marks 10**  
Discuss disk scheduling algorithms with example.

- Operating system can often compensate effectively for transient failures.
- As a general rule, an I/O system call will return one bit of information about the status of the call, signifying either success or failure.

#### Ques 6.6. Explain FCFS, SCAN & C-SCAN scheduling with example.

**i. FCFS scheduling:**

**ii. SSTF scheduling:**

- The SSTF algorithm selects the request with the least seek time from the current head position.

2. For our example request queue, the closest request to the initial head position (53) is at cylinder 65.
  3. Once we are at cylinder 65, the next closest request is at cylinder 67. From there, the request at cylinder 37 is closer than the one at 98, so 37 is served next.
- queue = 98, 183, 37, 12, 14, 124, 65, 67  
head starts at 53

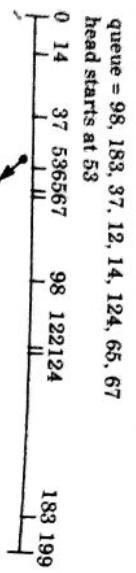


Fig. 5.6.2. SSTF disk scheduling.

4. Continuing, we service the request at cylinder 14, then 98, 122, 124, and finally 183.

### iii. SCAN scheduling:

1. In the SCAN algorithm, the disk arm starts at one end of the disk and moves towards the other end, servicing requests as it reaches each cylinder, until it gets to the other end of the disk.
2. At the other end, the direction of head movement is reversed, and servicing continues.
3. The head continuously scans back and forth across the disk.
4. Assuming that the disk arm is moving towards 0 and that the initial head position is again 53, the head will next service 37 and then 14.
5. At cylinder 0, the arm will reverse and will move towards the other end of the disk, servicing the requests at 65, 67, 98, 122, 124, and 183.
6. If a request arrives in the queue just in front of the head, it will be serviced almost immediately; a request arriving just behind the head will have to wait until the arm moves to the end of the disk, reverses direction, and comes back.

### w. C-SCAN scheduling:

1. Circular SCAN (C-SCAN) scheduling is a variant of SCAN designed to provide a more uniform wait time.
2. Like SCAN, C-SCAN moves the head from one end of the disk to the other, servicing requests along the way.
3. When the head reaches the other end, however, it immediately returns to the beginning of the disk without servicing any requests on the return trip.
4. The C-SCAN scheduling algorithm essentially treats the cylinders as a circular list that wraps around from the final cylinder to the first one.

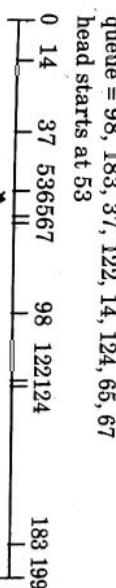


Fig. 5.6.3. SCAN disk scheduling.



### v. LOOK scheduling:

1. In this the disk arm reverses direction immediately without going all the way to the end of the disk.
2. Versions of SCAN and C-SCAN that follow this pattern are called LOOK and C-LOOK scheduling because they look for a request before continuing to move in a given direction.

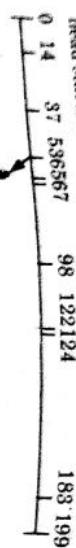
Fig. 5.6.4. C-SCAN disk scheduling.

I/O B (CSMT-Sem-4)

queue = 98, 183, 37, 122, 14, 124, 65, 67  
head starts at 53



queue = 98, 183, 37, 122, 14, 124, 65, 67  
head starts at 53



$$\begin{aligned} \text{Total time} &= (110 - 100) + (120 - 110) + (129 - 120) \\ &\quad + (147 - 129) + (186 - 147) + (186 - 64) \\ &= 10 + 10 + 9 + 18 + 39 + 122 + 23 + 14 + 17 \\ &= 262 \end{aligned}$$



### SSTF disk scheduling policies:

- Ques 5.7.** Explain the SSTF and SCAN disk scheduling policies.  
Obtain the total number of head movements needed to satisfy the following sequence of track requests for each of the two policies:  
Assume that the disk head is initially positioned over track 100 and is moving in the direction of decreasing track number.

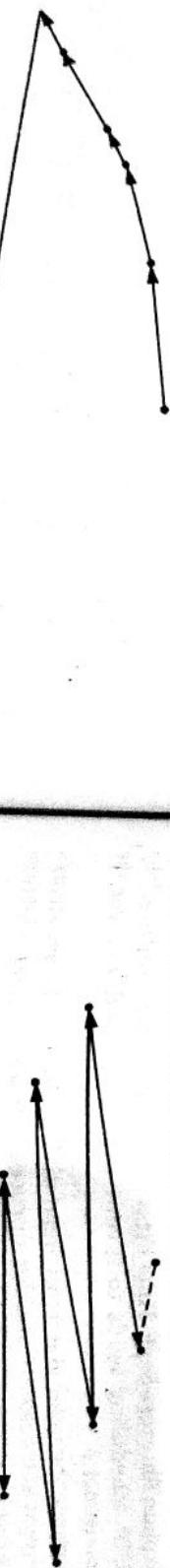
**AKTU 2011-12, Marks 10**

**Ques 5.8.** 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 §§ 147, 91, 177, 94, 150. What is total head movement for the following scheduling:

- FCFS
- SSTF
- C-Scan

**AKTU 2013-14, 2015-16; Marks 10**

- Answer**
- SSTF and SCAN disk scheduling algorithm : Refer Q. 5.6, Page 5-8B,  
Unit-5.
- Numerical :**  
SCAN disk scheduling policy :
- Disk initially at 100  
Track's are from 0 - 199  
queue = 27, 129, 110, 186, 147, 41, 10, 64, 120  
currently head at 100.



Total head movement,

$$\begin{aligned} &= (100 - 64) + (64 - 41) + (41 - 27) + (27 \\ &\quad - 10) + (10 - 0) + (110 - 0) + (120 - 110) + \\ &\quad (129 - 120) + (147 - 129) + (186 - 147) \end{aligned}$$

**Fig. 5.7.1.**

$$\begin{aligned} \text{Total distance} &= (143 - 86) + (147 - 86) + (147 - 91) + (177 - 91) \\ &\quad + (177 - 94) + (150 - 94) = 399 \end{aligned}$$

**Fig. 5.8.1.**

**SSTF tends to favour middle cylinders over the innermost:**

The current location of the head divides the cylinders into two groups. If the head is not in the center of the disk and a new request arrives, the new request is more likely to be in the group that includes the center of the disk; thus, the head is more likely to move in that direction.

ii. **Shortest Seek Time First (SSTF):**

|   |    |    |    |     |     |     |     |     |
|---|----|----|----|-----|-----|-----|-----|-----|
| 0 | 86 | 91 | 94 | 125 | 143 | 147 | 150 | 177 |
|---|----|----|----|-----|-----|-----|-----|-----|



Fig. 5.8.2.

Total distance =  $(147 - 143) + (150 - 147) + (177 - 150) + (177 - 94) + (94 - 91) + (91 - 86) = 125$

iii. **Circular Scan (C-scan):**

|   |    |    |    |     |     |     |     |     |
|---|----|----|----|-----|-----|-----|-----|-----|
| 0 | 86 | 91 | 94 | 125 | 143 | 147 | 150 | 177 |
|---|----|----|----|-----|-----|-----|-----|-----|



Fig. 5.8.3.

$$\text{Total distance} = (147 - 143) + (150 - 147) + (177 - 150) + (177 - 94) + (94 - 91) + (91 - 86) = 125$$

**Ques. 9.** Discuss in what sense is SCAN fairer than SSTF in disk scheduling.

**Answer**

1. In SCAN, the drive head sweeps across the entire surface of the disk, visiting the outermost cylinders before changing direction and sweeping back to the innermost cylinders.
2. It selects the next waiting requests whose location it will reach on its path backwards and forwards across the disk.
3. Thus, the movement time should be less than FCFS but the policy is clearly fairer than SSTF.

**Ques 5.10.**

i. Explain Shortest-Seek-Time First (SSTF) disk scheduling. Why

SSTF scheduling tends to favour middle cylinders over the innermost?

- ii. 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.
1. Draw track chart for FCFS, SSTF and SCAN algorithms of disk scheduling.
2. Calculate total head movement in tracks in each case.

**Answer**

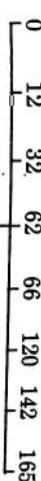
- i. SSTF disk scheduling: Refer Q. 5.6, Page 5-8B, Unit-5.

**FCFS :**



Fig. 5.10.1.

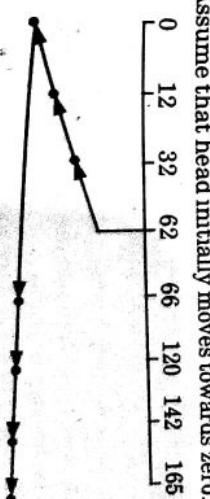
**SSTF :**



Total head movement

$$= (66 - 62) + (165 - 66) + (165 - 32) + (120 - 32) \\ + (120 - 12) + (142 - 12) \\ = 4 + 99 + 133 + 88 + 108 + 130 = 562$$

**SCAN :** Assume that head initially moves towards zero.



Total head movement

$$= (62 - 32) + (32 - 12) + (12 - 0) + (66 - 0) \\ + (120 - 66) + (142 - 120) + (165 - 142) \\ = 30 + 20 + 12 + 66 + 54 + 22 + 23 \\ = 227$$

# Chitrani and @man

## Ques 5.11 Discuss boot blocks.

**AKTU 2014-15, Marks 08**

### Answer

- 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 operating system, this block can be empty. This is called boot block.
- For a computer to start running, for instance, when it is powered up or rebooted, it must have an initial program to run. This initial bootstrap
- initializes all aspects of the system, from CPU registers to device operating system.
- To do its job, the bootstrap program finds the operating system kernel on disk, loads that kernel into memory, and jumps to an initial address to begin the operating system execution.

## Ques 5.12 Explain the following methods

**B**

7

**AKTU 2015-16, Marks 10**

OR

Discuss free space management in disk.

### Answer

#### i. Bit vector :

- Frequently, 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 example, consider a disk where blocks 2, 3, 4, 5, 8, 9, 10, 11, 12, 13, 17, 18, 25, 26, and 27 are free and the rest of the blocks are allocated. The free-space bit map would be :

0011100111110001100000011100000...

The main advantage of this approach is its relative simplicity and its efficiency in finding the first free block or  $n$  consecutive free blocks on the disk.

- Indeed, many computers supply bit-manipulation instructions that can be used effectively for that purpose.

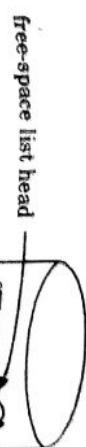


Fig. 5.12.1. Linked free-space list on disk

- Linked list :**  
Another approach to free-space management is to link together all the free disk blocks, keeping a pointer 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 block, and so on.
- For example, blocks 2, 3, 4, 5, 8, 9, 10, 11, 12, 13, 17, 18, 25, 26, and 27 were free, and the rest of the blocks were allocated.
- In this situation, we would keep a pointer to block 2 as the first free block. Block 2 would contain a pointer to block 3, which would point to block 4, which would point to block 5, which would point to block 8, and so on.
- This scheme is not efficient; to traverse the list, we must read each block, which requires substantial I/O time.
- However, traversing the free list is not a frequent action.
- Usually, the operating system simply needs a free block so that it can allocate that block to a file, so the first block in the free list is used.
- The FAT method incorporates free-block accounting into the allocation data structure.
- No separate method is needed.

#### ii. Grouping :

- A modification of the 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.

4. The addresses of a large number of free blocks can now be found quickly, unlike the situation when the standard linked list approach is used.
- iv. Counting :**
- Another approach takes advantage of the fact that, generally, several contiguous blocks may be allocated or freed simultaneously, particularly when space is allocated with the contiguous allocation algorithm or through clustering.
  - Thus, rather than keeping a list of  $n$  free disk addresses, we can keep the address of the first free block and the number ( $n$ ) of free contiguous blocks that follow the first block.
  - Each entry in the free-space list then consists of a disk address and a count.
  - Although each entry requires more space than a simple disk address, the overall list is shorter, as long as the count is generally greater than 1.
5. These entries can be stored in a balanced tree, rather than a linked list, for efficient lookup, insertion, and deletion.

**Que 5.13 | Write short note on RAID.**

**AKTU 2012-13, Marks 10**

**Answer**

- RAID (Redundant Arrays of Independent Disks) is a variety of disk organization techniques, commonly used to address the performance and reliability issues.
  - In the past RAIDS composed of small, cheap disks were viewed as a cost-effective alternative to large, expensive disks.
  - Today, RAIDS are used for their higher reliability and higher data-transfer rate, rather than for economic reasons.
  - Hence, the I in RAID, which once stood for "inexpensive," now stands for "Independent".
- RAID level configurations :** There are almost dozen RAID level schemes prevailing in RAID technology. But only the most prevailing RAID level schemes are summarized below.
- RAID Level 0 :**
    - This RAID level 0 configuration does not provide redundancy and so it does not exactly fit into the arrays of RAID technology.
    - In this RAID level 0, two disks are used to write data to two drives in an alternating way, which is striping.
    - This splitting of data will allow doubling of speed of a single hard drive and will also enhance its performance.
    - But if one drive fails in this array, then data loss is incurred. The capacity of RAID 0 is equal to the whole sum of the individual drives.

**RAID Level 1 :**

- From this RAID level 1, the redundancy factor becomes influential.
- Minimum use of hard drives in this level will be two and the data is written to both drives.
- It is like cloning or mirroring the data of the first drive to the second one and making drive one identical to the second one.
- If the first drive fails, then the data backup will be available from the drive 2.
- In this RAID level, the actual capacity of RAID 1 level array will be equal to half the capacity of the sum of individual drives.

**RAID Level 2 :**

- In this form of RAID data is striped in a way that each sequential bit is on different drive.
- Each data word is having its own hamming code and on each read, the Hamming code verifies the data accuracy and also corrects the single disk errors.
- In this level, the array can recover from multiple and simultaneous hard drive failures.
- Minimum two drives are required in this RAID level 2.

**RAID Level 3 :**

- In this level of RAID a minimum of three drives is required for implementation.
- The data block is split and is written on data disks.
- Stripe parity is generated on writes and this writing is recorded on parity disks and can be checked on reads.
- A RAID 3 array can recover from hard drive failures and is deployed in environments where applications need high speed throughputs, which can be video production, video editing and live streaming.
- In this level the read/write speed is high.

**RAID Level 4 :**

- In this RAID level the requirement of three drives is seen and in this level, the entire block is written on a disk.
- Parity is generated in writes and recorded on parity disks and checked on reads.
- This level of RAID has high reading speeds and is highly efficient.

**RAID Level 5 :**

- In this RAID 5 level, three drives are implemented and data block is written on a data disks and parity which is generated from writes is distributed onto three drives and is checked on reads.

## CONCEPT OUTLINE : PART-2

File organization refers to the way the data is stored in file.

Four methods of organizing file:

### 7. RAID Level 6:

- a. In the situation of drive failure, the reads can be calculated from the distributed parity and the drive failure is masked from the user.
- c. But in situations of single drive failure, the performance of the entire array gets depleted.

### 8. RAID Level 6:

- s. This level of RAID 6, offers fault tolerance in case of two drive failures and still the system functions.
- b. In this level, block level stripping is observed along with double distributed parity.
- c. In case of data recovery, the time for recovery takes place on the size of the disk drive.
- d. Double parity offers additional time for rebuilding the array without the data being at risk if single additional drive fails, while the data recovery through rebuild is happening.

### 8. RAID Level 10 (1 + 0):

- a. In this level of RAID minimum requirement of drives is 4.
- b. RAID 10 will also have fault tolerance and will also have redundancy.
- c. It will have the splitting of data feature seen in RAID 0 level and will also have mirroring feature seen in RAID 1 level.
- d. RAID 10 array can recover from multiple and simultaneous hard drive failures and is ideal for high end server applications.

### 9. RAID Level 0 + 1:

- a. This level of RAID requires a minimum of 4 drives and multiple drive failures can be handled by this RAID level.
- b. This level is a combination of RAID 0 and RAID 1 level and is used in imaging applications meant file servers.
- c. It offers high performance and reduces emphasis on reliability.
- d. In the RAID 0 + 1 a second striped set to mirror the primary set is created.
- e. The array continues to operate with one or more drives failure in mirror set.

### Questions-Answers

#### Long Answer Type and Medium Answer Type Questions

##### Ques 5.14. Write a short note on file systems.

#### Answer

- 1. The file system permits users to create data collections, called files, with desirable properties:
  - i. **Long term existence :** Files are stored on disk or other secondary storage and do not disappear when a user logs off.
  - ii. **Sharable between processes :** Files have names and can have associated access permissions that permit controlled sharing.
  - iii. **Structure :** Depending on the file system, a file can have an internal structure that is convenient for particular applications.
- 2. File is the collection of data created by user.
- 3. It provides a means to store data organized as file as well as a collection of function that can be performed on file.
- 4. It maintains a set of attributes associated with the file.

#### Operation included in file system are :

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

#### **File System : File Concept, File Organization and Access**

#### **Mechanism, File Directories, and File Sharing, File System Implementation Issues, File System Protection and Security.**

## PART-2

1. Name : It is the only information which is in human-readable form.

- 2. Identifier :** The file is identified by a unique tag(number) within file system.

- 3. Type :** It is needed for systems that support different types of files.

- 4. Location :** Pointer to file location on device.

- 5. Size :** The current size of the file.

- 6. Protection :** This controls and assigns the power of reading, writing, executing.

- 7. Time, date, and user identification :** This is the data for protection, security, and usage monitoring.

**Ques 5.15** Write a short note on file organization.

**Answer**

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.

**Four methods of organizing files :**

**1. Sequential file organization :**

- a. Records are stored and accessed in a particular sorted order using a key field.

- b. Retrieval requires searching sequentially through the entire file record by record to the end.

**2. Random or direct file organization :**

- a. Records are stored randomly but accessed directly.

- b. To access a file which is stored randomly, a record key is used to determine where a record is stored on the storage media.

- c. Magnetic and optical disks allow data to be stored and accessed randomly.

**3. Serial file organization :**

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

- b. This type of organization is mainly used on magnetic tapes.

**4. Indexed-sequential file organization method :**

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

- b. For example, on a magnetic drum, records are stored sequentially on the tracks. However, each record is assigned an index that can be used to access it directly.

**Ques 5.16.** Explain file organization and access mechanism.

**AKTU 2016-17, Marks 10**

**Answer**

**File organization :** Refer Q. 5.15, Page 5-21B, Unit-5.

**Access mechanism :** Following are the access mechanism which are provided to access a file from the operating system are as follows :

**1. Sequential Access :**

- a. It is the simplest access mechanism, in which information stored in a file are accessed in an order such that one record is processed after the other.

- b. For example editors and compilers usually access files in this manner.

**2. Direct Access :**

- a. It is an alternative method for accessing a file, which is based on the disk model of a file, since disk allows random access to any block or record of a file.

- b. For this method, a file is viewed as a numbered sequence of blocks or records which are read/written in an arbitrary manner. i.e., there is no restriction on the order of reading or writing.

**3. Index Access :**

- a. In this method an index is created which contains a key field and pointers to the various blocks.

- b. To find an entry in the file for a key value, we first search the index and then use the pointer to directly access a file and find the desired entry.

**Ques 5.17.** What do you mean by directory? What are the operations that can be performed on a directory?

OR

What is a directory? Define any two ways to implement the directory.

**AKTU 2013-14, Marks 10**

OR

What is directory? Explain any two ways to implement the directory.

**AKTU 2013-14, Marks 10**

**Answer**

**What is directory?** A directory is a table of contents for a file system, listing the names of files and their locations.

**Two ways to implement the directory:**

1. The directory can be viewed as a symbol table that translates file names into their directory entries.

2. If we take such a view, we see that the directory itself can be organized in many ways.

3. The organization must be allowed to insert entries, to delete entries, to search for a named entry, and to list all the entries in the directory.

**Directory implementation :**

Directory is implemented in two ways :

1. **Linear List :**

- Linear list is a simplest method.
- 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.
- For creating new file, it searches the directory for the name whether same name already exists.
- Linear search is the main disadvantage.
- Directory information is used frequently and users would notice a slow implementation of access to it.

2. **Hash table :**

- Hash table decreases the directory search time.
- Insertion and deletion are also fairly 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 :**

- Search :** Directory structure is searched for finding particular file in the directory. Files have symbolic names and similar names may indicate a relationship between files.
- Create a file :** When a new file is created, an entry must be added to the directory.
- Delete a file :** When a file is deleted, an entry must be removed from the directory.
- Rename a file :** Name of the files must be changeable when the content or use of the file changes. Renaming a file may also allow its position within the directory structure to be changed.
- List directory :** All or portion of the directory may be requested. Request is made by a user and result in a listing of all files owned by that user plus some of the attributes of each file.

**Que 5.18.]** Describe schemes for defining logical structure of directory.

**Answer**

Different types of directory structure are :

1. **Single-level directory :**

- The simplest directory structure is the single-level directory.

All files are contained in the same directory, which is easy to support and understand.

A single-level directory has significant limitations, when the number of files increases or when the system has more than one user.

Since all files are in the same directory, they must have unique names. If two users call their data file test.txt, then the unique name rule is violated.

Fortunately, most file systems support file names of up to 255 characters, so it is relatively easy to select unique file names.

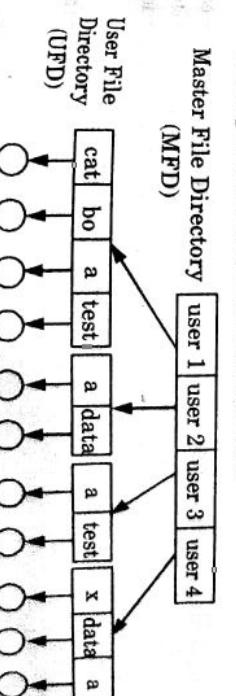
Even a single user on a single-level directory may find it difficult to remember the names of all the files as the number of file increases.



**Fig. 5.18.1. Single-level directory.**

2. **Two-level directory :**

- In the two-level directory structure, each user has his own User File Directory (UFD).
- The UFDs have similar structures, but each lists only the files of a single user.
- When a user job starts or a user logs in, the system's Master File Directory (MFD) is searched.
- The MFD is indexed by user name or account number, and each entry points to the UFD for that user.



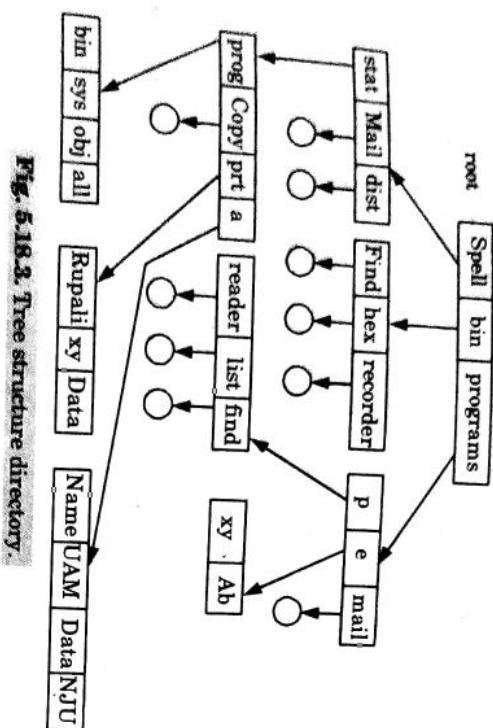
**Fig. 5.18.2. Two-level directory structure.**

- When a user refers to a particular file, only his own UFD is searched.
- Thus, different users may have files with the same name, as long as all the file names within each UFD are unique.
- To create a file for a user, the operating system searches only that user's UFD to ascertain whether another file of that name exists.
- To delete a file, the operating system confines its search to the local UFD; thus, it cannot accidentally delete another user's file that has the same name.

**AKTU 2014-15, Marks 10**

**3. Tree structured directories :**

- MS-DOS system is a tree structure directory.
  - It allows users to create their own sub-directory and to organize their files accordingly.
  - A sub-directory contains a set of files or sub-directories.
  - A directory is simply another file, but it is treated in a special way.
- Fig. 5.18.3 shows tree structured directories.

**Fig. 5.18.3 Tree structure directory**

e.

All the directories have the same internal format. One bit in each directory entry defines the entry as a file (0) or as a sub-directory (1). Special system calls are used to create and delete directories.

f.

In normal use, each user has a current directory.

g.

Current directory should contain most of the files that are of current interest to the user.

h.

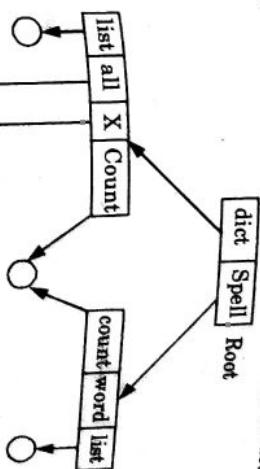
When a reference is made to a file, the current directory is searched.

i. Path name is used to search for any operation on file with another directory.

**4. Acyclic graph directories :**

- It allows directories to have shared subdirectories and files.
- Same file or directory may be in two different directories.
- Graph with no cycles is a generalization of the tree structured subdirectory scheme.
- Shared files and subdirectories can be implemented by using links.

- A link is effectively a pointer to another file or subdirectory.
- A link is implemented as an absolute or relative path name.
- An acyclic graph directory structure is more flexible than a simple tree structure, but sometimes it is more complex.
- Fig. 5.18.4 shows an acyclic graph directory structure.

**Fig. 5.18.4 Acyclic graph directory structure****Ques 5.19. Explain the concept of file sharing.****Answer**

- File sharing is the sharing of files among several users, while working together on a project.
- It is the private or public distribution of data or resources in a network with different levels of sharing privileges.
- With the advent of the Internet, a file transfer system called the File Transfer Protocol (FTP) has become widely-used.
- FTP can be used to access (read and possibly write to) files shared among a particular set of users with a password to gain access to files shared from an FTP server site.
- Many FTP sites offer public file sharing or at least the ability to view or copy files by downloading them, using a public password (which happens to be "anonymous").
- File sharing implies a system in which users write to as well as read files or in which users are allotted some amount of space for personal files on a common server, giving access to other users as they see fit.
- File sharing can be viewed as part of file systems and their management.

**Operating Systems**

8. Operating systems also provide file-sharing methods, such as network file sharing (NFS).
- Most file-sharing tasks use two basic sets of network criteria, as follows :
- a. **Peer-to-Peer (P2P) file sharing :**
- This is the most popular, but controversial, method of file sharing because of the use of peer-to-peer software.
  - Network computer users locate shared data with third-party software.
  - P2P file sharing allows users to directly access, download and edit files.

**b. File hosting services :**

- This P2P file-sharing alternative provides a broad selection of popular online material.
- These services are quite often used with internet collaboration methods, including email, blogs, forums, or other mediums, where direct download links from the file hosting services can be included.
- These service websites usually host files to enable users to download them.

**Que 5.20.** How blocks are allocated in secondary storage management ?

OR  
What are the different methods of allocating disk space ?

Describe the various file allocation methods with their relative advantages and disadvantages.

**AKTU 2011-12, Marks 10**

**Answer****Methods of allocating disk space are :**

- i. **Contiguous allocation :**
- Contiguous allocation requires that each file occupy a set of contiguous blocks on the disk.
  - Due to this reason, it is also known as continuous file access. Disk addresses define a linear ordering on the disk. With this ordering, assuming that only one job is accessing the disk, accessing block  $b \pm 1$  after block  $b$  normally requires no head movement.
  - When head movement is needed (from the last sector of one cylinder to the first sector of the next cylinder), the head need only move from one track to the next.
  - Thus, the number of disk seeks required for accessing contiguously allocated files is minimal, as is seek time when a seek is finally needed.

Contiguous allocation of a file is defined by the disk address length (in block units) of the first block.

If the file is  $n$  blocks long and starts at location  $b$ , then it occupies blocks  $b, b + 1, b + 2, \dots, b + n - 1$ .

The directory entry for each file indicates the address of the starting block and the length of the area allocated for this file.

| file  | start | length |
|-------|-------|--------|
| count | 0     | 2      |
| tr    | 14    | 3      |
| mail  | 19    | 6      |
| list  | 28    | 4      |
| f     | 6     | 2      |

| count | 1 | 2 | 3 | <i>f</i> |
|-------|---|---|---|----------|
| 0     |   |   |   |          |
| 4     |   |   |   |          |
| 8     |   |   |   |          |
| 12    |   |   |   |          |
| 16    |   |   |   |          |
| 20    |   |   |   |          |
| 24    |   |   |   |          |
| 28    |   |   |   |          |
| 32    |   |   |   |          |
| 36    |   |   |   |          |
| 40    |   |   |   |          |
| 44    |   |   |   |          |
| 48    |   |   |   |          |
| 52    |   |   |   |          |
| 56    |   |   |   |          |
| 60    |   |   |   |          |
| 64    |   |   |   |          |
| 68    |   |   |   |          |
| 72    |   |   |   |          |
| 76    |   |   |   |          |
| 80    |   |   |   |          |
| 84    |   |   |   |          |
| 88    |   |   |   |          |
| 92    |   |   |   |          |
| 96    |   |   |   |          |
| 100   |   |   |   |          |
| 104   |   |   |   |          |
| 108   |   |   |   |          |
| 112   |   |   |   |          |
| 116   |   |   |   |          |
| 120   |   |   |   |          |
| 124   |   |   |   |          |
| 128   |   |   |   |          |
| 132   |   |   |   |          |
| 136   |   |   |   |          |
| 140   |   |   |   |          |
| 144   |   |   |   |          |
| 148   |   |   |   |          |
| 152   |   |   |   |          |
| 156   |   |   |   |          |
| 160   |   |   |   |          |
| 164   |   |   |   |          |
| 168   |   |   |   |          |
| 172   |   |   |   |          |
| 176   |   |   |   |          |
| 180   |   |   |   |          |
| 184   |   |   |   |          |
| 188   |   |   |   |          |
| 192   |   |   |   |          |
| 196   |   |   |   |          |
| 200   |   |   |   |          |
| 204   |   |   |   |          |
| 208   |   |   |   |          |
| 212   |   |   |   |          |
| 216   |   |   |   |          |
| 220   |   |   |   |          |
| 224   |   |   |   |          |
| 228   |   |   |   |          |
| 232   |   |   |   |          |
| 236   |   |   |   |          |
| 240   |   |   |   |          |
| 244   |   |   |   |          |
| 248   |   |   |   |          |
| 252   |   |   |   |          |
| 256   |   |   |   |          |
| 260   |   |   |   |          |
| 264   |   |   |   |          |
| 268   |   |   |   |          |
| 272   |   |   |   |          |
| 276   |   |   |   |          |
| 280   |   |   |   |          |
| 284   |   |   |   |          |
| 288   |   |   |   |          |
| 292   |   |   |   |          |
| 296   |   |   |   |          |
| 300   |   |   |   |          |
| 304   |   |   |   |          |
| 308   |   |   |   |          |
| 312   |   |   |   |          |
| 316   |   |   |   |          |
| 320   |   |   |   |          |
| 324   |   |   |   |          |
| 328   |   |   |   |          |
| 332   |   |   |   |          |
| 336   |   |   |   |          |
| 340   |   |   |   |          |
| 344   |   |   |   |          |
| 348   |   |   |   |          |
| 352   |   |   |   |          |
| 356   |   |   |   |          |
| 360   |   |   |   |          |
| 364   |   |   |   |          |
| 368   |   |   |   |          |
| 372   |   |   |   |          |
| 376   |   |   |   |          |
| 380   |   |   |   |          |
| 384   |   |   |   |          |
| 388   |   |   |   |          |
| 392   |   |   |   |          |
| 396   |   |   |   |          |
| 400   |   |   |   |          |
| 404   |   |   |   |          |
| 408   |   |   |   |          |
| 412   |   |   |   |          |
| 416   |   |   |   |          |
| 420   |   |   |   |          |
| 424   |   |   |   |          |
| 428   |   |   |   |          |
| 432   |   |   |   |          |
| 436   |   |   |   |          |
| 440   |   |   |   |          |
| 444   |   |   |   |          |
| 448   |   |   |   |          |
| 452   |   |   |   |          |
| 456   |   |   |   |          |
| 460   |   |   |   |          |
| 464   |   |   |   |          |
| 468   |   |   |   |          |
| 472   |   |   |   |          |
| 476   |   |   |   |          |
| 480   |   |   |   |          |
| 484   |   |   |   |          |
| 488   |   |   |   |          |
| 492   |   |   |   |          |
| 496   |   |   |   |          |
| 500   |   |   |   |          |

**Fig 5.20.1. Contiguous allocation of disk space.**

**Advantages of contiguous allocation :**

- It is the simplest allocation scheme and can be implemented easily.
- Gives excellent performance.

**Disadvantages of contiguous allocation :**

- The size of the file up to maximum limit must be known at the creation time.
- Some disk space is wasted due to fragmentation. Compaction is required to remove fragmentation.

**ii. Linked allocation :**

- With linked allocation, each file is a linked list of disk blocks; the disk blocks may be scattered anywhere on the disk.
- The directory contains a pointer to the first and last blocks of the file.
- For example, a file of five blocks might start at block 9 and continue at block 16, then block 1, then block 10, and finally block 25.
- Each block contains a pointer to the next block.
- These pointers are not made available to the user.
- Thus, if each block is 512 bytes in size, and a disk address (the pointer) requires 4 bytes, then the user sees blocks of 508 bytes.

- When the  $i^{\text{th}}$  block is first written, a block is obtained from the free-space manager, and its address is put in the  $i^{\text{th}}$  index block entry.

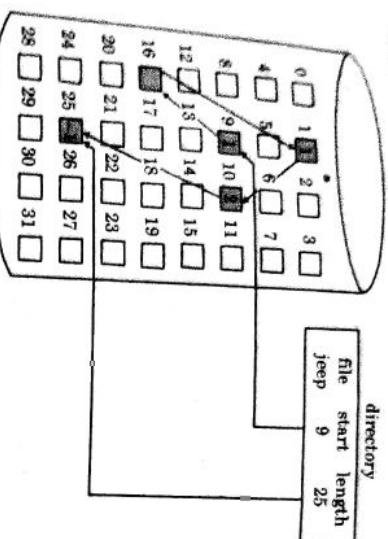


Fig. 5.29.2. Linked allocation of disk space.

**Advantages of linked allocation :**

1. There is no waste of memory.
2. There is no external fragmentation. Only internal fragmentation is possible in the last block.
3. Directories are required to recognize only starting block address and end block address.
4. There is no need to declare size of the file at the time of its creation.

**Disadvantages of linked allocation :**

1. Space is wasted in storing pointers.
2. Access time, seek time and latency time are higher than the contiguous allocation.

**iii. Indexed allocation :**

1. Indexed allocation solves the problems caused by linked allocation by bringing all the pointers together into one location: the index block.
2. Each file has its own index block, which is an array of disk-block addresses.
3. The  $i^{\text{th}}$  entry in the index block points to the  $i^{\text{th}}$  block of the file.
4. The directory contains the address of the index block. To find and read the  $i^{\text{th}}$  block, we use the pointer in the  $i^{\text{th}}$  index-block entry.
5. When the file is created, all pointers in the index block are set to null.

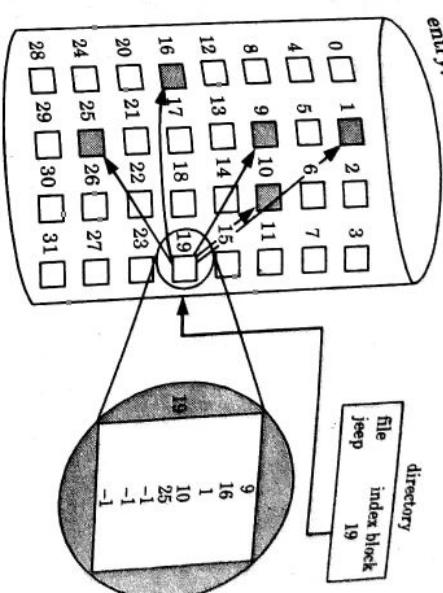


Fig. 5.29.3. Indexed allocation of disk space.

**Advantages of index allocation :**

1. The entire block is available for data using this method.
2. There is no external fragmentation using this method.
3. Directory entry needs only to keep the starting block number of a file.

**Disadvantage of index allocation :**

1. The entire index table must be in memory all the time to make it work.
2. Large space overhead (index).

**Ques 5.21.** Discuss the mechanisms used for indexed allocation.

**Answer**

**Mechanisms used for indexed allocation :**

1. **Linked scheme :**
  - i. In this an index block is normally one disk block. Thus, it can be read and written directly by itself.
  - ii. To allow for large files, we can link together several index blocks.
  - iii. For example, an index block might contain a small header giving the name of the file and a set of the first 100 disk-block addresses.
2. **Multilevel index :**
  - i. A variant of linked representation uses a first-level index block to point to a set of second-level index blocks, which in turn point to the file blocks.

- Ques 5.21** (CS/IT-Sem-4)
- First field in the each record is referred as the key field. The key field uniquely identifies the record; thus key values for different records are always different.
- To access a block, the operating system uses the first-level index to find a second-level index block and then uses that block to find the desired data block.
  - This approach could be continued to a third or fourth level, depending on the desired maximum file size. With 4,096-byte blocks, we could store 1,024 four-byte pointers in an index block, of up to 4 GB.
  - Two levels of indexes allow 1,048,576 data blocks and a file size of up to 4 GB.

### 3. Combined scheme :

- Another alternative, used is to keep the first, say, 15 pointers of the index block in the files mode. The first 12 of these pointers point to direct blocks; i.e., they contain addresses of blocks that contain data of the file.
- Thus, the data for small files (of no more than 12 blocks) do not need a separate index block.
- If the block size is 4 KB, then up to 48 KB of data can be accessed directly.
- The next three pointers point to indirect blocks.
- Under this method, the number of blocks that can be allocated to a file exceeds the amount of space addressable by the four-byte file pointers used by many operating systems.

**Ques 5.22.** Write short notes on :

- Sequential file
- Indexed file

### Answer

#### i. Sequential file:

- Sequential file uses fixed format for records. All records are of the same length, consisting of the same number of fixed length field in a particular order.

2. Fig. 5.22.1 shows the sequential file.

|  |  |  |  |
|--|--|--|--|
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

**ANSWER 2013-14, 2015-16; Marks 10**

#### ii. Indexed file :

- To achieve the flexibility, a structure is needed that employs multiple indexes. In general indexed file, the concept of sequentiality and a single key are abandoned. Records are only accessed through their indexes. Records of variable lengths can be used in this access.
- Two types of indexes are used. An exhaustive index contains one entry for every record in the main file.
- An index is itself organized as a sequential file for ease of searching. A partial index contains entries to record where the field of interest exists.
- With records of variable length, some records will not contain all fields.
- When a new record is added to the main file, all the index file must be updated.
- Fig. 5.22.2 shows the index with relative files.

- Indexed file are used mostly in applications where timelines of information are critical and where data are rarely processed exhaustively. Examples are airline reservation systems and inventory control system.
- Last name logical record number

|   |  |
|---|--|
| A |  |
| B |  |
| C |  |
| T |  |

|     |     |     |
|-----|-----|-----|
| PQR | XYZ | 123 |
|-----|-----|-----|

Access is permitted or denied depending on several factors, one of which is the type of access required. Different types of operations many be controlled in access type. These are :

c. **Read :** Read from the file

i. **Write :** Write the file

ii. **Execute :** Load the file into memory and execute it.

iii. **Append :** Write a new information at the end of the file.

iv. **Delete :** Delete the file and free its space for possible reuse.

v. **List :** List the name and attributes of the file.

#### Access control :

Various users may need different types of access to a file or directory.

a. When a user requests a particular file, the operating system checks the access list associated with that file.

b. If that user is listed for the requested access, the access is allowed. Otherwise a protection violation occurs, and the user job is denied access to the file.

d. Many system recognize three classifications of users in connection with each file for access control which are following :

i. **Owner :** User who created the file.

ii. **Group :** Set of users who are sharing the file and need similar access.

iii. **Universe :** All other users in the system constitute the universe.

#### Que 5.24] Write short note on file system protection.

**AKTU 2011-12, Marks 10**

#### Que 5.25. Discuss access matrix.

**AKTU 2014-15 Marks 05**

What is access matrix ? How it can be represented ?

#### Answer

In file system computer file needs protection which is of two types :

##### 1. Reliability:

- It means protect the file system from physical damage
- File system can be damaged by following ways :

- Hardware problem
- Power failure
- Head crashes
- Bugs in file system software

##### 2. Security:

- Security means protect the file system from improper access.
- Security can be provided by controlling access to files.

##### 3. Controlled access :

- Protection mechanisms provide controlled access by limiting the type of file access that can be made.

#### Answer

In file system computer file needs protection which is of two types :

##### 1. Reliability:

- It means protect the file system from physical damage
- File system can be damaged by following ways :

- Hardware problem
- Power failure
- Head crashes
- Bugs in file system software

##### 2. Security:

- Security means protect the file system from improper access.
- Security can be provided by controlling access to files.

##### 3. Controlled access :

- Protection mechanisms provide controlled access by limiting the type of file access that can be made.

The capability list is associated with a domain, but it is never directly accessible to a process executing in that domain. Rather, the capability list is itself a protected object, maintained by the operating system and accessed by the user only indirectly.

**the operating mechanism :**

- Current objects :** Current objects are a finite set of entities to which access is to be controlled. The set is denoted by 'O'. A typical example of an object is a file.
- Current subjects :** Current subjects are a finite set of entities that access current objects. The set is denoted by 'S'. A typical example of a subject is a process.
- Generic rights :** A finite set of generic rights,  $R = \{r_1, r_2, r_3, \dots, r_m\}$ , gives various access rights that subjects can have to objects. Typical examples of such rights are read, write, execute, own, delete etc.

**Ques 5.26.** Write short note on implementation of access matrix.

**AKTU 2012-13, Marks 10**

**Answer**

- In general, the matrix will be sparse; that is, most of the entries will be empty.
- Although data structure techniques are available for representing sparse matrices, they are not particularly useful for this application, because of the way in which the protection facility is used.

**Following are four methods for implementing the access matrix:**

- The simplest implementation of the access matrix is a global table consisting of a set of ordered triples  $\langle \text{domain}, \text{object}, \text{rights-set} \rangle$ .
- Whenever an operation  $M$  is executed on an object  $O_j$  within domain  $D_i$ , the global table is searched for a triple  $\langle D_i, O_j, R_k \rangle$ , with  $M \in R_k$ .

- If this triple is found, the operation is allowed to continue; otherwise, an exception (or error) condition is raised.

**2 Access lists for objects :**

- Each column in the access matrix can be implemented as an access list for one object.

- Obviously, the empty entries can be discarded.

- The resulting list for each object consists of ordered pairs  $\langle \text{domain}, \text{rights-set} \rangle$ , which define all domains with a non-empty set of access rights for that object.

- This approach can be extended easily to define a list plus a default set of access rights.

**3 Capability lists for domains :**

- Rather than associating the columns of the access matrix with the objects as access lists, we can associate each row with its domain.

- A capability list for a domain is a list of objects together with the operations allowed on those objects.

- An object is often represented by its physical name or address, called a capability.

The capability list is associated with a domain, but it is never directly accessible to a process executing in that domain. Rather, the capability list is itself a protected object, maintained by the operating system and accessed by the user only indirectly.

**the operating mechanism :**

- Lock-key scheme :** The lock-key scheme is a compromise between access lists and capability lists.
- Each object has a list of unique bit patterns, called locks.
- Similarly, each domain has a list of unique bit patterns, called keys.
- A process executing in a domain can access an object only if that domain has a key that matches one of the locks of the object.
- As with capability lists, the list of keys for a domain must be managed by the operating system on behalf of the domain.
- Users are not allowed to examine or modify the list of keys (or locks) directly.

**Ques 5.27. Explain the need of system protection with the help of examples.**

**AKTU 2013-14, Marks 05**

**Answer**

**The need of system protection is due to following reason :**

- To prevent malicious misuse of the system by users or programs.
- To ensure that each shared resource is used only in accordance with system policies, which may be set either by system designers or by system administrators.
- To ensure that errant programs cause the minimal amount of damage possible.
- Protection systems only provide the mechanisms for enforcing policies and ensuring reliable systems. It is up to administrators and users to implement those mechanisms effectively.

**An example : UNIX**

UNIX associates domains with users.

- Certain programs operate with the SUID bit set, which effectively changes the user ID, and therefore the access domain, while the program is running (and similarly for the SGID bit). Unfortunately this has some potential for abuse.
- An alternative used on some systems is to place privileged programs in special directories, so that they attain the identity of the directory owner when they run.
- This prevents crackers from placing SUID programs in random directories around the system.
- Yet another alternative is to not allow the changing of ID at all. Instead, special privileged daemons are launched at boot time, and user processes send messages to these daemons when they need special tasks performed.

**VERY IMPORTANT QUESTIONS**

*Following questions are very important. These questions may be asked in your SESSIONALS as well as UNIVERSITY EXAMINATION.*

**Q. 1. Define I/O devices. Describe the techniques for performing I/O.**

■■■ Refer Q. 5.1 and Q. 5.2.

**Q. 2. Explain I/O buffering with its types.**

■■■ Refer Q. 5.3 and Q. 5.4.

**Q. 3. Discuss disk scheduling algorithm with example.**

■■■ Refer Q. 5.6.

**Q. 4. Discuss the following terms :**

i. Caching                          iv. Boot block

ii. Spooling                        v. Bit vector

iii. Error handling              vi. Counting

■■■ Refer Q. 5.5, Q. 5.11 and Q. 5.12.

**Q. 5. Explain RAID.**

■■■ Refer Q. 5.13.  
**Q. 6. Explain file organization and access mechanism.**  
 ■■■ Refer Q. 5.16.  
**Q. 7. Define directory. Define different ways to implement the directory.**  
 ■■■ Refer Q. 5.17.

**Q. 8. Write short note on :**

i. Sequential file  
 ii. Indexed file

■■■ Refer Q. 5.22.

**Q. 9. Define access matrix. Define implementation and representation of access matrix.**

■■■ Refer Q. 5.25 and Q. 5.26.

©©©



## Computer System (2 Marks Questions)

### Memory Based Questions

**1. Define operating system. Explain in short.**

**AKTU 2010-11, Matric 02**

An operating system acts as an intermediary between the user of a computer and computer hardware.

**2. What is the purpose of an operating system ?**

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

**3. What are advantages of batch system ?**

Advantages of batch system are :

- Move much of the work of the operator to the computer.
- Increased performance since it is possible for job to start as soon as the previous job finished.

**4. What are the disadvantages of batch system ?**

Disadvantages of batch system :

- Turnaround time can be large from user standpoint.
- Difficult to debug program.
- A job could enter an infinite loop.
- A job could corrupt the monitor, thus affecting pending jobs.
- Due to lack of protection scheme, one batch job can affect pending jobs.

**15. What is spooling ?**

■■■ Spooling refers to putting jobs in a buffer which is a special area in memory or on a disk where a device can access them when it is ready.

**AKTU 2010-11, Matric 02**