



K.S. INSTITUTE OF TECHNOLOGY, BENGALURU - 560109
FIRST INTERNAL TEST QUESTION PAPER 2023-24 ODD SEMESTER

SET: A

USN

--	--	--	--	--	--	--	--	--	--

Degree : B.E
 Branch - Stream : ~~C&E~~/CSE
 Course Title : Operating System
 Duration : 1 ½ Hr (90 minutes)

Semester : IIIrd
 Course Type / Code : BCS303
 Date :
 Max Marks : 50

Note: Answer **ONE full** question from each Module.

K-Levels: K1-Remembering, K2-Understanding, K3-Applying, K4-Analyzing, K5-Evaluating, K6-Creating

Q No.	Questions	Marks	CO	K-Level			
Module 1							
1(a)	Define Operating system and Explain dual mode of operating system with a neat diagram	5	CO1	K2			
(b)	Identify the difference between Multiprogramming and Timesharing system	5	CO1	K3			
(c)	Design a Single-Processor System, Multiprocessor System, Clustered System in detail with neat diagrams.	10	CO1	K3			
OR							
2(a)	Discuss about a system program and its categories?	5	CO1	K2			
(b)	Design a Virtual machine in Java Platform with neat diagram?	5	CO1	K3			
(c)	Design and explain about operating system structure in detail?	10	CO1	K3			
Module 2							
3(a)	Utilize a process and Determine the different states of a process with an example and its process control block(PCB)?	5	CO2	K3			
(b)	Identify the purpose of Short-term Scheduler, Long-term Scheduler, Medium-term Scheduler	5	CO2	K3			
(c)	Apply IPC mechanism in message passing systems and explain it in detail.	10	CO2	K3			
OR							
4(a)	Identify the difference between Single Threaded and Multithreaded process with neat diagram and example?	5	CO2	K3			
(b)	Identify the benefits of multithreaded programming?	5	CO2	K3			
(c)	Solve the average waiting time and average turnaround time by drawing the Gantt Chart using FCFS, SRTF, RR(q=2ms) and priority algorithms. Lower priority number represents higher priority :		10	CO2	K3		
	Process	Arrival Time				Burst Time	Priority
	P1	0				9	3
	P2	1				4	2
	P3	2				9	1
P4	3	5	4				
Module 3							
5(a)	Build Pseudocode for the Producer process and Consumer process?	5	CO3	K3			

(b)	Identify a scenario for critical section problem? Explain its hardware based solution for the critical section problem?	5	CO3	K2
OR				
6(a)	Make use of Test And Set() that Mutual-Exclusion is preserved	5	CO3	K3
(b)	Discuss the conditions where deadlock occurs	5	CO3	K2

T. Naga Jyothi



Name & Signature of
Course In charge:

Dr. Chandan V Reddy



Name & Signature of
Module Coordinator:



HOD



Principal

Seetha



K.S. INSTITUTE OF TECHNOLOGY, BENGALURU - 560109
FIRST INTERNAL TEST QUESTION PAPER 2023-24 ODD SEMESTER

SET: B

USN

--	--	--	--	--	--	--	--	--	--

Degree : B.E
 Branch - Stream : **CBE / CSE**
 Course Title : **Operating System**
 Duration : **1 ½ Hr (90 minutes)**

Semester :3rd
 Course Type / Code :BCS303
 Date :3/1 2024
 Max Marks :50


Note: Answer **ONE full** question from each Module.

K-Levels: K1-Remebering, K2-Understanding, K3-Applying, K4-Analyzing, K5-Evaluating, K6-Creating

Q No.	Questions	Marks	CO	K-Level															
Module 1																			
1(a)	Explain the role of operating system with user and system viewpoints.	5	CO1	K2															
(b)	Identify the goals while designing an Operating system and Differentiate between Mechanism and Policies	5	CO1	K3															
(c)	(i) Identify the differences between Symmetric and Asymmetric Multiprocessing (ii) Distinguish between modular kernel approach and layered approach with a neat sketch	10	CO1	K3															
OR																			
2(a)	Explain about Storage structure and its hierarchy in detail with neat diagram.	5	CO1	K2															
(b)	Compare and contrast between client server computing and peer-to-peer computing	5	CO1	K3															
(c)	Using neat diagram, explain the concept of virtual machines	10	CO1	K3															
Module 2																			
3(a)	Define Cooperating process. Identify the reasons for process cooperation.	5	CO2	K3															
3(b)	Distinguish between direct and indirect communication	5	CO2	K3															
3(c)	Consider the following set of processes with CPU burst time(in ms)	10	CO2	K3															
	<table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="width: 30%;">Process</th> <th style="width: 30%;">Arrival Time</th> <th style="width: 30%;">Burst Time</th> </tr> </thead> <tbody> <tr><td>P1</td><td align="center">0</td><td align="center">6</td></tr> <tr><td>P2</td><td align="center">1</td><td align="center">3</td></tr> <tr><td>P3</td><td align="center">2</td><td align="center">1</td></tr> <tr><td>P4</td><td align="center">3</td><td align="center">4</td></tr> </tbody> </table>				Process	Arrival Time	Burst Time	P1	0	6	P2	1	3	P3	2	1	P4	3	4
	Process				Arrival Time	Burst Time													
	P1				0	6													
	P2				1	3													
P3	2	1																	
P4	3	4																	
Calculate average waiting time and average turnaround time for the above process using FCFS, SRTF and RR scheduling algorithm (time quantum=2ms). Represent it in Gantt chart.																			
OR																			
4(a)	Determine the purpose of Scheduler? Identify the scheduling criteria	5	CO2	K3															
(b)	Identify the challenges faced while implementing multicore systems	5	CO2	K3															
(c)	Build Multithreading Models and explain them in detail	10	CO2	K3															

Module 3

5(a)	Identify the purpose of Process Synchronization and Apply the approaches used to handle critical sections in OS	5	CO3	K3
(b)	Show how semaphores provide solution to implement mutual exclusion?	5	CO3	K2
OR				
	Design a Software based solution for Critical section problem and prove that mutual exclusion property is preserved.	5	CO3	K3
(b)	Show a scenario where Deadlock occurs with example.	5	CO3	K2

T. Naga Jyothi

 Name & Signature of
 Course In charge:

Do Chandra V Reddy

 Name & Signature of
 Module Coordinator:


 HOD


 Principal



K.S. INSTITUTE OF TECHNOLOGY, BENGALURU - 560109
SECOND INTERNAL TEST QUESTION PAPER 2023-24 ODD SEMESTER

SET: A

USN

--	--	--	--	--	--	--	--	--	--

Degree : B.E
 Branch - Stream : CCE
 Course Title : OPERATING SYSTEM
 Duration : 1 Hr (60 minutes)

Semester : III
 Course Type / Code : BCS303
 Date : 6-2-2024
 Max Marks : 25

Note: Answer **ONE full** question from each Module.

K-Levels: K1-Remebering, K2-Understanding, K3-Applying, K4-Analyzing, K5-Evaluating, K6-Creating

Q No.	Questions	Marks	CO	K-Level																																																																					
Module 3																																																																									
1(a)	Experiment with different methods to recovery from deadlocks and Build a resource allocation graph by experimenting with Processes P0 &P1 and two resources R1 and R2	10	CO3	K3																																																																					
OR																																																																									
2(a)	<p>Apply Banker's algorithm determine whether the following system is in a safe state.</p> <table border="1" style="margin: 10px auto; border-collapse: collapse;"> <thead> <tr> <th rowspan="2">Processes</th> <th colspan="3">Allocation</th> <th colspan="3">Max</th> <th colspan="3">Available</th> </tr> <tr> <th>A</th> <th>B</th> <th>C</th> <th>A</th> <th>B</th> <th>C</th> <th>A</th> <th>B</th> <th>C</th> </tr> </thead> <tbody> <tr> <td>P0</td> <td align="center">0</td> <td align="center">0</td> <td align="center">2</td> <td align="center">0</td> <td align="center">0</td> <td align="center">4</td> <td align="center">1</td> <td align="center">0</td> <td align="center">2</td> </tr> <tr> <td>P1</td> <td align="center">1</td> <td align="center">0</td> <td align="center">0</td> <td align="center">2</td> <td align="center">0</td> <td align="center">1</td> <td></td> <td></td> <td></td> </tr> <tr> <td>P2</td> <td align="center">1</td> <td align="center">3</td> <td align="center">5</td> <td align="center">1</td> <td align="center">3</td> <td align="center">7</td> <td></td> <td></td> <td></td> </tr> <tr> <td>P3</td> <td align="center">6</td> <td align="center">3</td> <td align="center">2</td> <td align="center">8</td> <td align="center">4</td> <td align="center">2</td> <td></td> <td></td> <td></td> </tr> <tr> <td>P4</td> <td align="center">1</td> <td align="center">4</td> <td align="center">3</td> <td align="center">1</td> <td align="center">5</td> <td align="center">7</td> <td></td> <td></td> <td></td> </tr> </tbody> </table> <p>Identify the safe sequence and Choose a request from Process P2 arrives for (0,0,2) can the request be granted immediately</p>	Processes	Allocation			Max			Available			A	B	C	A	B	C	A	B	C	P0	0	0	2	0	0	4	1	0	2	P1	1	0	0	2	0	1				P2	1	3	5	1	3	7				P3	6	3	2	8	4	2				P4	1	4	3	1	5	7				10	CO3	K3
Processes	Allocation			Max			Available																																																																		
	A	B	C	A	B	C	A	B	C																																																																
P0	0	0	2	0	0	4	1	0	2																																																																
P1	1	0	0	2	0	1																																																																			
P2	1	3	5	1	3	7																																																																			
P3	6	3	2	8	4	2																																																																			
P4	1	4	3	1	5	7																																																																			
Module 4																																																																									
3(a)	Build a model for multi-step processing of a user program and explain it in detail?	5	CO4	K3																																																																					
(b)	Make use of swapping, Construct a model for swapping of two processes using a disk as a backing store and explain its disadvantages	10	CO4	K3																																																																					
OR																																																																									
4(a)	Build a model with hardware address protection with base and limit registers and Explain the necessity of base and limit registers.	5	CO4	K3																																																																					
(b)	Apply paging hardware with TLB and explain it in detail with a neat block diagram.	10	CO4	K3																																																																					

A
 Name & Signature of
 Course In charge:

Do Chandra V Reddy
RS
 Name & Signature of
 Module Coordinator:

RS
 HOD

R. Kumar
 Principal

Select



K.S. INSTITUTE OF TECHNOLOGY, BENGALURU - 560109
SECOND INTERNAL TEST QUESTION PAPER 2023-24 ODD SEMESTER

SET: B

Degree : B.E
 Branch - Stream : CCE
 Course Title : OPERATING SYSTEM
 Duration : 1 Hr (60 minutes)

USN

--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--

Semester : III
 Course Type / Code : BCS303
 Date : 6-2-2024
 Max Marks : 25

Note: Answer ONE full question from each Module.

K-Levels: K1-Remebering, K2-Understanding, K3-Applying, K4-Analyzing, K5-Evaluating, K6-Creating

Q No.	Questions	Marks	CO	K-Level																																																																					
Module 3																																																																									
1(a)	Make Use of deadlock prevention technique and explain them in detail.	10	CO3	K3																																																																					
OR																																																																									
2(a)	<p>Apply Banker's algorithm to determine whether the following system is in a safe state.</p> <table border="1" style="width: 100%; border-collapse: collapse; margin: 10px 0;"> <thead> <tr> <th rowspan="2">Processes</th> <th colspan="3">Allocation</th> <th colspan="3">Max</th> <th colspan="3">Available</th> </tr> <tr> <th>A</th> <th>B</th> <th>C</th> <th>A</th> <th>B</th> <th>C</th> <th>A</th> <th>B</th> <th>C</th> </tr> </thead> <tbody> <tr> <td>P0</td> <td align="center">0</td> <td align="center">1</td> <td align="center">0</td> <td align="center">7</td> <td align="center">5</td> <td align="center">3</td> <td align="center">3</td> <td align="center">3</td> <td align="center">2</td> </tr> <tr> <td>P1</td> <td align="center">2</td> <td align="center">0</td> <td align="center">0</td> <td align="center">3</td> <td align="center">2</td> <td align="center">2</td> <td> </td> <td> </td> <td> </td> </tr> <tr> <td>P2</td> <td align="center">3</td> <td align="center">0</td> <td align="center">2</td> <td align="center">9</td> <td align="center">0</td> <td align="center">2</td> <td> </td> <td> </td> <td> </td> </tr> <tr> <td>P3</td> <td align="center">2</td> <td align="center">1</td> <td align="center">1</td> <td align="center">2</td> <td align="center">2</td> <td align="center">2</td> <td> </td> <td> </td> <td> </td> </tr> <tr> <td>P4</td> <td align="center">0</td> <td align="center">0</td> <td align="center">2</td> <td align="center">4</td> <td align="center">3</td> <td align="center">3</td> <td> </td> <td> </td> <td> </td> </tr> </tbody> </table> <p>Identify the need matrix and calculate the safe sequence</p>	Processes	Allocation			Max			Available			A	B	C	A	B	C	A	B	C	P0	0	1	0	7	5	3	3	3	2	P1	2	0	0	3	2	2				P2	3	0	2	9	0	2				P3	2	1	1	2	2	2				P4	0	0	2	4	3	3				10	CO3	K3
Processes	Allocation			Max			Available																																																																		
	A	B	C	A	B	C	A	B	C																																																																
P0	0	1	0	7	5	3	3	3	2																																																																
P1	2	0	0	3	2	2																																																																			
P2	3	0	2	9	0	2																																																																			
P3	2	1	1	2	2	2																																																																			
P4	0	0	2	4	3	3																																																																			
Module 4																																																																									
3(a)	Apply First fit, Best fit, Worst fit for the given memory partitions of 100K,500K,200K,300K and 600K to place 212K,417K,112K and 426K	5	CO4	K3																																																																					
(b)	Make use of structure of page table with suitable diagrams and explain it in detail.	10	CO4	K3																																																																					
OR																																																																									
4(a)	Identify the difference between Internal and External Fragmentation	5	CO4	K3																																																																					
(b)	Identify the usage of contiguous memory allocation and explain it with suitable diagram	10	CO4	K3																																																																					

Name & Signature of Course In charge:

Name & Signature of Module Coordinator:

HOD

Principal



K S INSTITUTE OF TECHNOLOGY

Bangalore – 560109

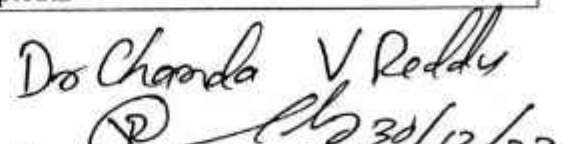
DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

CIE Question paper Scrutiny format

Course Name	OPERATING SYSTEM
Course Code	BCS303
Course Incharge	Mrs.T.Naga Jyothi
Academic year	2023-2024
Semester	3 rd
CIE #	IA – 1
Set	A <input checked="" type="checkbox"/> B <input type="checkbox"/>
Scrutiny parameters	
Whether questions are according to assessment plan?	Yes <input checked="" type="checkbox"/> No <input type="checkbox"/> ; If No, Suggestions:
Whether questions prepared are within the covered syllabus?	Yes <input checked="" type="checkbox"/> No <input type="checkbox"/> ; If No, Suggestions:
Whether all questions are mapped to CO/PO properly?	Yes <input checked="" type="checkbox"/> No <input type="checkbox"/> ; If No, Suggestions:
Whether questions framed are according to Blooms level?	Yes <input checked="" type="checkbox"/> No <input type="checkbox"/> ; If No, Suggestions:
Whether marks distribution for each question are correct?	Yes <input checked="" type="checkbox"/> No <input type="checkbox"/> ; If No, Suggestions:
Whether questions paper follows the format displayed?	Yes <input checked="" type="checkbox"/> No <input type="checkbox"/> ; If No, Suggestions:
Difficulty level	Very High <input type="checkbox"/> High <input checked="" type="checkbox"/> Moderate <input checked="" type="checkbox"/> Low <input type="checkbox"/>
Percentage of Similarity questions in Set A & B	10
Final decision	Accepted without corrections <input type="checkbox"/> Accepted with minor corrections <input type="checkbox"/> Not accepted <input type="checkbox"/>


30/12/23

Signature with date
of CIE Question paper setter


30/12/23
Name and Signature with date
of CIE Question paper Scrutiniser

Scheme of Evaluation-SET A

1. Define an Operating system. Explain dual mode of operation with a diagram? [5M]

Ans:

- An operating system is a program that manages computer hardware.
- It acts as an intermediary between computer user and computer hardware.

Dual mode of operation

(i) To ensure proper execution of the operating system, It is required to distinguish between the execution of operating system code and user – defined code.

(ii) The approach by most computer systems is to provide hardware support that allows to differentiate among various modes of execution.

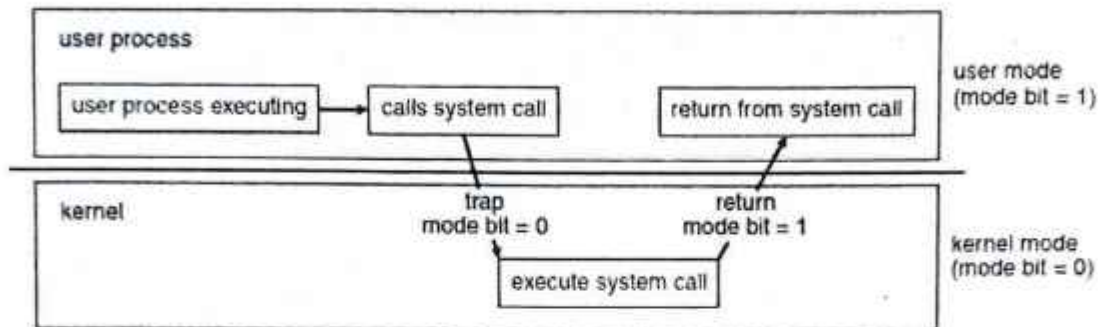


Figure 1.10 Transition from user to kernel mode.

(iii) There are two modes of execution namely **user mode and kernel mode**.

A bit(mode bit) is added to the hardware of the computer to indicate the current mode i.e Kernel(0) and User(1).

(ii) When the computer executed on behalf of the user application, The system is in user mode.

(iii) When a user requests a service from Operating system, The system must transition from user to kernel mode to fulfill the request.

Initially At system boot time, The hardware starts in kernel mode

The operating system is loaded and starts user application in user mode.

Whenever an interrupt occurs, The hardware switches from user mode to kernel mode.

Whenever operating system gains control of the computer, it is in kernel mode.

Advantage:

1. It protects the operating system from errant users and errant user from one another.
2. Hardware allows privileged instructions to be executed in kernel mode.
3. Privileged instructions are executed in kernel mode.

Ex: I/O control, timer management, interrupt management.

2. Identify the difference between Multiprogramming and timesharing system? (5M)

Ans:

Multiprogramming System	Timesharing System
1. Multiprogramming system increases CPU utilization by organizing jobs so that CPU has always one to execute	1. Time sharing system is a logical extension of multiprogramming in which the CPU executes multiple jobs by switching among them, but the switches occur so frequently that the users interact with each program while it is running.
2. The operating system keeps	2. The CPU executes multiple jobs by

(1+2+2)

several jobs in memory simultaneously. As main memory is small to accommodate all jobs, the jobs are kept on the disk in the job pool.	switching among them but the switches occur so frequently that the user can interact with each program while it is running.
3. No communication between the user and the system	3. Direct communication between the user and the system
4. No user interaction	4. User interaction

3. Design a single processor systems, Multiprocessor systems, Clustered systems in detail with a neat diagram? (10M)

(2+4+4M)

Single processor system:

1. There is one main CPU capable of executing a general purpose instruction set, including instructions from user processes.
2. Single processor systems have special purpose processors as well. These processors are the disk, keyboard and graphic controllers.
3. Special purpose processors run on a limited instruction set and do not run on user processes.

Multi processor system:

1. Multiprocessor system are known as parallel systems (have two or more processors in communication) sharing the common bus, clock, memory and peripheral devices.
2. Multiprocessors have three main advantages.

(i) Increased Throughput

- By increasing the number of processors, more work is done in less time.
- The speed-up ratio with N processors is not N but it is less than N.
- When multiple processors cooperate on a task, a certain amount of overhead is incurred in keeping all the parts working correctly.

(ii) Economy of scale

- Multiprocessor system can cost less than equivalent multiple single-processor systems because they share peripherals, mass storage and power supplies.

(iii) Increased Reliability:

If functions are distributed properly among several processors then the failure of one processor will not halt the system, only it slows down.

The ability to continue providing service proportional to the level of surviving hardware is called graceful degradation.

Multiprocessor systems are categorized in to two types----

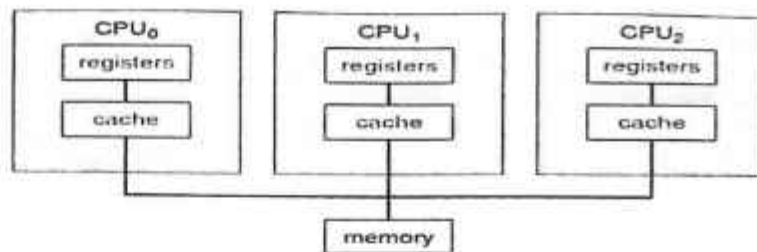
1. Symmetric 2. Assymmetric

Symmetric:

Each and every processor performs all the tasks with in the Operating system

All processors are considered as Peers.

Ex: AIX system designed by IBM



Symmetric multiprocessor architecture

Each and every processor has its own registers and cache

Benefits

1. Many processes can run simultaneously without causing performance deterioration.

2. Asymmetric:

Each processor is assigned a specific task i.e boss processor controls the system and all other processor look to the boss for instruction.

The difference between symmetric and asymmetric may result from h/w or s/w.

Clustered system:

1. Clustered system gathers multiple CPU's which are loosely coupled.
2. Clustered computers share storage and closely linked via LAN.
3. Clustering is used to provide high-availability service i.e., service will continue even if one or more systems in the cluster fail.
4. Clustering can be done symmetrically or asymmetrically.
5. In asymmetric clustering, one machine is in hot-standby mode while other are running applications.
6. hotstand by mode does nothing but monitor the active server. If the server fails, hotstand by mode host becomes active server.
7. If the server fails, the hot stand by host becomes the active server.
8. Two types of clustering (i) Symmetric (Peer-Peer) Clustering
(ii) Asymmetric clustering

Symmetric clustering:

Two or more hosts are running applications and monitoring each other.

Asymmetric Clustering:

One machine is in hot-standby mode while other is running the applications.

It does nothing and monitors the active server. If the server fails, the hot stand by host becomes the active server.

9. Clustering can be implemented via LAN, MAN, WAN, SAN

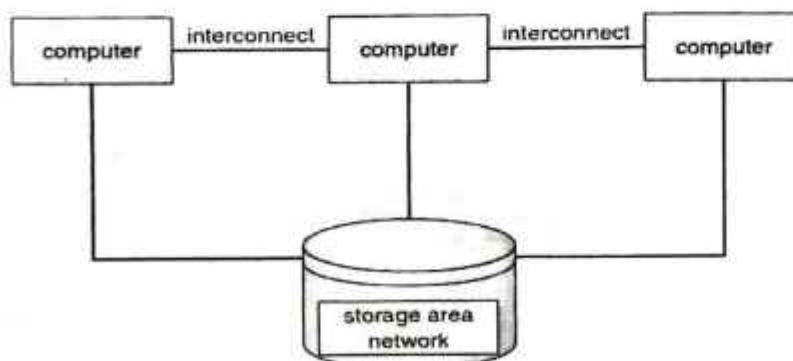


Figure 1.8 General structure of a clustered system.

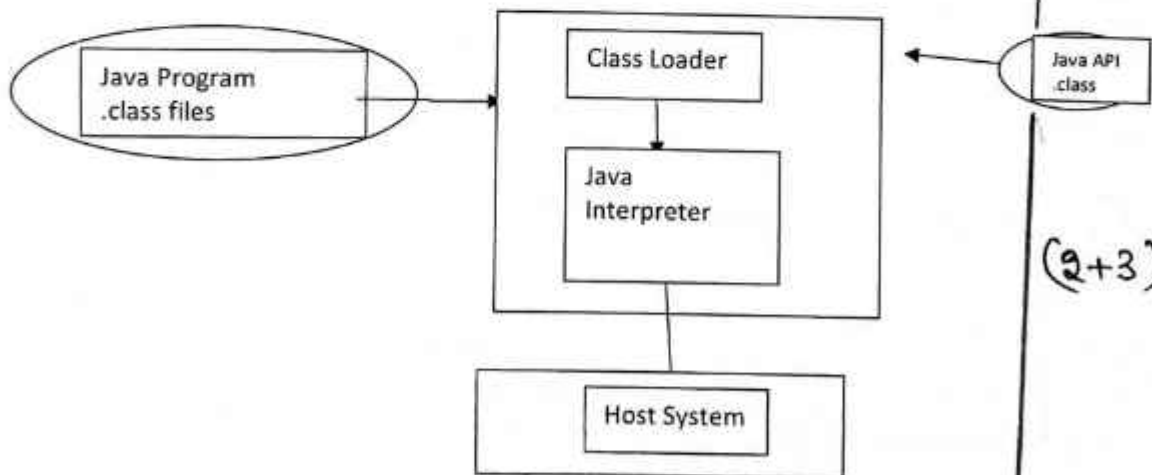
2(a) Discuss about system program and its categories? (5M)

Ans: System programs known as system utilities, provides a convenient environment for program development and execution. System programs are divided in to the following

- **File management.** These programs create, delete, copy, rename, print, dump, list, and generally manipulate files and directories.
- **Status information.** Some programs simply ask the system for the date, time, amount of available memory or disk space, number of users, or similar status information. Others are more complex, providing detailed performance, logging, and debugging information. Typically, these programs format and print the output to the terminal or other output devices or files or display it in a window of the GUI. Some systems also support a registry, which is used to store and retrieve configuration information.
- **File modification.** Several text editors may be available to create and modify the content of files stored on disk or other storage devices. There may also be special commands to search contents of files or perform transformations of the text.
- **Programming-language support.** Compilers, assemblers, debuggers, and interpreters for common programming languages (such as C, C++, Java, and PERL) are often provided with the operating system or available as a separate download.
- **Program loading and execution.** Once a program is assembled or compiled, it must be loaded into memory to be executed. The system may provide absolute loaders, relocatable loaders, linkage editors, and overlay loaders. Debugging systems for either higher-level languages or machine language are needed as well.
- **Communications.** These programs provide the mechanism for creating virtual connections among processes, users, and computer systems. They allow users to send messages to one another's screens, to browse Web pages, to send e-mail messages, to log in remotely, or to transfer files from one machine to another.
- **Background services.** All general-purpose systems have methods for launching certain system-program processes at boot time. Some of these processes terminate after completing their tasks, while others continue to run until the system is halted. Constantly running system-program processes are known as services, subsystems, or daemons. One example is the network daemon discussed in Section 2.4.5. In that example, a system needed a service to listen for network connections in order to connect those requests to the correct processes. Other examples include process schedulers that start processes according to a specified schedule, system error monitoring services, and print servers. Typical systems have dozens

(1+4)M

2(b) Design a virtual machine in Java platform with neat diagram?(5M)



(2+3)M

- Java is a popular object oriented programming language introduced by sun-microsystems in 1995.
- Java objects are specified with the class construct.
- Java program consists of one or more classes.
- JVM is specification for an abstract computer.

- It consists of class loader and Java Interpreter that executes the architectural neural bytecodes.
- The class loader loads the compiled .class files from both Java program and Java API for execution by the Java interpreter.
- After a class is loaded, the verifier checks that the .class file is a valid Java bytecode and does not overflow or underflow the stack.
- If the class passes verification, it is run by Java Interpreter.
- JVM automatically manages memory by performing garbage collection.
- JVM may be implemented in software on top of host OS.
- A **Faster software technique (JIT)** is used. The first time when Java method is invoked, the byte codes for the method are turned into native machine language for the host system.

2(c) Design and explain about Operating system structure in detail? -- (10M)

(2+3+3+2)

Ans:

Operating-System Structure

Simple Structure

- Many operating systems do not have well-defined structures. They started as small, simple, and limited systems and then grew beyond their original scope. Eg: MS-DOS.
- In MS-DOS, the interfaces and levels of functionality are not well separated. Application programs can access basic I/O routines to write directly to the display and disk drives. Such freedom leaves MS-DOS in bad state and the entire system can crash down when user programs fail.
- UNIX OS consists of two separable parts: the kernel and the system programs. The kernel is further separated into a series of interfaces and device drivers. The kernel provides the file system, CPU scheduling, memory management, and other operating-system functions through system calls.

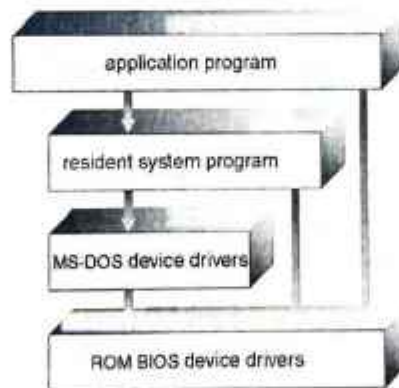


Figure: MS-DOS layer structure.

Layered Approach

- The OS is broken into number of layers (levels). Each layer rests on the layer below it, and relies on the services provided by the next lower layer.
- Bottom layer (layer 0) is the hardware and the topmost layer is the user interface.
- A typical layer, consists of data structure and routines that can be invoked by higher-level layer.
- Advantage of layered approach is simplicity of construction and debugging.
- The layers are selected so that each uses functions and services of only lower-level layers. So simplifies debugging and system verification. The layers are debugged one by one from the lowest and if any layer doesn't work, then error is due to that layer only, as the lower layers are already debugged. Thus, the design and implementation are simplified.
- A layer need not know how its lower-level layers are implemented. Thus hides the operations from higher layers.

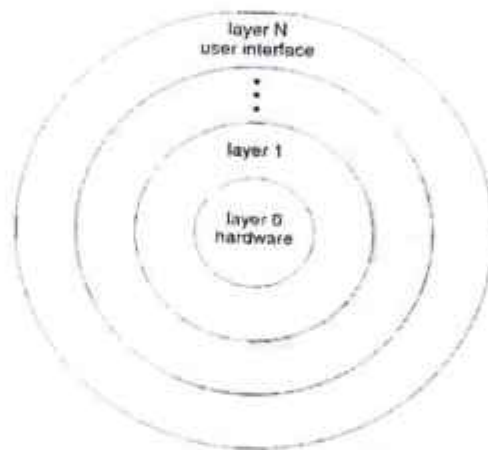


Figure: A layered Operating System

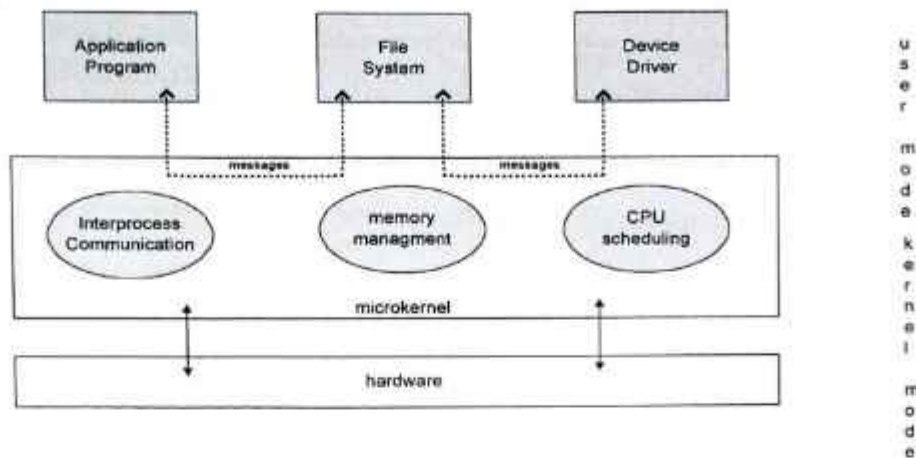
Disadvantages of layered approach:

- The various layers must be appropriately defined, as a layer can use only lower-level layers.
- Less efficient than other types, because any interaction with layer 0 required from top layer. The system call should pass through all the layers and finally to layer 0. This is an overhead.

Microkernels

- This method structures the operating system by removing all nonessential components from the kernel and implementing them as system and user-level programs thus making the kernel as small and efficient as possible.
- The removed services are implemented as system applications.
- Most microkernels provide basic process and memory management, and message passing between other services.

- The main function of the microkernel is to provide a communication facility between the client program and the various services that are also running in user space.



Benefit of microkernel –

- System expansion can also be easier, because it only involves adding more system applications, not rebuilding a new kernel.
- Mach was the first and most widely known microkernel, and now forms a major component of Mac OS X.

Disadvantage of Microkernel -

- Performance overhead of user space to kernel space communication

Modules

- Modern OS development is object-oriented, with a relatively small core kernel and a set of **modules** which can be linked in dynamically.
- Modules are similar to layers in that each subsystem has clearly defined tasks and interfaces, but any module is free to contact any other module, eliminating the problems of going through multiple intermediary layers.
- The kernel is relatively small in this architecture, similar to microkernels, but the kernel does not have to implement message passing since modules are free to contact each other directly. Eg: Solaris, Linux and MacOSX.

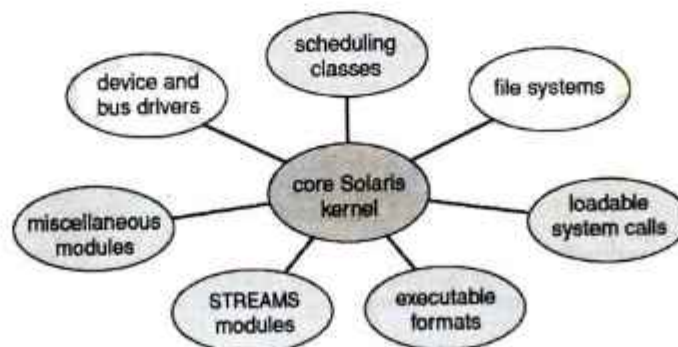


Figure: Solaris loadable modules

- The Max OS architecture relies on the Mach microkernel for basic system management services, and the BSD kernel for additional services. Application services and dynamically loadable modules (kernel extensions) provide the rest of the OS functionality.
- Resembles layered system, but a module can call any other module.
- Resembles microkernel, the primary module has only core functions and the knowledge of how to load and communicate with other modules.

Module 2

3(a) Utilize a process and Determine the different states of a process and its **Process control block?** (5M)

Ans:

Process means program in execution.

Ex : Opening a word document is an example of a process.

The following are the below mentioned states of a process.

New: The Process is being created.

Ex: Creating a new word document.

Running: Instructions are being executed.

Ex: Writing data on a document

Waiting: The processes are waiting for some event to occur.

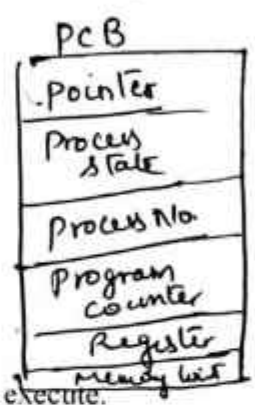
Ex: Microsoft Word document is waiting for a printer

Ready: The process is waiting to be assigned to a processor.

Ex: Word processor document acquired all resources and ready to execute.

Terminated: The process has finished its execution.

Ex: Word processor terminates after writing data on it.



(1+3M)
+1

3(b) Identify the purpose of short term scheduler, Long term scheduler, Medium-term scheduler? (5M)

Short term scheduler:

1. Short term scheduler selects the process among the list of processes that are ready to execute and allocates the CPU to one of them.

2. Short term scheduler executes at least once every 100 milliseconds.

Long term scheduler :

1. Long term scheduler or job scheduler selects processes from the job pool and loads them in to memory for execution.

2. Long term scheduler controls the degree of multiprogramming.

Medium-term scheduler:

(1.5+1.5+2M) M

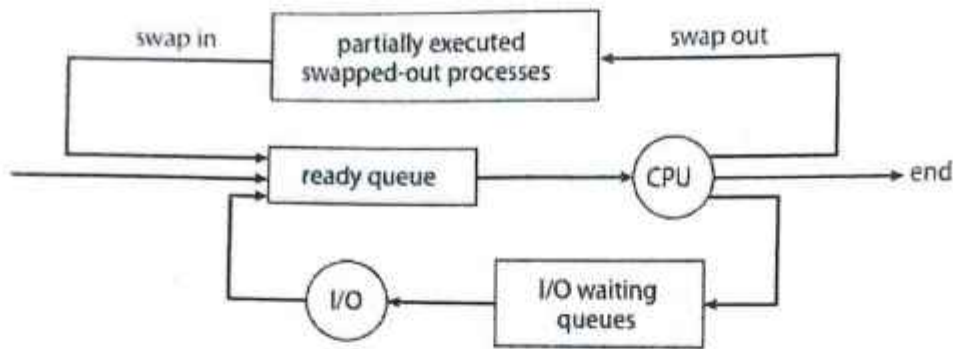


Figure 3.7 Addition of medium-term scheduling to the queuing diagram.

1. The medium term scheduler is used to remove a process from memory (active contention of CPU) and thus reduce the degree of multiprogramming.
2. Later the process can be reintroduced in to memory and the execution can be continued from where it left off. This scheme is referred as **swapping**.
3. The process is swapped in and swapped out to and from the memory is done by medium term scheduler.

3. Apply IPC mechanism in message passing systems and explain it in detail? (10 Marks)

Ans:

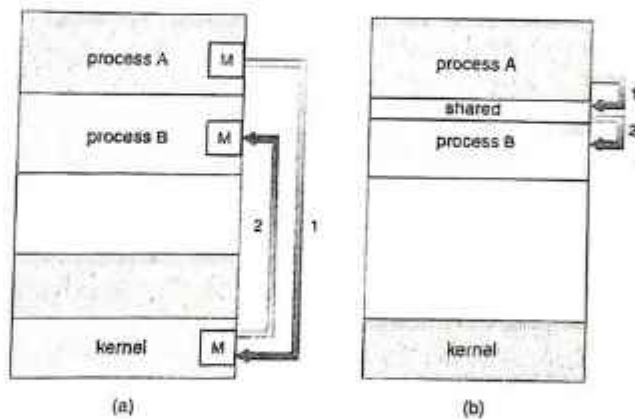


Figure 3.13 Communications models. (a) Message passing. (b) Shared memory.

Message Passing requires system calls for every message transfer, and is therefore slower, but it is simpler to set up and works well across multiple computers. Message passing is generally preferable when the amount and/or frequency of data transfers is small.

A mechanism to allow process communication without sharing address space. It is used in distributed systems.

- Message passing systems uses system calls for "send message" and "receive message".
- A communication link must be established between the cooperating processes before messages can be sent.
- There are three methods of creating the link between the sender and the receiver

(2+8M)

- Direct or indirect communication (naming)
- Synchronous or asynchronous communication (Synchronization)
- Automatic or explicit buffering.

1. Naming

Processes that want to communicate must have a way to refer to each other. They can use either direct or indirect communication.

a) Direct communication the sender and receiver must explicitly know each other's name. The syntax for `send()` and `receive()` functions are as follows-

- `send(P, message)` – send a message to process P
- `receive(Q, message)` – receive a message from process Q

Properties of communication link:

- A link is established automatically between every pair of processes that wants to communicate. The processes need to know only each other's identity to communicate.
- A link is associated with exactly one pair of communicating processes
- Between each pair, there exists exactly one link.

Types of addressing in direct communication –

- Symmetric addressing – the above-described communication is symmetric communication. Here both the sender and the receiver processes have to name each other to communicate.
- Asymmetric addressing – Here only the sender's name is mentioned, but the receiving data can be from any system.

`send(P, message)` --- Send a message to process P

`receive(id, message)`. Receive a message from any process

Disadvantages of direct communication – any changes in the identifier of a process, may have to change the identifier in the whole system (sender and receiver), where the messages are sent and received.

b) Indirect communication uses shared mailboxes, or ports.

A mailbox or port is used to send and receive messages. Mailbox is an object into which messages can be sent and received. It has a unique ID. Using this identifier messages are sent and received.

Two processes can communicate only if they have a shared mailbox. The send and receive functions are –

- send (A, message) – send a message to mailbox A
- receive (A, message) – receive a message from mailbox A

Properties of communication link:

- A link is established between a pair of processes only if they have a shared mailbox. A link may be associated with more than two processes
- Between each pair of communicating processes, there may be any number of links, each link is associated with one mailbox.
- A mail box can be owned by the operating system. It must take steps to –
 - create a new mailbox
 - send and receive messages from mailbox
 - delete mailboxes.

2. Synchronization

The send and receive messages can be implemented as either **blocking** or **non-blocking**.

- Blocking (synchronous) send** - sending process is blocked (waits) until the message is received by receiving process or the mailbox.
- Non-blocking (asynchronous) send** - sends the message and continues (does not wait)
- Blocking (synchronous) receive** - The receiving process is blocked until a message is available
- Non-blocking (asynchronous) receive** - receives the message without block. The received message may be a valid message or null.

3. Buffering

When messages are passed, a temporary queue is created. Such queue can be of three capacities:

- Zero capacity** – The buffer size is zero (buffer does not exist). Messages are not stored in the queue. The senders must block until receivers accept the messages.
- Bounded capacity**- The queue is of fixed size(n). Senders must block if the queue is full. After sending 'n' bytes the sender is blocked.
- Unbounded capacity** –
The queue is of infinite capacity and the sender never blocks.

4(a)

Identify the difference between Single threaded and multithreaded process with neat diagram and example? (5M)

Ans:

- A thread is a basic unit of CPU utilization. It comprises of thread ID, a Program counter,

(2+3)M

a register set and a stack.

- It shares with other threads belonging to the same process its code-section & data-section.
- A traditional (or heavy weight) process has a single thread of control.
- If a process has multiple threads of control, it can perform more than one task at a time. Such a process is called **multi-threaded process** (Figure 2.1).

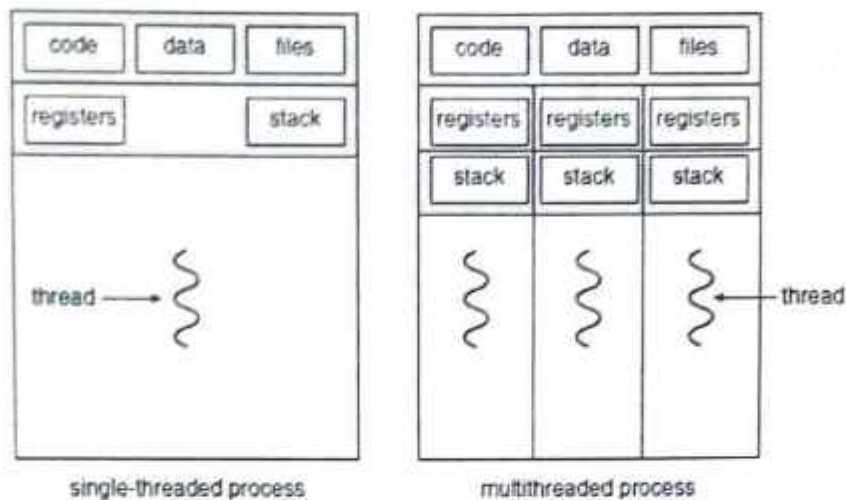


Figure 2.1 Single-threaded and multithreaded processes

Motivation

- 1) The software-packages that run on modern PCs are multithreaded.
An application is implemented as a separate process with several threads of control.
For ex: A word processor may have
 - first thread for displaying graphics
 - second thread for responding to keystrokes and
 - third thread for performing grammar checking.
- 2) In some situations, a single application may be required to perform several similar tasks.
For ex: A web-server may create a separate thread for each client request.
This allows the server to service several concurrent requests.
- 3) RPC servers are multithreaded.
When a server receives a message, it services the message using a separate thread.
This allows the server to service several concurrent requests.
- 4) Most OS kernels are multithreaded;
Several threads operate in kernel, and each thread performs a specific task, such as
 - managing devices or
 - interrupt handling.

4(b) Identify the benefits of Multithreaded Programming? (5M)

Ans:

The benefits of multithreaded programming can be broken down into four major categories:

1. **Responsiveness.** Multithreading an interactive application may allow a program to continue running even if part of it is blocked or is performing a lengthy operation, thereby increasing responsiveness to the user. This quality is especially useful in designing user interfaces. For instance, consider what happens when a user clicks a button that results in the performance of a time-consuming operation. A single-threaded application would be unresponsive to the user until the operation had completed. In contrast, if the time-consuming operation is performed in a separate thread, the application remains responsive to the user.
2. **Resource sharing.** Processes can only share resources through techniques such as shared memory and message passing. Such techniques must be explicitly arranged by the programmer. However, threads share the memory and the resources of the process to which they belong by default. The benefit of sharing code and data is that it allows an application to have several different threads of activity within the same address space.
3. **Economy.** Allocating memory and resources for process creation is costly. Because threads share the resources of the process to which they belong, it is more economical to create and context-switch threads. Empirically gauging the difference in overhead can be difficult, but in general it is significantly more time consuming to create and manage processes than threads. In Solaris, for example, creating a process is about thirty times

(5M)

4©

Solve the average waiting time and average turnaround time by drawing the Gantt Chart using FCFS, SRTF, RR($q=2\text{ms}$) and priority algorithms. Lower priority number represents higher priority :

Process	Arrival Time	Burst Time	Priority
P1	0	9	3
P2	1	4	2
P3	2	9	1
P4	3	5	4

(2+5+
2.5+
2.5+2.5)M

Ans:

FCFS Scheduling:**GANTT CHART:**

P1	P2	P3	P4
(0ms-----9ms)	(9ms-----13ms)	(13ms-----22ms)	(22ms-----27ms)

Turn Around Time=Completion time-Arrival Time

Processes	Turn Around time	Waiting time
-----------	------------------	--------------

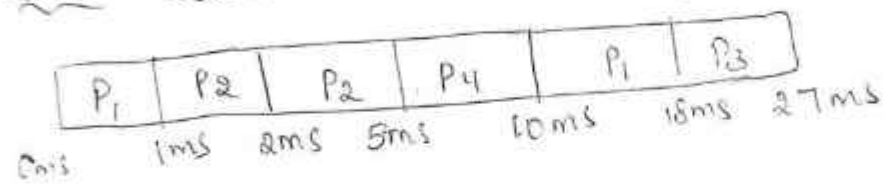
Robin

process	Turnaround time	waiting time
P1	9ms	0ms
P2	12ms	8ms
P3	20ms	11ms
P4	24ms	19ms

Average turnaround time = 16.25ms
 Average Waiting time = 9.5ms

SRTF (Shortest remaining time first)

GANTT CHART



P1 - 8ms
 P2 - 9
 P4 - 5

	<u>T.A.T</u>	<u>W.T</u>
P1	18ms	9ms
P2	4ms	0ms
P3	25ms	16ms
P4	7ms	2ms

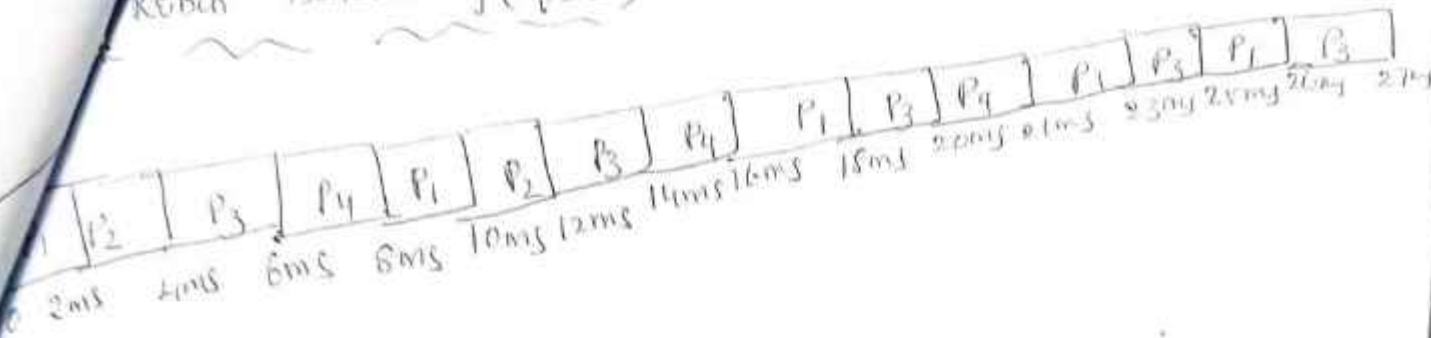
2.5M

Average Turnaround time = $\frac{18+4+25+7}{4} = \frac{54}{4} = 13.5$

Average waiting time = $\frac{9+0+16+2}{4} = \frac{27}{4} = 6.75$

25/6/25
 26/10/25
 10/10/25

Robin Scheduling (q=2)



Processes	T.A.T	W.T
P1	26ms	17ms
P2	11ms	7ms
P3	25ms	16ms
P4	18ms	13ms

$$\text{Waiting Time} = \text{T.A.T} - \text{B.T}$$

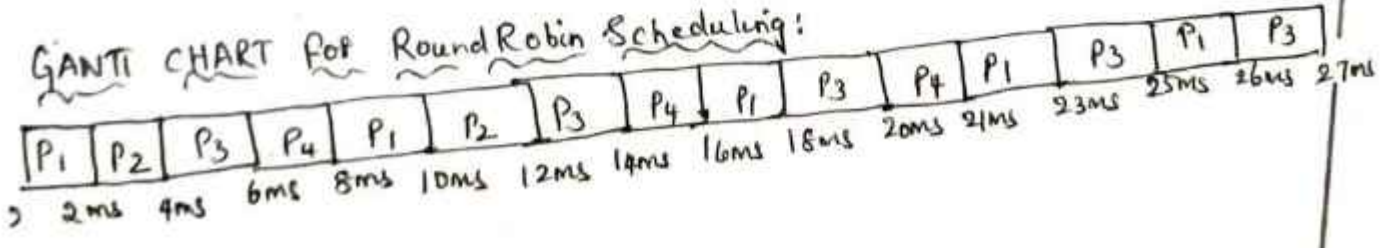
$$= \text{Turnaround Time} - \text{Burst Time}$$

2.57

$$\text{Average Turnaround Time} = \frac{(26 + 11 + 25 + 18)}{4} = \frac{80}{4} = 20$$

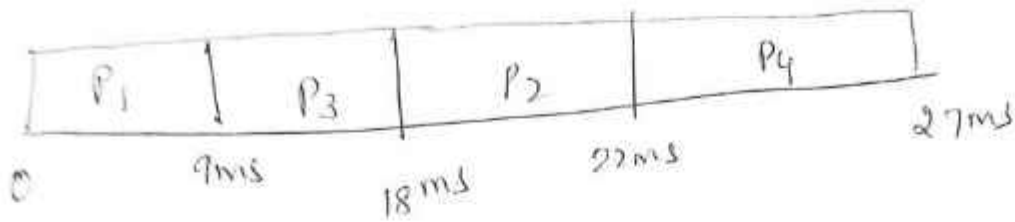
$$\text{Average Waiting Time} = \frac{(17 + 7 + 16 + 13)}{4} = 13.25 \text{ms}$$

GANTTI CHART FOR Round Robin Scheduling:



Priority scheduling algorithm

Gantt chart



<u>Process</u>	<u>T.A.T</u>	<u>W.T</u>
P ₁	9ms	0ms (9-9)
P ₂	21ms	(9ms - 4) = 18ms - 1ms = 17ms
P ₃	16ms	7ms (16-9) → (18-2-9)
P ₄	24ms	19ms (22-3) → (27-5-3) (2-5-1)
	<u>70ms</u>	<u>(0+17+7+19) = 43ms</u>

Average Turn around time = $\frac{70}{4} = 17.5ms$

Average Waiting time = $\frac{43ms}{4} = 10.75ms$

5(a) Build pseudocode for the Producer and Consumer process? (5M)

Producer Process:

```
While(true)
{
  while(counter==BUFFER_SIZE)
  {
    buffer[in]=next_produced
    in=(in+1)%BUFFER_SIZE;
    counter++;
  }
}
```

(2.5M)

Consumer Process:

```
While(true)
{
  while(counter==0)
  nextconsumed=buffer[out]
  out=out+1%BUFFER_SIZE
  counter--;
}
```

(2.5M)

5(b) Identify a scenario for critical section problem? Explain its hardware based solution for the critical section problem?

5M

Ans:

Critical-section problem:

Consider a system consisting of n processes $[P_0, P_1, P_2, \dots, P_{n-1}]$
Critical section is the segment of code in which a process may be

- Changing common variables
- Updating a variable
- Writing a file

Each process has a critical section in which shared data can be accessed.

Scenario:

If a process P_1 is modifying the value of a shared variable named "x" and P_2 is trying to read the value before completion of updating which results in

DATA INCONSISTENCY.

Hardware based solution for critical section problem:

(2+3M)

- A simple tool "lock" is being used.
- Race conditions are prevented by the following restrictions.
 1. process must acquire a lock before entering a critical section.
 2. Process must release a lock when it exits critical section.

```

do
{
    acquire lock
        critical section
    release lock
        remainder section
}

```

6(a) Prove that mutual exclusion is preserved with TestAndSet() (5M)

Ans:

```

do {
    while (test_and_set(&lock))
        ; /* do nothing */

        /* critical section */

    lock = false;

        /* remainder section */
} while (true);

```

(3+2M)

- Mutual exclusion is preserved by declaring a variable **lock**
- If the process P1 is willing to enter in its critical section, First P1 should acquire the lock by initializing to TRUE and it should release the lock before leaving the critical section such that no other process could enter in its critical section by checking variable **lock**.

6(b) Discuss the conditions where deadlock occurs?

Ans :

The following are the necessary conditions that should hold simultaneously for a deadlock.

1. Mutual Exclusion:

At least one resource must be held in a nonsharable mode ,that is only one process at a time can use the resource . If another process requests the resources, The requesting process must be delayed until the resource has been released.

(5M)

2. Hold and Wait:

A process must be holding at lease one resource and waiting to acquire additional resources that are currently being held by other processes.

3. No pre-emption:

Resources cannot be preempted ; that is a resource can be released only voluntarily by the process holding it, after that process has completed its task.

4. Circular Wait:

A set {P0,P1....Pn} of waiting processes must exist such that P0 is waiting for a resource held by P1,P1 is waiting for resource held by P2.....Pn-1.

A

Pls

COURSE INCHARGE

HOD



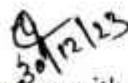
K S INSTITUTE OF TECHNOLOGY

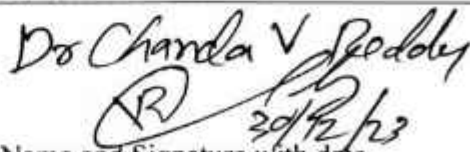
Bangalore – 560109

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

CIE Question paper Scrutiny format

Course Name	OPERATING SYSTEM
Course Code	BCS303
Course Incharge	Mrs.T.Naga Jyothi
Academic year	2023-2024
Semester	3 rd
CIE #	IA – 1
Set	A <input type="checkbox"/> B <input checked="" type="checkbox"/>
Scrutiny parameters	
Whether questions are according to assessment plan?	Yes <input checked="" type="checkbox"/> No <input type="checkbox"/> ; If No, Suggestions:
Whether questions prepared are within the covered syllabus?	Yes <input checked="" type="checkbox"/> No <input type="checkbox"/> ; If No, Suggestions:
Whether all questions are mapped to CO/PO properly?	Yes <input checked="" type="checkbox"/> No <input type="checkbox"/> ; If No, Suggestions:
Whether questions framed are according to Blooms level?	Yes <input checked="" type="checkbox"/> No <input type="checkbox"/> ; If No, Suggestions:
Whether marks distribution for each question are correct?	Yes <input checked="" type="checkbox"/> No <input type="checkbox"/> ; If No, Suggestions:
Whether questions paper follows the format displayed?	Yes <input checked="" type="checkbox"/> No <input type="checkbox"/> ; If No, Suggestions:
Difficulty level	Very High <input type="checkbox"/> High <input type="checkbox"/> Moderate <input checked="" type="checkbox"/> Low <input type="checkbox"/>
Percentage of Similarity questions in Set A & B	10%
Final decision	Accepted without corrections <input type="checkbox"/> Accepted with minor corrections <input type="checkbox"/> Not accepted <input type="checkbox"/>


Signature with date
of CIE Question paper setter


Name and Signature with date
of CIE Question paper Scrutiniser

Scheme of evaluation—SetB

1. Explain the role of operating system with user and system view points?(5M)

(4+1)

Ans:

Operating system controls the hardware and coordinates its use among various application programs for various users.

Os works like a government and it provides an environment with which other programs can do useful work

User View:

- The user's view of the computer varies according to the interface being used.
- Most of the user's view for a single computer → Computers (PC, Monitor, Keyboard, Mouse and system unit) → It is being designed for one user to utilise the resources and the goal is to maximize the work that the user is performing
- Operating system is designed for ease of use and how various hardware and software resources are shared.
- If the user sits at a terminal to a mainframe computer. Other users are accessing the same computer through other terminals. These users **share resources and exchange information.**
Os is designed for Resource Utilisation
- If the users sit at workstations connected to network of other workstations and servers then OS is designed for individual usability and resource utilization.
- In case of an embedded systems, Some computers have no user view at all.

System view:

- From computer point of view, Os acts like a resource allocator.

1(b). Identify the goals while designing an operating system and Differentiate between mechanism and policies? (5M)

Ans:

Goals:

There are divided in to two types of goals namely (i) User goals (ii) System goals.

User Goals:

1. Convenient to use
2. Easy to learn
3. Reliable
4. Safe
5. Fast

System goals:

1. Easy to design
2. Easy to create
3. Error free
4. Easy to implement and operate the system in an efficient manner
5. System should be flexible, reliable, error free and efficient.

Difference between Mechanism and Policy:

- Mechanism describes how to do something and Policy describes what will be done.
- The policy and mechanism should be separated in order to ensure flexibility.
- Ex: Timer construct is a mechanism for ensuring CPU protection.
- Deciding how long the timer is to be set for a particular user is a policy decision

(2+2+1)M

1(c)(i) Identify the difference between Symmetric and Assymmetric Multiprocessing?(4M)

Ans:

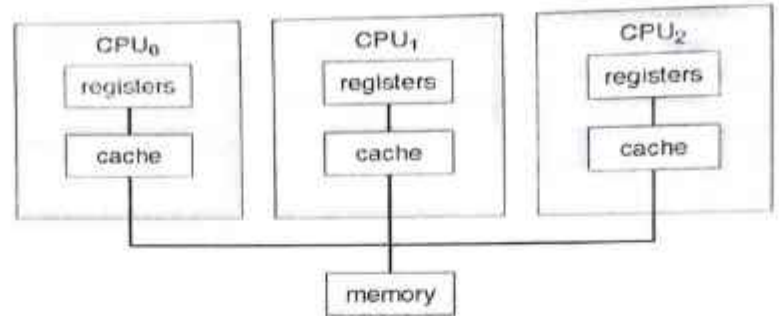
Symmetric Multiprocessing:

- 1) **Asymmetric multiprocessing – (Master/Slave architecture)** Here each processor is assigned a specific task, by the master processor. A master processor controls the other processors in the system. It schedules and allocates work to the slave processors.
- 2) **Symmetric multiprocessing (SMP) –** All the processors are considered peers. There is no

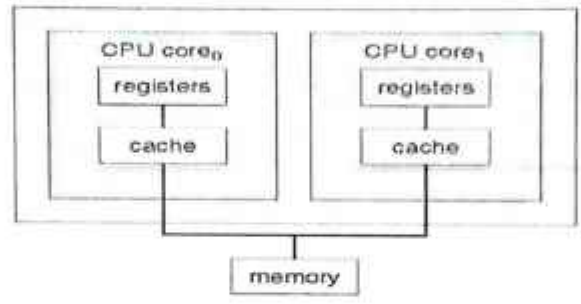
(1+1+1+1)M

master-slave relationship. All the processors have their own registers and CPU, only memory is shared.

SMP Architecture



The benefit of this model is that many processes can run simultaneously. N processes can run if there are N CPUs—without causing a significant deterioration of performance. Operating systems like Windows, Windows XP, Mac OS X, and Linux—now provide support for SMP. A recent trend in CPU design is to include multiple compute **cores** on a single chip. The communication between processors within a chip is faster than communication between two single processors.



1(c)(ii)

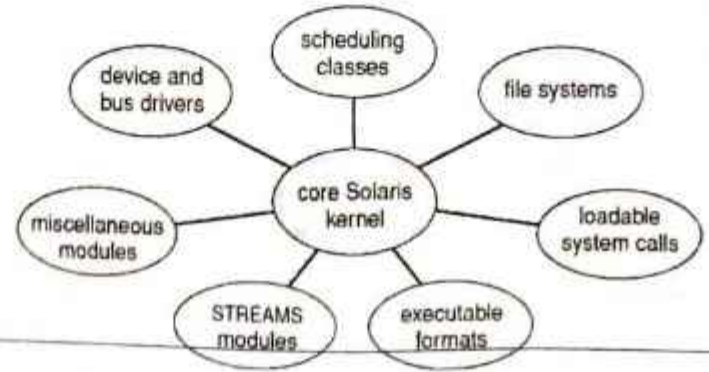
Distinguish between modular kernel approach and layered kernel approach with a neat sketch?
(6M)

Ans:

(3+3M)

Modular kernel approach:

- Modern OS development is object-oriented, with a relatively small core kernel and a set of **modules** which can be linked in dynamically.
- Modules are similar to layers in that each subsystem has clearly defined tasks and interfaces, but any module is free to contact any other module, eliminating the problems of going through multiple intermediary layers.
- The kernel is relatively small in this architecture, similar to microkernels, but the kernel does not have to implement message passing since modules are free to contact each other directly.
Eg: Solaris, Linux and MacOSX



1M

- The Max OSX architecture relies on the Mach microkernel for basic system management services, and the BSD kernel for additional services. Application services and dynamically loadable modules (kernel extensions) provide the rest of the OS functionality.
- Resembles layered system, but a module can call any other module.
Resembles microkernel, the primary module has only core functions and the knowledge of how to load and communicate with other modules.

Layered Approach:

- The OS is broken into number of layers (levels). Each layer rests on the layer below it, and relies on the services provided by the next lower layer.
- Bottom layer (layer 0) is the hardware and the topmost layer is the user interface.
- A typical layer, consists of data structure and routines that can be invoked by higher-level layer.
- Advantage of layered approach is simplicity of construction and debugging.
- The layers are selected so that each uses functions and services of only lower-level layers. So simplifies debugging and system verification. The layers are debugged one by one from the lowest and if any layer doesn't work, then error is due to that layer only, as the lower layers are already debugged. Thus, the design and implementation are simplified.
- A layer need not know how its lower-level layers are implemented. Thus hides the operations from higher layers.

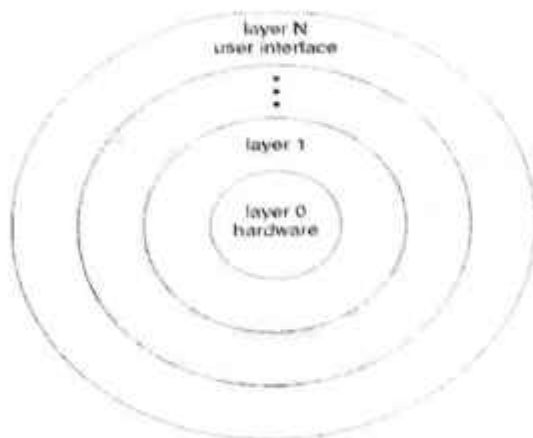


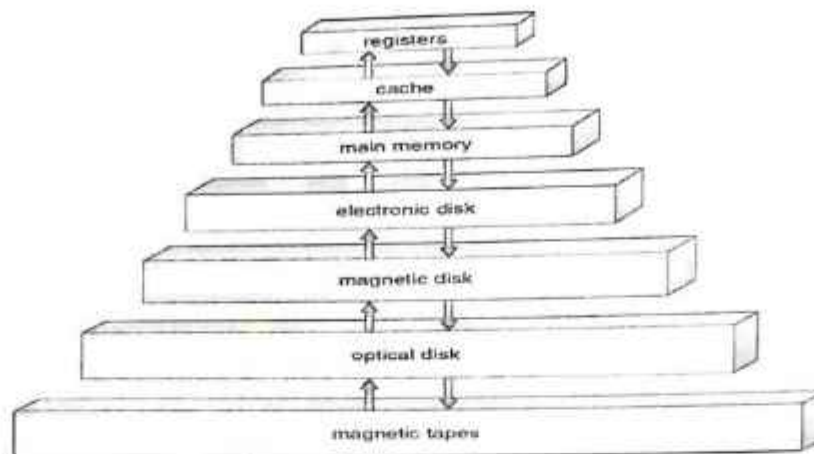
Fig: A layered Operating system

Disadvantages of layered approach:

- The various layers must be appropriately defined, as a layer can use only lower-level layers.
- Less efficient than other types, because any interaction with layer 0 required from top layer. The system call should pass through all the layers and finally to layer 0. This is an overhead.

2(a) Explain about storage structure and its hierarchy in detail with neat diagram? (5M)

Ans:



Storage hierarchy

- Computer programs must be in main memory (**RAM**) to be executed. Main memory is the large memory that the processor can access directly. It commonly is implemented in a semiconductor technology called **dynamic random-access memory (DRAM)**. Computers provide Read Only Memory (ROM), whose data cannot be changed.
- All forms of memory provide an array of memory words. Each word has its own address. Interaction is achieved through a sequence of load or store instructions to specific memory addresses.
- A typical instruction-execution cycle, as executed on a system with a **Von Neumann** architecture, first fetches an instruction from memory and stores that instruction in the **instruction register**. The instruction is then decoded and may cause operands to be fetched from memory and stored in some internal register. After the instruction on the operands has been executed, the result may be stored back in memory
- Ideally, we want the programs and data to reside in main memory permanently. This arrangement usually is not possible for the following two reasons:
 1. Main memory is usually too small to store all needed programs and data permanently
 2. Main memory is a *volatile* storage device that loses its contents when power is turned off.
- Most computer systems provide **secondary storage** as an extension of main memory. The main requirement for secondary storage is that it will be able to hold large quantities of data permanently.
- The most common secondary-storage device is a **magnetic disk**, which provides storage for both programs and data.
- The wide variety of storage systems in a computer system can be organized in a hierarchy as shown in the figure, according to speed, cost and capacity. The higher levels are expensive, but they are fast. As we move down the hierarchy, the cost per bit generally decreases, whereas the access time and the capacity of storage generally increases.
- **Volatile storage** loses its contents when the power to the device is removed. In the absence of expensive battery and generator backup systems, data must be written to **nonvolatile storage** for safekeeping. In the hierarchy shown in figure, the storage

(2M+3M)

systems above the electronic disk are volatile, whereas those below are nonvolatile.

2(b) Compare and contrast between client server computing and peer-to-peer Computing? (5M)

Ans:

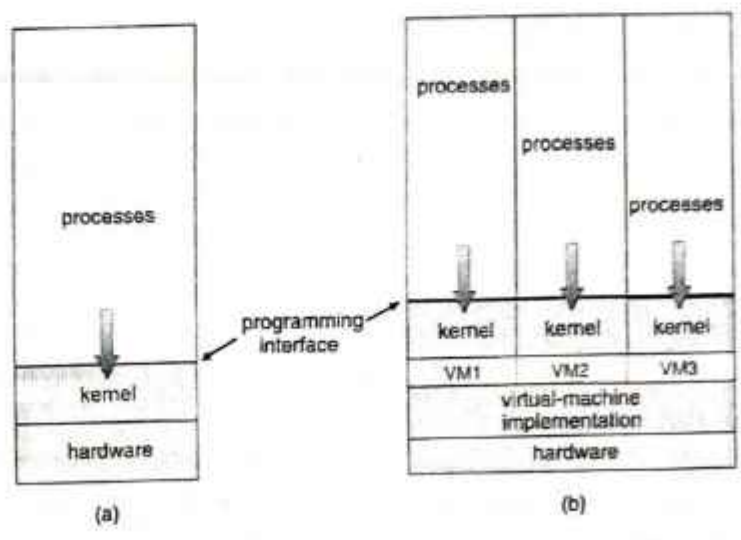
Client server computing	Peer-to-peer computing
1. When it comes to a client-server network, clients and servers are distinguished because of distinctive servers and clients present	1. When it comes to peer-to-peer network, both clients and servers are not distinguished
2. It majorly concentrates on sharing the formation	2. It majorly concentrates on the connectivity part
3. It mainly prefers the centralised server to keep the data and to satisfy the requests from the clients	3. Each and every peer stores its own data and provides services to other nodes.
4. In case of client-server network, The server replies to the services which are asked by the client	4. In case of peer-to-peer network, every node can accomplish both request and response.
5. The client-server is expensive when compared to peer-peer network	5. The peer-peer network is affordable as compared to client-server network
6. More stable form	6. Less stable
7. It can be used both in small and large networks	7. Mostly preferred for short networks

(2.5 + 2.5M)

2(c) Using neat diagram, explain the concept of virtual machines? (10M)

Ans:

- The fundamental idea behind a virtual machine is to abstract the hardware of a single computer (the CPU, memory, disk drives, network interface cards, and so forth) into several different execution environments, thereby creating the illusion that each separate execution environment is running its own private computer.
- Creates an illusion that a process has its own processor with its own memory.
- Host OS is the main OS installed in system and the other OS installed in the system are called guestOS.



(2M)

Figure: System modes. (A) Non-virtual machine (b) Virtual machine
 Virtual machines first appeared as the VM Operating System for IBM mainframes in 1972.

Implementation

- The virtual-machine concept is useful, it is difficult to implement.
- Work is required to provide an exact duplicate of the underlying machine. Remember that the underlying machine has two modes: user mode and kernel mode.
- The virtual-machine software can run in kernel mode, since it is the operating system. The virtual machine itself can execute in only user mode.

Benefits

- Able to share the same hardware and run several different execution environments (OS).
- Host system is protected from the virtual machines and the virtual machines are protected from one another. A virus in guest OS, will corrupt that OS but will not affect the other guest systems and host systems.
- Even though the virtual machines are separated from one another, software resources can be shared among them. Two ways of sharing s/w resource for communication are:
 - To share a file system volume (part of memory).
 - To develop a virtual communication network to communicate between the virtual machines.
- The operating system runs on and controls the entire machine. Therefore, the current system must be stopped and taken out of use while changes are made and tested. This period is commonly called system development time. In virtual machines such problem is eliminated. User programs are executed in one virtual machine and system development is done in another environment.
- Multiple OS can be running on the developer's system **concurrently**. This helps in rapid prototyping and testing of programmer's code in different environments.
- **System consolidation** – two or more systems are made to run in a single system.

(8M)

Simulation –

Here the host system has one system architecture and the guest system is compiled in different architecture. The compiled guest system programs can be run in an emulator that translates each instructions of guest program into native instructions set of host system.

Para-Virtualization –

This presents the guest with a system that is similar but not identical to the guest's preferred system. The guest must be modified to run on the para-virtualized hardware.

3(a) Define cooperating process. Identify the reason for process cooperation? (5M)

Ans:

Cooperating process is defined as processes that affects other processes or be affected by other executing in the system.

Reasons for process cooperation:

- **Information Sharing** - There may be several processes which need to access the same file. So the information must be accessible at the same time to all users.
- **Computation speedup** - Often a solution to a problem can be solved faster if the problem can be broken down into sub-tasks, which are solved simultaneously (particularly when multiple

(+4)

processors are involved.)

- 1) **Modularity** - A system can be divided into cooperating modules and executed by sending information among one another.
- 1) **Convenience** - Even a single user can work on multiple tasks by information sharing.

3(b) Distinguish between direct and indirect communication? (5M)

Ans : **Direct communication** the sender and receiver must explicitly know each other's name. The syntax for send() and receive() functions are as follows-

- **send (P, message)** - send a message to process P
- **receive(Q, message)** - receive a message from process Q

(2.5M)

Properties of communication link:

- A link is established automatically between every pair of processes that wants to communicate. The processes need to know only each other's identity to communicate.
- A link is associated with exactly one pair of communicating processes
- Between each pair, there exists exactly one link.

Types of addressing in direct communication -

- **Symmetric addressing** - the above-described communication is symmetric communication. Here both the sender and the receiver processes have to name each other to communicate.
- **Asymmetric addressing** - Here only the sender's name is mentioned, but the receiving data can be from any system.

send (P, message) --- Send a message to process P
receive (id, message). Receive a message from any process

Disadvantages of direct communication - any changes in the identifier of a process, may have to change the identifier in the whole system (sender and receiver), where the messages are sent and received.

b) Indirect communication uses shared mailboxes, or ports.

(2.5M)

A mailbox or port is used to send and receive messages. Mailbox is an object into which messages can be sent and received. It has a unique ID. Using this identifier messages are sent and received.

Two processes can communicate only if they have a shared mailbox. The send and receive functions are -

- **send (A, message)** - send a message to mailbox A
- **receive (A, message)** - receive a message from mailbox A

Properties of communication link:

- A link is established between a pair of processes only if they have a shared mailbox
- link may be associated with more than two processes
- Between each pair of communicating processes, there may be any number of

links, each link associated with one mailbox.

- A mail box can be owned by the operating system. It must take steps to -
 - create a new mailbox
 - send and receive messages from mailbox
 - delete mailboxes.

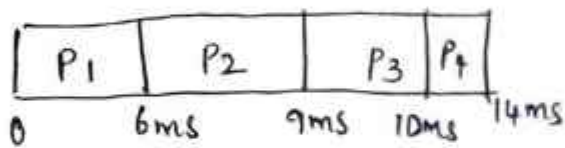
3(e) Consider the following set of processes with CPU burst time(in ms)

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

Calculate average waiting time, average turnaround time for the above process using FCFS, SRTF and RR scheduling algorithm. Represent it with GANNT Chart. (10M)

FCFS Scheduling:

GANNT CHART



$$\text{Turnaround Time} = \text{Completion Time} - \text{Arrival Time}$$

$$\text{Waiting Time} = \text{Turnaround Time} - \text{Burst Time}$$

<u>Process</u>	<u>Turnaround Time</u>	<u>Waiting Time</u>
P1	6ms	0ms
P2	9-1=8ms	8-3=5ms
P3	10-2=8ms	8ms-1=7ms
P4	14-3=11ms	11-4=7ms

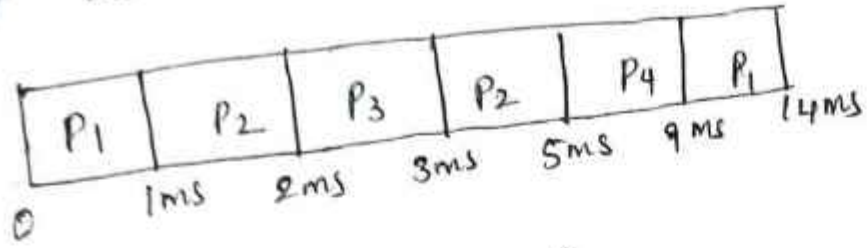
Average Turnaround Time = $(6+8+8+11)/4 = 33/4 = 8.25\text{ms}$

Average Waiting Time = $(0+5+7+7)/4 = 19/4 = 4.75\text{ms}$

(2M)

Shortest Remaining Time first)

GANNT CHART

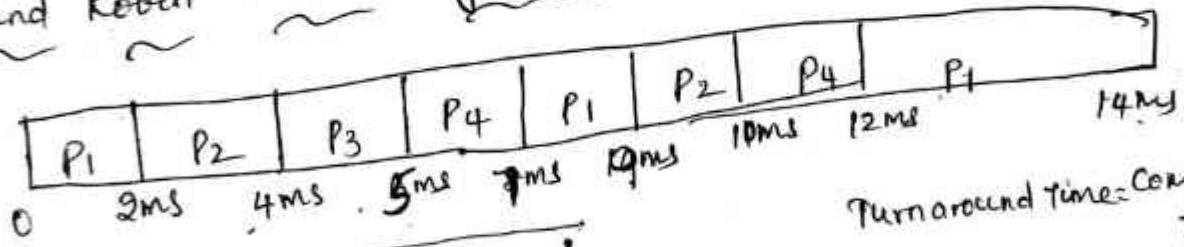


Process	<u>T.A.T</u>	<u>W.T</u>
P1	14ms	8ms
P2	4ms	1ms
P3	1ms	0ms
P4	6ms	2ms

Average Turn around time = $(14 + 4 + 1 + 6) / 4 = \frac{25}{4} = 6.25ms$

Average Waiting time = $(8 + 1 + 0 + 2) / 4 = \frac{11}{4} = 2.75ms$

Round Robin Scheduling algorithm: (Q=2ms)



Turn around time = Completion time - Arrival time

Waiting time = T.A.T - Burst time

Process	<u>T.A.T</u>	<u>W.T</u>	
P1	14 - 0 = 14ms	8ms	(14 - 6)ms
P2	10ms - 1 = 9ms	6ms	(9 - 3)ms
P3	5 - 2 = 3ms	2ms	(3 - 1) = 2ms
P4	12 - 3 = 9ms	5ms	(9 - 4) = 5ms

Average Turn around time = $(14 + 9 + 3 + 9) / 4 = 8.75ms$

Average Waiting time = $(8 + 6 + 2 + 5) / 4 = \frac{21}{4} = 5.25ms$

4M

4M

4(a) Determine the purpose of scheduler? Identify the scheduling criteria? (5M)

Ans:

Schedulers are software which selects an available program to be assigned to CPU.

Scheduling Criteria:

- Criteria to compare CPU-scheduling algorithms:

1) CPU Utilization

- > We must keep the CPU as busy as possible.
- > In a real system, it ranges from 40% to 90%.

Throughput

- > Number of processes completed per time unit.
- > For long processes, throughput may be 1 process per hour:
For short transactions, throughput might be 10 processes per second.

2) Turnaround Time

- > The interval from the time of submission of a process to the time of completion.
- > Turnaround time is the sum of the periods spent
 - waiting to get into memory
 - waiting in the ready-queue
 - executing on the CPU and
 - doing I/O.

3) Waiting Time

- > The amount of time that a process spends waiting in the ready-queue.

4) Response Time

- > The time from the submission of a request until the first response is produced.
- > The time is generally limited by the speed of the output device.

4(b) Identify the challenges faced while implementing multi core systems. (5M)

Ans:

1. Dividing activities:

This involves examining applications to find areas that can be divided into separate, concurrent tasks and thus can run in parallel on individual cores.

2. Balance:

While identifying tasks that can run in parallel, programmers must ensure that the tasks perform equal work of equal value.

In some instances, a certain task may not contribute as much value to the overall process as other tasks; using separate execution core to run that task.

3. Data Splitting:

The data accessed and manipulated by the tasks must be divided to run on separate cores.

4. Data Dependency:

The data accessed by the tasks must be examined for dependencies between two or more tasks.

5. Testing and Debugging: When a program is running in parallel on multiple cores, there are many different execution paths. Testing and Debugging concurrent programs is more difficult than testing and debugging single threaded applications.

4(c)

Build Multithreading models and explain them in detail? (10M)

Ans:

Multithreading models:

Multi-Threading Models

- Support for threads may be provided at either
 - 1) The user level, for **user threads** or

(1+4)M

(5M)

- 2) By the kernel, for **kernel threads**.
- User-threads are supported above the kernel and are managed without kernel support. Kernel-threads are supported and managed directly by the OS.
- Three ways of establishing relationship between user-threads & kernel-threads:
 - 1) Many-to-one model
 - 2) One-to-one model and
 - 3) Many-to-many model.

Many-to-One Model

- Many user-level threads are mapped to one kernel thread (Figure 2.2).
- Advantage:
 - 1) Thread management is done by the thread library in user space, so it is efficient.
- Disadvantages:
 - 1) The entire process will block if a thread makes a blocking system-call.
 - 2) Multiple threads are unable to run in parallel on multiprocessors.
- For example:
 - Solaris green threads
 - GNU portable threads.

(2.5M + 2.5M + 2.5M + 2.5)

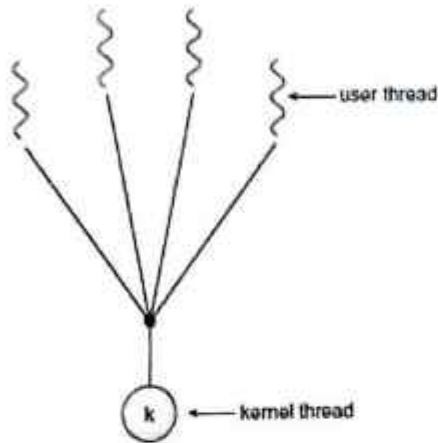


Figure 2.2 Many-to-one model

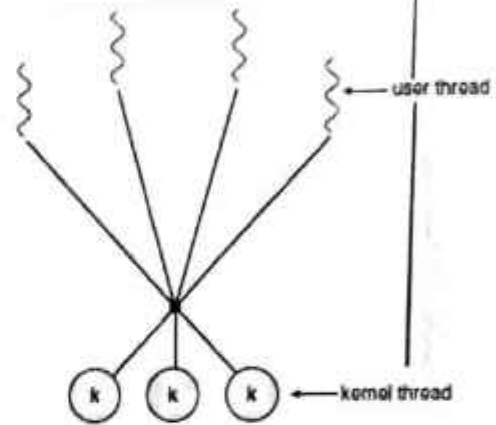


Figure 2.3 One-to-one model

One-to-One Model

- Each user thread is mapped to a kernel thread (Figure 2.3).
- Advantages:
 - 1) It provides more concurrency by allowing another thread to run when a thread makes a blocking system-call.
 - 2) Multiple threads can run in parallel on multiprocessors.
- Disadvantage:
 - 1) Creating a user thread requires creating the corresponding kernel thread.
- For example:
 - Windows NT/XP/2000
 - Linux

2.5M

Many-to-Many Model

- Many user-level threads are multiplexed to a smaller number of kernel threads (Figure 2.4).
- Advantages:
 - 1) Developers can create as many user threads as necessary
 - 2) The kernel threads can run in parallel on a multiprocessor.

2.5M

3) When a thread performs a blocking system-call, kernel can schedule another thread for execution.

Two Level Model

- A variation on the many-to-many model is the two level-model (Figure 2.5).
- Similar to M:M, except that it allows a user thread to be bound to kernel thread.
- For example:
 - HP-UX
 - Tru64 UNIX

2.5M

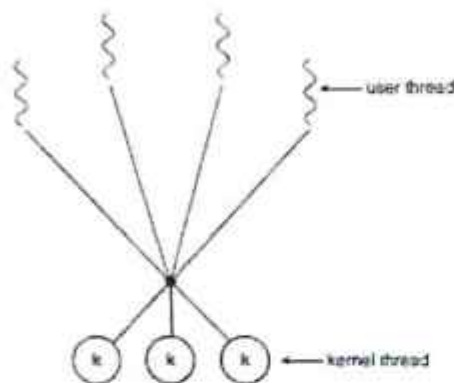


Figure 2.4 Many-to-many model

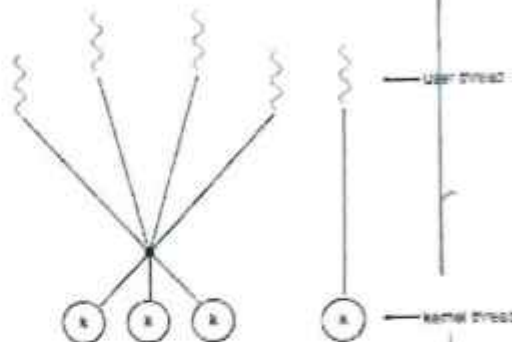


Figure 2.5 Two-level

5(a) Identify the purpose of process synchronization and Apply the approaches used to handle critical section in OS?(5)

Ans:

- Process synchronization is required different processes concurrently access the shared data.
- Process synchronization is used to maintain DATA CONSISTENCY.
- Orderly execution of cooperating processes is achieved through process synchronization.

1M

Approaches to handle critical section problem

Two approaches used to handle critical-sections.

- Preemptive kernels
- Non-Preemptive kernels

4M

Preemptive kernel:

- It allows a process to be preempted while it is running in kernel mode.
- Suitable for real-time programming

Non-preemptive kernel:

- It does not allow a process running in kernel mode to be preempted.
- More responsive

5(b) Show how Semaphores provide solution to implement mutual exclusion problem?(5)

- A semaphore is a synchronization-tool.
- It used to control access to shared-variables so that only one process may at any point in time change the value of the shared-variable.
- A semaphore(S) is an integer-variable that is accessed only through 2 atomic-operations:
 - wait() and
 - signal().
- wait() is termed P ("to test").
signal() is termed V ("to increment").

(1+2+2)M

• Definition of wait():

```
wait(S) {
  while (S <= 0)
    ; // busy wait
  S--;
}
```

Definition of signal():

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

- When one process modifies the semaphore-value, no other process can simultaneously modify that same semaphore-value. Thus mutual exclusion condition is preserved.
- Also, in the case of wait(S), following 2 operations must be executed without interruption:
 - 1) Testing of S(S<=0) and
 - 2) Modification of S (S--)

6(a) Design a Software based solution for Critical section problem and prove that mutual exclusion property is preserved. (5M)

Peterson's Solution

- This is a classic software-based solution to the critical-section problem.
- This is limited to 2 processes.
- The 2 processes alternate execution between
 - critical-sections and
 - remainder-sections.
- The 2 processes share 2 variables (Figure 2.13):


```
int turn;
boolean flag[2];
```

 - where turn = i indicates whose turn it is to enter its critical-section. (i.e., if turn==i, then process Pi is allowed to execute in its critical-section).
 - flag =true used to indicate if a process is ready to enter its critical-section. (i.e. if flag[i]=true, then Pi is ready to enter its critical-section).

(2+3M)

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

Figure 2.13 The structure of process Pi in Peterson's solution

- To enter the critical-section,
 - firstly process Pi sets flag[i] to be true and
 - then sets turn to the value j.
- If both processes try to enter at the same time, turn will be set to both i and j at roughly the sametime.
- The final value of turn determines which of the 2 processes is allowed to enter its critical-section first.
- To prove that this solution is correct, we show that:
 - 1) Mutual-exclusion is preserved.
 - 2) The progress requirement is satisfied.
 - 3) The bounded-waiting requirement is met.

6(b) Show a scenario where deadlock occurs with example?(5M)

Ans: Deadlock occurs when 2 or more processes are waiting indefinitely for an event that is caused by only one of the waiting processes.

(2+3M)

Example: Consider there are 2 processes P0 and P1 and there are R1 and R2 resource type each is having a single instance.

P0 is holding a resource type R1 and is waiting for an instance of resource type R2 which is acquired by P1

P1 is holding a resource type R2 and is waiting for an instance of resource type R1 which is acquired by P0

Thus both P0 and P1 are indefinitely waiting

30
COURSE INCHARGE

VR
HOD



K S INSTITUTE OF TECHNOLOGY

Bangalore – 560109

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING(CCE)

CIE Question paper Scrutiny format

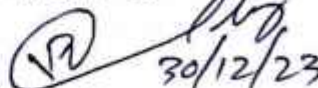
Course Name	Object Oriented Programming with Java
Course Code	BCS306A
Course Incharge	Mrs.Rajashree M Byalal
Academic year	2023-2024
Semester	3 rd
CIE #	IA - 1
Set	A <input checked="" type="checkbox"/> B <input type="checkbox"/>
Scrutiny parameters	
Whether questions are according to assessment plan?	Yes <input checked="" type="checkbox"/> No <input type="checkbox"/> ; If No, Suggestions:
Whether questions prepared are within the covered syllabus?	Yes <input checked="" type="checkbox"/> No <input type="checkbox"/> ; If No, Suggestions:
Whether all questions are mapped to CO/PO properly?	Yes <input checked="" type="checkbox"/> No <input type="checkbox"/> ; If No, Suggestions:
Whether questions framed are according to Blooms level?	Yes <input checked="" type="checkbox"/> No <input type="checkbox"/> ; If No, Suggestions:
Whether marks distribution for each question are correct?	Yes <input checked="" type="checkbox"/> No <input type="checkbox"/> ; If No, Suggestions:
Whether questions paper follows the format displayed?	Yes <input checked="" type="checkbox"/> No <input type="checkbox"/> ; If No, Suggestions:
Difficulty level	Very High <input type="checkbox"/> High <input checked="" type="checkbox"/> Moderate <input type="checkbox"/> Low <input type="checkbox"/>
Percentage of Similarity questions in Set A & B	0%
Final decision	Accepted without corrections <input type="checkbox"/> Accepted with minor corrections <input type="checkbox"/> Not accepted <input type="checkbox"/>

(Rajashree M Byalal)

R. Byalal.

Signature with date
of CIE Question paper setter

Dr. Chandra J. Reddy


30/12/23

Name and Signature with date
of CIE Question paper Scrutiniser



K S INSTITUTE OF TECHNOLOGY

Bangalore – 560109

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

CIE Question paper Scrutiny format

Course Name	OPERATING SYSTEM
Course Code	BCS303
Course Incharge	Mrs.T.Naga Jyothi
Academic year	2023-2024
Semester	3 rd
CIE #	IA – 2
Set	A <input checked="" type="checkbox"/> B <input type="checkbox"/>
Scrutiny parameters	
Whether questions are according to assessment plan?	Yes <input checked="" type="checkbox"/> No <input type="checkbox"/> ; If No, Suggestions:
Whether questions prepared are within the covered syllabus?	Yes <input checked="" type="checkbox"/> No <input type="checkbox"/> ; If No, Suggestions:
Whether all questions are mapped to CO/PO properly?	Yes <input checked="" type="checkbox"/> No <input type="checkbox"/> ; If No, Suggestions:
Whether questions framed are according to Blooms level?	Yes <input checked="" type="checkbox"/> No <input type="checkbox"/> ; If No, Suggestions:
Whether marks distribution for each question are correct?	Yes <input checked="" type="checkbox"/> No <input type="checkbox"/> ; If No, Suggestions:
Whether questions paper follows the format displayed?	Yes <input checked="" type="checkbox"/> No <input type="checkbox"/> ; If No, Suggestions:
Difficulty level	Very High <input type="checkbox"/> High <input type="checkbox"/> Moderate <input checked="" type="checkbox"/> Low <input type="checkbox"/>
Percentage of Similarity questions in Set A & B	N I L L
Final decision	Accepted without corrections <input checked="" type="checkbox"/> Accepted with minor corrections <input type="checkbox"/> Not accepted <input type="checkbox"/>

Signature with date
of CIE Question paper setter

Dr Chanda V Reddy

Name and Signature with date
of CIE Question paper Scrutiniser



K.S. INSTITUTE OF TECHNOLOGY, BANGALORE - 560109
SECOND INTERNAL TEST 2023 - 24 ODD SEMESTER

SET-A

SCHEME AND SOLUTION

Semester : III

Degree

: B.E

Branch - stream : Computer and Communication
Engineering - CSE

Course Type / Code : Core / BCS303

Course Title : Operating systems

Max Marks : 25

1. Experiment with different methods to recovery from deadlocks and
Build a resource allocation graph by experimenting with Processes P0
& P1 and two resources R1 and R2 (10M)

Ans:

The system recovers from the deadlock automatically. There are two options for breaking a deadlock one is simply to abort one or more processes to break the circular wait. The other is to preempt some resources from one or more of the deadlocked processes.

Process Termination

To eliminate deadlocks by aborting a process, use one of two methods. In both methods, the system reclaims all resources allocated to the terminated processes.

1. Abort all deadlocked processes:

This method clearly will break the deadlock cycle, but at great expense; 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.

2. 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.

If the partial termination method is used, then we must determine which deadlocked process (or processes) should be terminated. Many factors may affect which process is chosen, including:

1. What the priority of the process is
2. How long the process has computed and how much longer the process will compute before completing its designated task
3. How many and what types of resources the process has used.
4. How many more resources the process needs in order to complete
5. How many processes will need to be terminated?
6. Whether the process is interactive or batch

(8M + 2M)

= (4M + 4M + 2M)

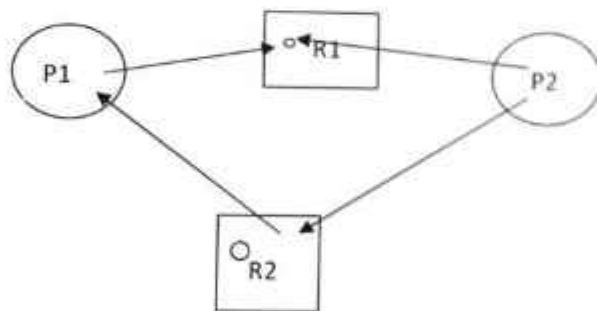
Resource Preemption

To eliminate deadlocks using resource preemption, we successively preempt some resources from processes and give these resources to other processes until the deadlock cycle is broken.

If preemption is required to deal with deadlocks, then three issues need to be addressed:

1. **Selecting a victim.** Which resources and which processes are to be preempted? As in process termination, we must determine the order of preemption to minimize cost. 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.
2. **Rollback.** If we preempt a resource from a process, what should be done with that process? Clearly, it cannot continue with its normal execution; it is missing some needed resource. We must roll back the process to some safe state and restart it from that state. Since it is difficult to determine what a safe state is, the simplest solution is a total rollback: abort the process and then restart it.
3. **Starvation.** How do we ensure that starvation will not occur? That is, how can we guarantee that resources will not always be preempted from the same process?

RESOURCE ALLOCATION GRAPH:



3. **Build a model for multi-step processing of a user program and explain it in detail?**

Ans:

1. User programs typically refer to memory addresses with symbolic names
2. These symbolic names must be mapped or bound to physical memory addresses.
3. Address binding of instructions to memory addresses can happen at 3 different stages.

Compile Time-

If it is known at compile time where a program will reside in physical memory, then absolute code can be generated by the compiler, containing actual physical addresses. However, if the load address changes at some later time, then the program will have to be recompiled.

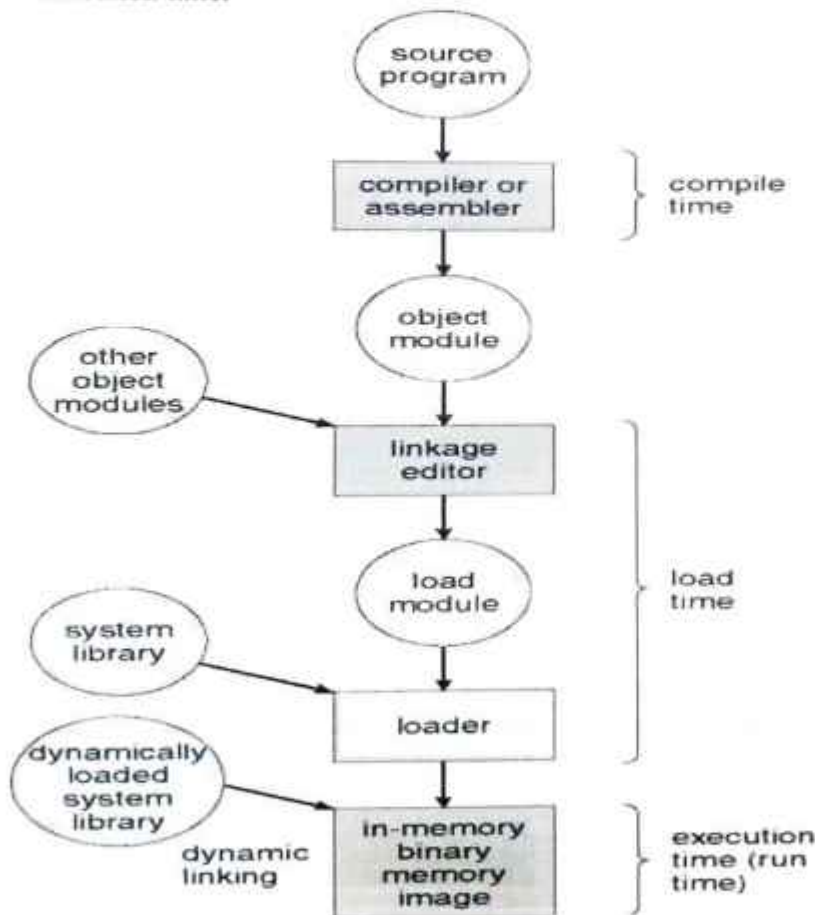
Load Time-

If the location at which a program will be loaded is not known at compile time, then the compiler must generate relocatable code, which references addresses relative to the start of the program. If that starting address changes, then the program must be reloaded but not recompiled.

Execution Time-

If a program can be moved around in memory during the course of its execution, then binding must be delayed until execution time.

(8+2)M



3(b) 4. Make use of swapping. Construct a model for swapping of two processes using a disk as a backing store and explain its disadvantages.

(10M)

Swapping:

- A process must be loaded in to memory in order to execute.
- If there is not enough memory available to keep all running processes in memory at the same time, then some processes that are not currently using the CPU may have their memory swapped out to a fast local disk called the backing store.
- Swapping is the process of moving a process from memory to backing store and moving another process from backing store to memory. Swapping is a very slow process compared to other operations.
- A variant of swapping policy is used for priority-based scheduling algorithms. If a higher-priority process arrives and wants service, the memory manager can swap out the lower-priority process and then load and execute the higher-priority process. When the higher-priority process finishes, the lower-priority process can be swapped back in and continued. This variant of swapping is called roll out, roll in.

(6+2+2)M

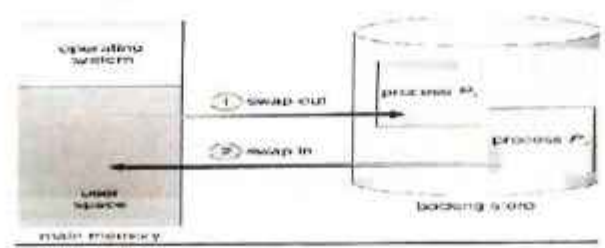
Swapping depends up on address-binding:

- If binding is done at load-time, then process cannot be easily moved to a different location.
- If binding is done at execution-time, then a process can be swapped in to a different memory-space, because the physical-addresses are computed during execution-time.

Major part of swap-time is transfer-time; i.e. total transfer-time is directly proportional to the amount of memory swapped.

Disadvantages:

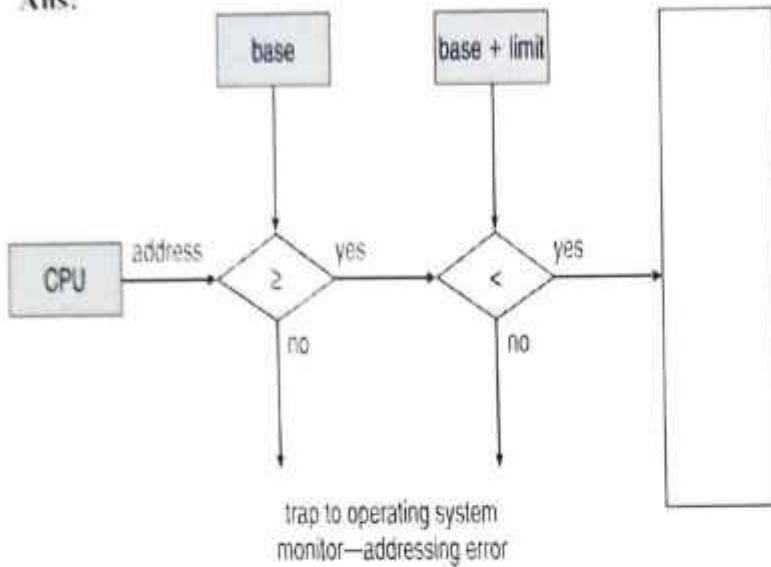
1. Context-switch time is fairly high.
2. If we want to swap a process, we must be sure that it is completely idle. Two solutions:
 - i) Never swap a process with pending I/O. Execute I/O operations only in to OS buffers



Swapping of two processes as a disk store

4(a) Build a model with hardware address protection with base and limit registers and Explain the necessity of base and limit registers. (5M)

Ans:



(2+3M)

Main memory, cache and CPU registers in the processors are the only storage spaces that CPU can access directly. The program and data must be brought into the memory from the disk, for the process to run. Each process has a separate memory space and must access only this range of legal addresses. Protection of memory is required to ensure correct operation. This prevention is provided by hardware implementation.

- Two registers are used - a base register and a limit register. The base register holds the smallest legal physical memory address; the limit register specifies the size of the range.
- For example, The base register holds the smallest legal physical memory address; the limit register specifies the size of the range. For example, if the base register holds 300040 and limit register is 420940, then the program can legally access all addresses from 300040 through 420940 (inclusive).

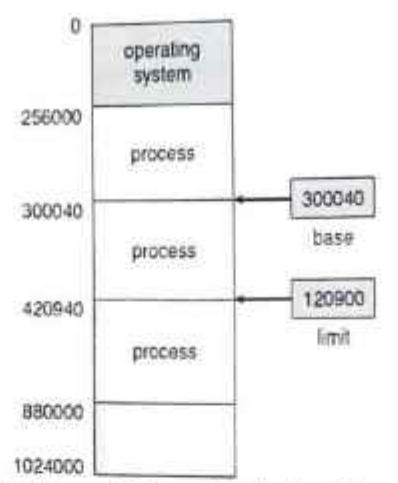


Figure: A base and a limit-register define a logical-address space

The base and limit registers can be loaded only by the operating system, which uses a special privileged instruction. Since privileged instructions can be executed only in kernel mode only the operating system can load the base and limit registers.

4(b) Q. Apply paging hardware with TLB and explain it in detail with a neat block diagram? (10M)

Ans:

Translation Lookaside Buffer

- A special, small, fast lookup hardware cache, called a translation look-aside buffer (TLB).
- Each entry in the TLB consists of two parts :a key(or tag)and a value.
- When the associative memory is presented with an item, the item is compared with all keys simultaneously. If the item is found, the corresponding value field is returned. The search is fast; the hardware, however, is expensive. Typically, the number of entries in a TLB is small, often numbering between 64 and 1,024.
- The TLB contains only a few of the page-table entries.

(8+2M)

Working:

- When logical-address is generated by the CPU, its page-number is presented to the TLB.
- If the page-number is found (TLB hit), its frame-number is immediately available and used to access memory
- If page-number is not in TLB (TLB miss), a memory-reference to page table must be made. The obtained frame-number can be used to access memory (Figure 1)

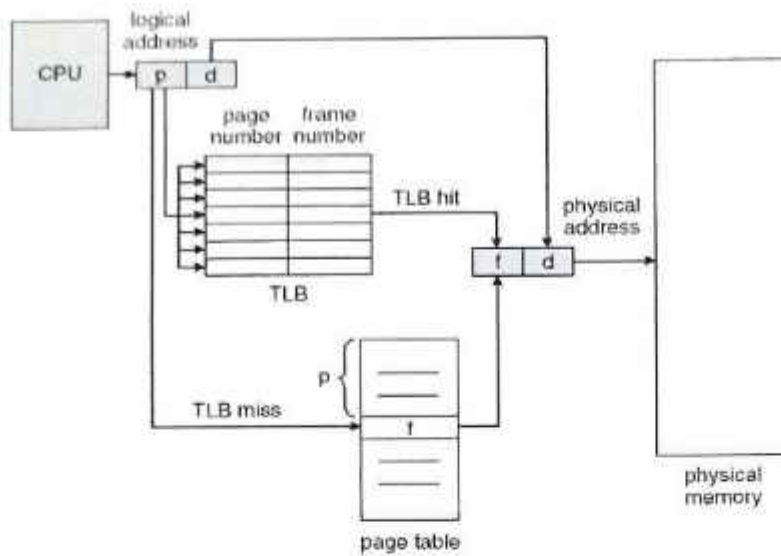


Figure 1: Paging hardware with TLB

- In addition, we add the page-number and frame-number to the TLB, so that they will be found quickly on the next reference.
- If the TLB is already full of entries, the OS must select one for replacement.
- Percentage of times that a particular page-number is found in the TLB is called hit ratio.

Advantage: Search operation is fast. Disadvantage: Hardware is expensive.

- Some TLBs have wired down entries that can't be removed.
- Some TLBs store ASID (address-space identifier) in each entry of the TLB that uniquely identify each process and provide address space protection for that process.

Protection

- Memory-protection is achieved by protection-bits for each frame.
- The protection-bits are kept in the page-table.
- One protection-bit can define a page to be read-write or read-only.
- Every reference to memory goes through the page-table to find the correct frame-number.
- Firstly, the physical-address is computed. At the same time, the protection-bit is checked to verify that no writes are being made to a read-only page.
- An attempt to write to a read-only page causes a hardware-trap to the OS (or memory protection violation).

Signature of Course
Incharge

Signature of
HOD



K S INSTITUTE OF TECHNOLOGY

Bangalore – 560109

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

CIE Question paper Scrutiny format

Course Name	OPERATING SYSTEM
Course Code	BCS303
Course Incharge	Mrs.T.Naga Jyothi
Academic year	2023-2024
Semester	3 rd
CIE #	IA – 2
Set	A <input type="checkbox"/> B <input checked="" type="checkbox"/>
Scrutiny parameters	
Whether questions are according to assessment plan?	Yes <input checked="" type="checkbox"/> No <input type="checkbox"/> ; If No, Suggestions:
Whether questions prepared are within the covered syllabus?	Yes <input checked="" type="checkbox"/> No <input type="checkbox"/> ; If No, Suggestions:
Whether all questions are mapped to CO/PO properly?	Yes <input checked="" type="checkbox"/> No <input type="checkbox"/> ; If No, Suggestions:
Whether questions framed are according to Blooms level?	Yes <input type="checkbox"/> No <input type="checkbox"/> ; If No, Suggestions:
Whether marks distribution for each question are correct?	Yes <input type="checkbox"/> No <input type="checkbox"/> ; If No, Suggestions:
Whether questions paper follows the format displayed?	Yes <input checked="" type="checkbox"/> No <input type="checkbox"/> ; If No, Suggestions:
Difficulty level	Very High <input type="checkbox"/> High <input type="checkbox"/> Moderate <input checked="" type="checkbox"/> Low <input type="checkbox"/>
Percentage of Similarity questions in Set A & B	- 0 -
Final decision	Accepted without corrections <input checked="" type="checkbox"/> Accepted with minor corrections <input type="checkbox"/> Not accepted <input type="checkbox"/>

A

Signature with date
of CIE Question paper setter

Dr Chanda V Reddy
(Signature)

Name and Signature with date
of CIE Question paper Scrutiniser



SET- B

SCHEME AND SOLUTION

Degree : B.E

Semester : III

Branch : Computer and Communication Engg

Course : BCS303

Course Title : Operating systems

Code
Max Marks: 25

1. Make use of deadlock prevention technique and explain them in detail?

Ans:

DEADLOCK PREVENTION

(10M)

Deadlock can be prevented by ensuring that at least one of the four necessary conditions cannot hold.

Mutual Exclusion

- The mutual-exclusion condition must hold for non-sharable resources. Sharable resources, do not require mutually exclusive access and thus cannot be involved in a deadlock.
- Ex: Read-only files are example of a sharable resource. If several processes attempt to open a read-only file at the same time, they can be granted simultaneous access to the file. A process never needs to wait for a sharable resource.
- Deadlocks cannot prevent by denying the mutual-exclusion condition, because some resources are intrinsically non-sharable.

Hold and Wait

To ensure that the hold-and-wait condition never occurs in the system, then guarantee that, whenever a process requests a resource, it does not hold any other resources.

- One protocol that can be used requires each process to request and be allocated all its resources before it begins execution.
- Another protocol allows a process to request resources only when it has none. A process may request some resources and use them. Before it can request any additional resources, it must release all the resources that it is currently allocated.

Ex:

- Consider a process that copies data from a DVD drive to a file on disk, sorts the file, and then prints the results to a printer. If all resources must be requested at the beginning of the process, then the process must initially request the DVD drive, disk file, and printer. It will hold the printer for its entire execution, even though it needs the printer only at the end.
- The second method allows the process to request initially only the DVD drive and disk file. It copies from the DVD drive to the disk and then releases both the DVD drive and the disk file. The process must then again request the disk file and the printer. After copying the disk file to the printer, it releases these two resources and terminates.

(2.5+2.5+2.5
+2.5M)

The two main disadvantages of these protocols:

1. Resource utilization may be low, since resources may be allocated but unused for a long period.
2. Starvation is possible.

No Preemption

The third necessary condition for deadlocks is that there be no preemption of resources that have already been allocated.

To ensure that this condition does not hold, the following protocols can be used:

- If a process is holding some resources and requests another resource that cannot be immediately allocated to it, then all resources the process is currently holding are preempted.
- The preempted resources are added to the list of resources for which the process is waiting. The process will be restarted only when it can regain its old resources, as well as the new ones that it is requesting.

If a process requests some resources, first check whether they are available. If they are, allocate them.

If they are not available, check whether they are allocated to some other process that is waiting for additional resources. If so, preempt the desired resources from the waiting process and allocate them to the requesting process.

If the resources are neither available nor held by a waiting process, the requesting process must wait. While it is waiting, some of its resources may be preempted, but only if another process requests them.

A process can be restarted only when it is allocated the new resources it is requesting and recovers any resources that were preempted while it was waiting.

Circular Wait

One way to ensure that this condition never holds is to impose a total ordering of all resource types and to require that each process requests resources in an increasing order of enumeration.

To illustrate, let $R = \{R_1, R_2, \dots, R_m\}$ be the set of resource types. Assign a unique integer number to each resource type, which allows to compare two resources and to determine whether one precedes another in ordering. Formally, it is defined as a one-to-one function

$F: R \rightarrow N$, where N is the set of natural numbers.

Example: if the set of resource types R includes tape drives, disk drives, and printers, then the function F might be defined as follows:

$$\begin{aligned} F(\text{tapedrive}) &= 1 & F(\text{diskdrive}) &= 5 & F(\text{printer}) &= 12 \end{aligned}$$

Now consider the following protocol to prevent deadlocks. Each process can request resources only in an increasing order of enumeration. That is, a process can initially request any number of instances of a resource type R_i . After that, the process can request instances of resource type R_j if and only if $F(R_j) > F(R_i)$.

2. Make use of structure of page table with suitable diagrams and explain it in detail.

(10M)

Ans: Structure of the Page Table

The most common techniques for structuring the page table:

1. Hierarchical Paging
2. Hashed Page-tables
3. Inverted Page-tables

1. Hierarchical Paging

- Problem: Most computers support a large logical-address space (232 to 264). In these systems, the page-table itself becomes excessively large.
- Solution: Divide the page-table in to smaller pieces.

(2+3.5+3)

Two Level Paging Algorithm:

- The page-table itself is also paged.
- This is also known as a forward-mapped page-table because address translation works from the outer page-table inwards.

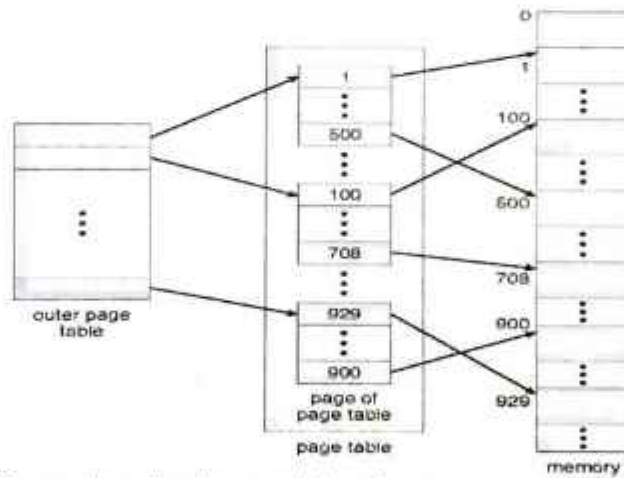


Figure: A two-level page-table scheme

For example:

Consider the system with a 32-bit logical-address space and a page-size of 4KB.

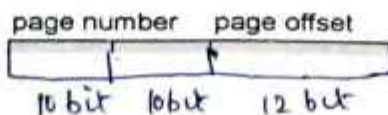
A logical-address is divided into

- 20-bit page-number and
- 12-bit page-offset.

Since the page-table is paged, the page-number is further divided in to

- 10-bit page-number and
- 10-bit page-offset.

Thus, a logical-address is as follows:



- where p_1 is an index into the outer page table, and p_2 is the displacement within the page of the inner page table

The address-translation method for this architecture is shown in below figure. Because address translation works from the outer page table inward, this scheme is also known as a forward-mapped page table.

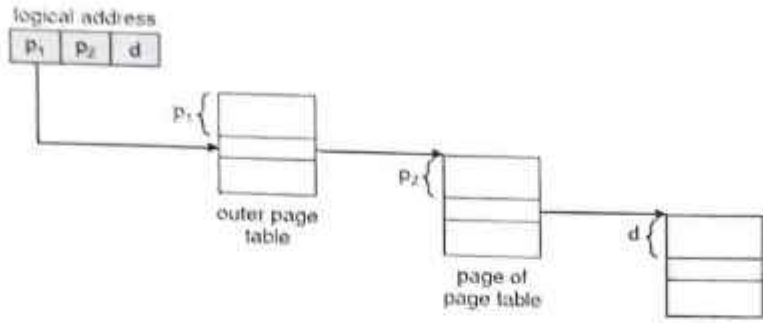


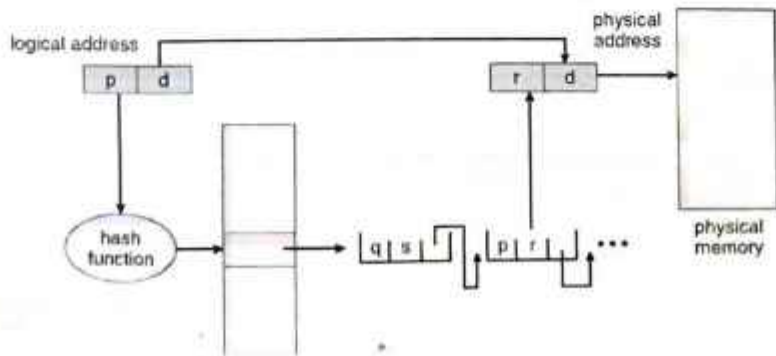
Figure: Address translation for a two-level 32-bit paging architecture

2. HashedPageTables

- This approach is used for handling address spaces larger than 32 bits.
- The hash-value is the virtual page-number.
- Each entry in the hash-table contains a linked-list of elements that has to the same location (to handle collisions).
- Each element consists of 3 fields:
 1. Virtual page-number
 2. Value of the mapped page-frame and
 3. Pointer to the next element in the linked-list.

The algorithm works as follows:

- The virtual page-number is hashed in to the hash-table.
- The virtual page-number is compared with the first element in the linked-list.
- If there is a match, the corresponding page-frame (field 2) is used to form the desired physical-address.
- If there is no match, subsequent entries in the linked-list are searched for a matching virtual page-number.



3. Inverted Page Tables

- Has one entry for each real page of memory.
- Each entry consists of virtual-address of the page stored in that real memory-location and information about the process that owns the page.
- Each virtual-address consists of a triplet <process-id, page-number, offset>.
- Each inverted page-table entry is a pair <process-id, page-number>

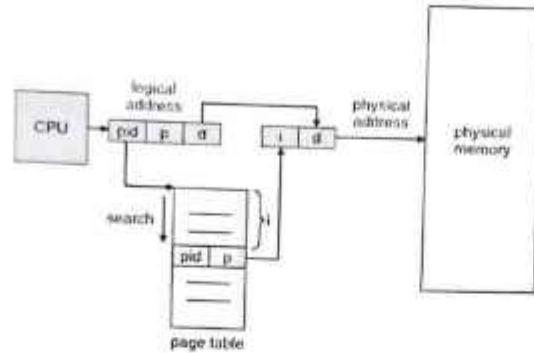


Figure: Inverted page-table

The algorithm works as follows:

1. When a memory-reference occurs, part of the virtual-address, consisting of <process-id, page-number>, is presented to the memory subsystem.
2. The inverted page-table is then searched for a match.
3. If a match is found, at entry i -then the physical-address < i , offset> is generated.
4. If no match is found, then an illegal address access has been attempted.

Advantage:

1. Decreases memory needed to store each page-table

Disadvantages:

1. Increases amount of time needed to search table when a page reference occurs.
2. Difficulty implementing shared-memory
3. **Identify** the internal difference between Internal and External Fragmentation? (5M)

Internal fragmentation

1. In internal fragmentation fixed-sized memory, blocks square measure appointed to process.
2. Internal fragmentation happens when the method or process is smaller than the memory.
3. The solution of internal fragmentation is the best-fit block.
4. Internal fragmentation occurs when memory is divided into fixed-sized partitions.
5. The difference between memory allocated and required space or memory is called Internal fragmentation.
6. Internal fragmentation occurs with paging and fixed partitioning.
7. It occurs on the allocation of a process to a partition greater than the process's leftover space causes degradation system performance.
8. It occurs in worst fit memory allocation method.

(2.5+2.5M)

External Fragmentation:

1. In external fragmentation, variable-sized memory blocks square measure appointed to the method.
2. External fragmentation happens when the method or process is removed.
3. The solution to external fragmentation is compaction and paging.
4. External fragmentation occurs when memory is divided into variable size partitions based on the size of processes.
5. The unused spaces formed between non-contiguous memory fragments are too small to serve a new process, which is called External fragmentation.
6. External fragmentation occurs with segmentation and dynamic partitioning.
7. It occurs on the allocation of a process to a partition greater which is exactly the same memory space as it is required.
8. It occurs in best fit and first fit memory allocation method.

4. Identify the usage of contiguous memory allocation and explain it with suitable diagram?

Ans:

(10 M)

Contiguous Memory Allocation:

- The main memory must accommodate both the operating system and the various user processes. Therefore we need to allocate the parts of the main memory in the most efficient way possible.
- Memory is usually divided into 2 partitions: One for the resident OS. One for the user processes.
- Each process is contained in a single contiguous section of memory.

1. Memory Mapping and Protection

- Memory-protection means protecting OS from user-process and protecting user-processes from one another.
- Memory-protection is done using
 - Relocation-register: contains the value of the smallest physical-address.
 - Limit-register: contains the range of logical-addresses.
- Each logical-address must be less than the limit-register.
- The MMU maps the logical-address dynamically by adding the value in the relocation- register. This mapped-address is sent to memory
- When the CPU scheduler selects a process for execution ,the dispatcher loads the relocation and limit-registers with the correct values.
- Because every address generated by the CPU is checked against these registers, we can protect the OS from the running-process.
- The relocation-register scheme provides an effective way to allow the OS size to change dynamically.
- Transient OS code: Code that comes&goes as needed to save memory-space and

FM

overhead for unnecessary swapping.

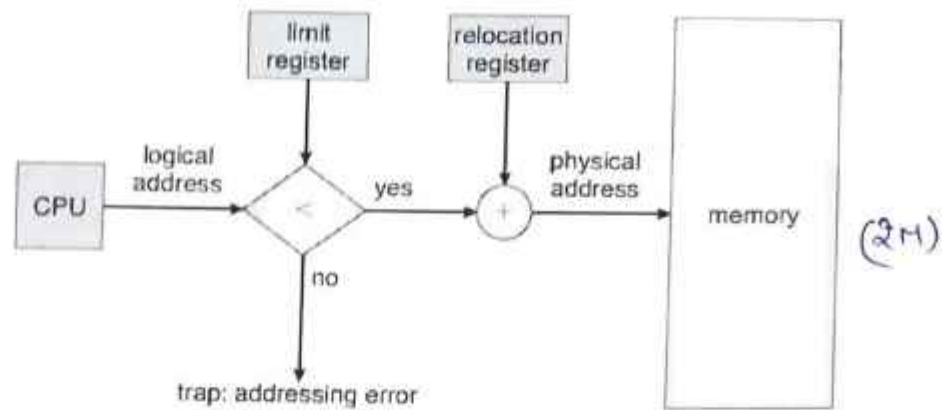


Figure: Hardware support for relocation and limit registers

Memory Allocation

Two types of memory partitioning are:

1. Fixed-sized partitioning
2. Variable-sized partitioning

1. Fixed-sized Partitioning

- The memory is divided into fixed-sized partitions.
- Each partition may contain exactly one process.
- The degree of multiprogramming is bound by the number of partitions.
- When a partition is free, a process is selected from the input queue and loaded into the free partition.
- When the process terminates, the partition becomes available for another process.

2. Variable-sized Partitioning

- The OS keeps a table indicating which parts of memory are available and which parts are occupied.
- A hole is a block of available memory. Normally, memory contains a set of holes of various sizes.
- Initially, all memory is available for user-processes and considered one large hole.
- When a process arrives, the process is allocated memory from a large hole.
- If we find the hole, we allocate only as much memory as is needed and keep the remaining memory available to satisfy future requests.

Three strategies used to select a free hole from the set of available holes:

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.
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.

First-fit and best fit are better than worst fit in terms of decreasing time and storage utilization.



Signature of Course Incharge



Signature of HOD

2(a) Apply Banker's Algorithm to determine whether the following system is in safe state (10M)

Processes	Allocation			Max			Available		
	A	B	C	A	B	C	A	B	C
P ₀	0	1	0	7	5	3	3	3	2
P ₁	2	0	0	3	2	2			
P ₂	3	0	2	9	0	2			
P ₃	2	1	1	2	2	2			
P ₄	0	0	2	4	3	3			

Identify the Need matrix & calculate the Safe sequence.

Step 1: Create a Need matrix

$$\text{Need}(P_0) = (7, 5, 3) - (0, 1, 0) = (7, 4, 3)$$

$$\text{Need}(P_1) = (3, 2, 2) - (2, 0, 0) = (1, 2, 2)$$

$$\text{Need}(P_2) = (9, 0, 2) - (3, 0, 2) = (6, 0, 0)$$

$$\text{Need}(P_3) = (2, 2, 2) - (2, 1, 1) = (0, 1, 1)$$

$$\text{Need}(P_4) = (4, 3, 3) - (0, 0, 2) = (4, 3, 1)$$

Step 2:

Initialise

$$k = (3, 3, 2)$$

finish matrix

P ₀	F
P ₁	F
P ₂	F
P ₃	F
P ₄	F

Step 3:

check P₀ Need \leq Available

$$(7, 4, 3) \leq (3, 3, 2) \quad \times$$

Not satisfied

Step 4

Check P₁ Need \leq Available

$$(1, 2, 2) \leq (3, 3, 2) \quad \checkmark$$

Satisfied

$$k = (3, 3, 2) + (2, 0, 0) = (5, 3, 2)$$

finish matrix

P ₀	F
P ₁	T
P ₂	F
P ₃	F
P ₄	F

Check P₂ Need \leq Available

$(6, 0, 0) \leq (5, 3, 2)$ ✗

P₂ Not satisfied

step 6 :

Check P₃ Need \leq Available

$(0, 1, 1) \leq (5, 3, 2)$ ✓

Satisfied

Work = $(5, 3, 2) + (2, 1, 1)$
 $= (7, 4, 3)$

P ₀	F
P ₁	T
P ₂	F
P ₃	T
P ₄	F

step 7 :

Check P₄ Need \leq Available

$(4, 3, 1) \leq (7, 4, 3)$ ✓

satisfied

work = work + Alloc(P₄)
 $= (7, 4, 3) + (0, 0, 2)$
 $= (7, 4, 5)$

finish matrix

P ₀	F
P ₁	T
P ₂	F
P ₃	T
P ₄	T

step 8

check

P₀ Need = $(7, 4, 5) \leq$ work $(7, 4, 5)$ ✓
 satisfied

work = $(7, 4, 5) + (0, 1, 0)$
 $= (7, 5, 5)$

Finish matrix

P ₀	F
P ₁	T
P ₂	F
P ₃	T
P ₄	F

P ₀	T
P ₁	T
P ₂	False
P ₃	T
P ₄	False

Step 1

check P₂ Need <= Available

$$(6, 0, 0) <= (7, 5, 5) \quad \checkmark$$

Finish matrix $(3, 5, 5)$

P ₀	T
P ₁	T
P ₂	T
P ₃	T
P ₄	T

$$\begin{aligned} \text{K₂} &= (7, 5, 5) + \text{Alloc}(P_2) \\ &= (7, 5, 5) + (3, 0, 2) \\ &= (10, 5, 7) \end{aligned}$$

Steps:

Safe sequence

Safe sequence for the above

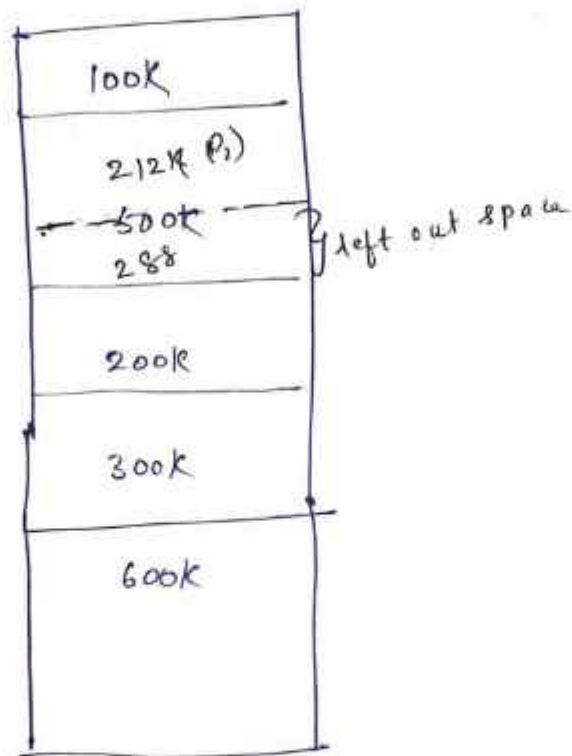
problem is ~~deadlock state~~

$$\langle P_1, P_3, P_4, P_0, P_2 \rangle$$

~~(P₄, P₃, P₁)~~

Q. First fit, Best fit, Worst fit for the given memory
 Partitions of 100K, 500K, 200K, 300K and 600K to place
 212K, 417K, 112K and 426K

Ans. First fit strategy



- $P_1 \rightarrow 212K$ is put in 500K partition
 $P_2 \rightarrow 417K$ is put in 600K partition
 $P_3 \rightarrow 112K$ is put in 288K partition
 $P_4 \rightarrow 426K$ has ~~partition~~ to wait

Best-fit strategy:

212K is put in 300K partition

417K is put in 500K partition

112K is put in 200K partition

426K is put in 600K partition

Worst-fit strategy:

212K is put in 600K partition

417K is put in 500K partition

112K is put in 388K partition

426K must wait

→ left out
space =
 $600 - 212$
 $= 388K$



K S INSTITUTE OF TECHNOLOGY

Bangalore – 560109

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

CIE Question paper Scrutiny format

Course Name	OPERATING SYSTEM
Course Code	BCS303
Course Incharge	Mrs.T.Naga Jyothi
Academic year	2023-2024
Semester	3 rd
CIE #	IA – II
Set	A <input checked="" type="checkbox"/> B <input type="checkbox"/>
Scrutiny parameters	
Whether questions are according to assessment plan?	Yes <input checked="" type="checkbox"/> No <input type="checkbox"/> ; If No, Suggestions:
Whether questions prepared are within the covered syllabus?	Yes <input checked="" type="checkbox"/> No <input type="checkbox"/> ; If No, Suggestions:
Whether all questions are mapped to CO/PO properly?	Yes <input checked="" type="checkbox"/> No <input type="checkbox"/> ; If No, Suggestions:
Whether questions framed are according to Blooms level?	Yes <input checked="" type="checkbox"/> No <input type="checkbox"/> ; If No, Suggestions:
Whether marks distribution for each question are correct?	Yes <input checked="" type="checkbox"/> No <input type="checkbox"/> ; If No, Suggestions:
Whether questions paper follows the format displayed?	Yes <input checked="" type="checkbox"/> No <input type="checkbox"/> ; If No, Suggestions:
Difficulty level	Very High <input type="checkbox"/> High <input checked="" type="checkbox"/> Moderate <input type="checkbox"/> Low <input type="checkbox"/>
Percentage of Similarity questions in Set A & B	Nil
Final decision	Accepted without corrections <input checked="" type="checkbox"/> Accepted with minor corrections <input type="checkbox"/> Not accepted <input type="checkbox"/>

Signature with date
of CIE Question paper setter

Name and Signature with date
of CIE Question paper Scrutiniser



K.S.INSTITUTE OF TECHNOLOGY, BANGALORE-560109

Department of Computer & Communication Engineering

THIRD INTERNAL TEST 2023-24 ODD SEMESTER

SET-A

SCHEME AND SOLUTION

Degree : B.E

Semester : III

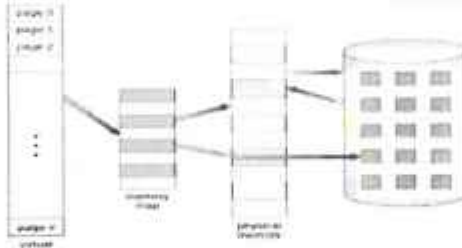
Branch-Stream: CCE

s

Course type/Code : BCS303

Course Title : Operating System

Max Marks : 25

Q.N O.	POINTS	MARKS
1(a).	<p>Identify the usage of virtual memory and mention its advantages?</p> <ul style="list-style-type: none">• Virtual memory is a technique that allows for the execution of partially loaded process.• Advantages:<ul style="list-style-type: none">▪ A program will not be limited by the amount of physical memory that is available user can able to write in to large virtual space.▪ Since each program takes less amount of physical memory, more than one program could be run at the same time which can increase the throughput and CPU utilization.▪ Less i/o operation is needed to swap or load user program in to memory. So each user program could run faster. 	(2+3M)

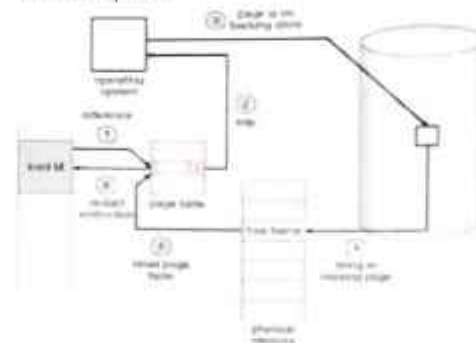
2(a) demand paging ? Explain the steps in handling page fault using appropriate diagram

Ans:

1. Demand paging is similar to paging system with swapping when we want to execute a process we swap the process in to memory otherwise it will not be loaded in to memory.

2. A swapper manipulates the entire processes, where as a pager manipulates individual pages of the process.

- Bring a page into memory only when it is needed
- Less I/O needed
- Less memory needed
- Faster response

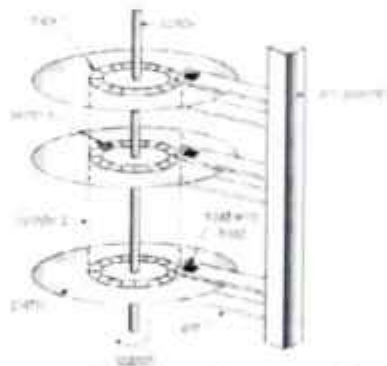


2+3M

3(a) Model the moving head disk mechanism and define the following terms (i) Seek time (ii) Rotational delay (iii) Disk Bandwidth.

4+2M

- **Seek Time:-** Seek time is the time required to move the disk arm to the required track.



- **Rotational Latency (Rotational Delay):-** Rotational latency is the time taken for the disk to rotate so that the required sector comes under the r/w head.
- **Positioning time or random access time** is the summation of seek time and rotational delay.
- **Disk Bandwidth:-** Disk bandwidth is the total number of bytes transferred divided by total time between the first request for service and the completion of last transfer.
- **Transfer rate** is the rate at which data flow between the drive and the computer.

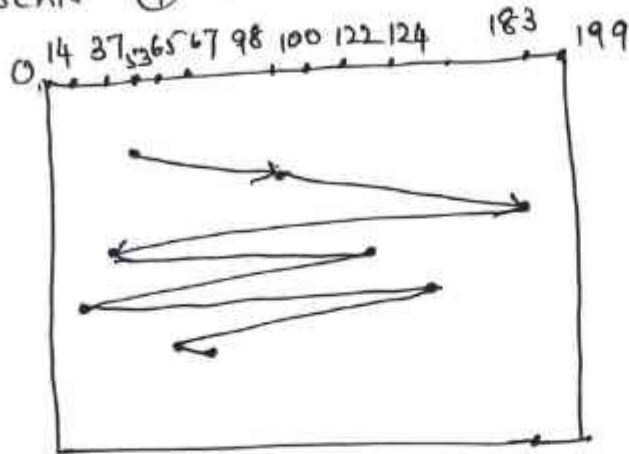
3(b)

Utilize different disk scheduling algorithms and explain it with an example.

Request=[98,183,37,122,14,124,65,67]

Head starts at 53.

- ① FCFS scheduling ② SSTF scheduling
③ SCAN ④ CSCAN ⑤ Look ⑥ Clook



SSTF: find the distance between RW head position and service request. Shortest distance has to be serviced first

3(c)

Identify the usage of swap-space management and explain it with an example.

- 1. The amount of swap space needed on a system can vary depending on the amount of physical memory, the amount of virtual memory it is backing, and the way in which the virtual memory is used. It can range from a few megabytes of disk space to gigabytes.
- 2. The swap space can overestimate or underestimated. It is safer to overestimate than to underestimate the amount of swap space required. If a system runs out of swap space due to underestimation of space, it may be forced to abort processes or may crash entirely. Overestimation wastes disk space that could otherwise be used for files, but it does no other harm.
- 3. Example:
 - 1. Solaris allocates swap space only when a page is forced out of physical memory, rather than when the virtual memory page is first created.
 - 2. Linux is similar to Solaris in that swap space is only used for anonymous memory or for regions of memory shared by several processes. Linux allows one or more swap areas to be established.

4(a)

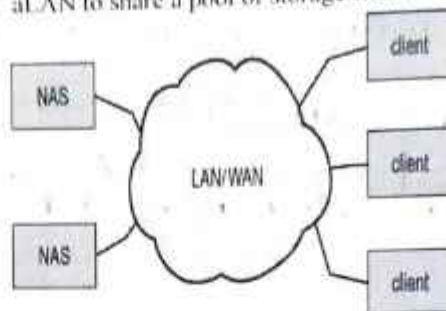
Model the NAS architecture with a neat diagram and explain it in detail?

Ans

Network-Attached Storage

1. A network-attached storage (NAS) device is a special-purpose storage system that is accessed remotely over a network as shown in the figure.
2. Clients access network-attached storage via a remote-procedure-call interface. The remote procedure calls (RPCs) are carried via TCP or UDP over an IP network usually the same local-area network (LAN) carries all data traffic to the clients.

3. Network-attached storage provides a convenient way for all the computers on a LAN to share a pool of storage files.



4(b) **Make use of** access matrix method and explain how to provide system protection.

ACCESS MATRIX

- Our model of protection can be viewed as a matrix, called an access matrix. It is a general model of protection that provides a mechanism for protection without imposing a particular protection policy.
- The rows of the access matrix represent domains, and the columns represent objects.
- Each entry in the matrix consists of a set of access rights.
- The entry $\text{access}(i,j)$ defines the set of operations that a process executing in domain D_i can invoke on object O_j .

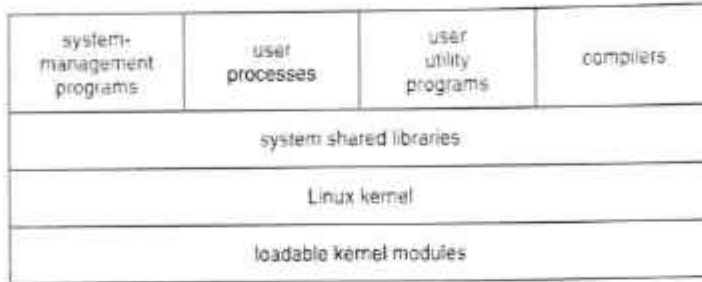
domain \ object	F_1	F_2	F_3	printer
D_1	read		read	
D_2				print
D_3		read	execute	
D_4	read write		read write	

In the above diagram, there are four domains and four objects—three files (F_1 , F_2 , F_3) and one printer. A process executing in domain D_1 can read files F_1 and F_3 . A process executing in domain D_4 has the same privileges as one executing in domain D_1 ; but in addition, it can also write onto files F_1 and F_3 .

When a user creates a new object O_j , the column O_j is added to the access matrix with the appropriate initialization entries, as dictated by the creator.

40

- Identify** the Linux kernel module and explain it in detail.
The **kernel** is responsible for maintaining the important abstractions of the operating system.
- Kernel code executes in kernel mode with full access to all the physical resources of the computer.
 - All kernel code and data structures are kept in the same single address space.



Signature of Course Incharge

Signature of HOD



K S INSTITUTE OF TECHNOLOGY

Bangalore – 560109

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

CIE Question paper Scrutiny format

Course Name	OPERATING SYSTEM
Course Code	BCS303
Course Incharge	Mrs.T.Naga Jyothi
Academic year	2023-2024
Semester	3 rd
CIE #	IA – I(I)
Set	A <input type="checkbox"/> B <input checked="" type="checkbox"/>
Scrutiny parameters	
Whether questions are according to assessment plan?	Yes <input checked="" type="checkbox"/> No <input type="checkbox"/> ; If No, Suggestions:
Whether questions prepared are within the covered syllabus?	Yes <input checked="" type="checkbox"/> No <input type="checkbox"/> ; If No, Suggestions:
Whether all questions are mapped to CO/PO properly?	Yes <input checked="" type="checkbox"/> No <input type="checkbox"/> ; If No, Suggestions:
Whether questions framed are according to Blooms level?	Yes <input checked="" type="checkbox"/> No <input type="checkbox"/> ; If No, Suggestions:
Whether marks distribution for each question are correct?	Yes <input checked="" type="checkbox"/> No <input type="checkbox"/> ; If No, Suggestions:
Whether questions paper follows the format displayed?	Yes <input checked="" type="checkbox"/> No <input type="checkbox"/> ; If No, Suggestions:
Difficulty level	Very High <input type="checkbox"/> High <input checked="" type="checkbox"/> Moderate <input type="checkbox"/> Low <input type="checkbox"/>
Percentage of Similarity questions in Set A & B	Nil
Final decision	Accepted without corrections <input checked="" type="checkbox"/> Accepted with minor corrections <input type="checkbox"/> Not accepted <input type="checkbox"/>

Signature with date
of CIE Question paper setter

Name and Signature with date
of CIE Question paper Scrutiniser



THIRD INTERNAL TEST 2023-24 ODD SEMESTER

SET-B

SCHEME AND SOLUTION

Degree : B.E

Semester : III

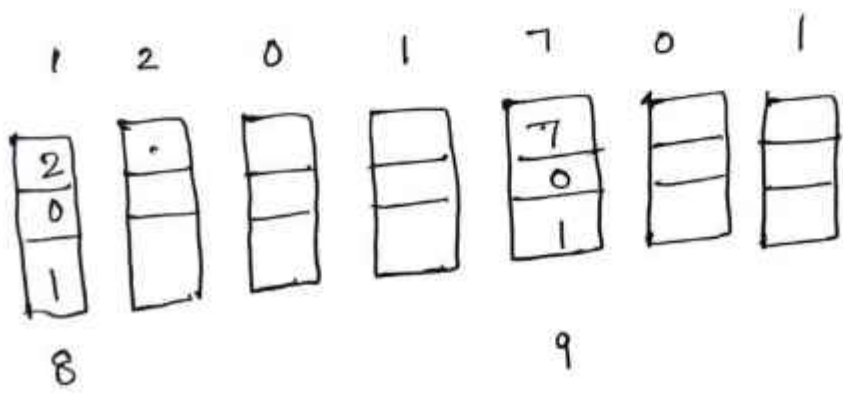
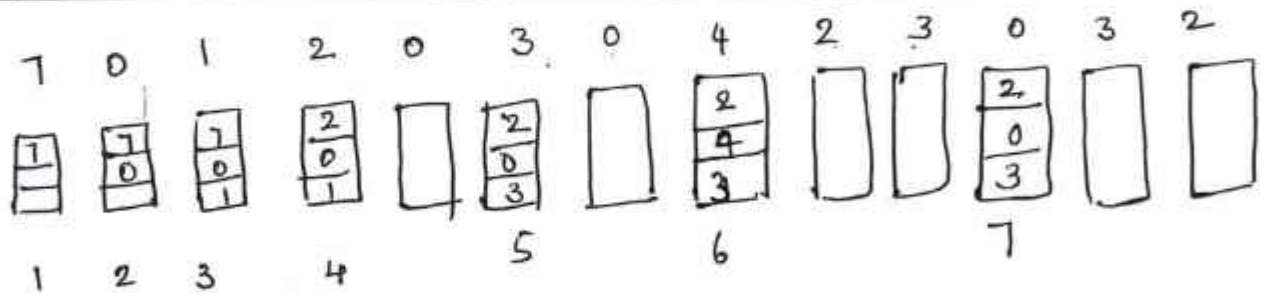
Branch-Stream: CCE

Course type/Code : BCS303

Course Title : Operating System

Max Marks : 25

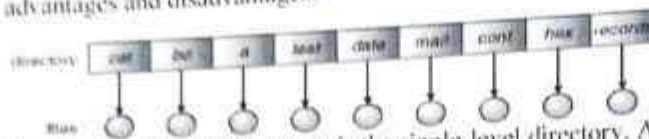
Qno.		Marks
1.	<p>Solve and find the effective access time , given the memory access time is 200 nano seconds and average page fault service time is 8 milliseconds</p> <p>Effective access time = $(1-P)*ma + P*pagefault = 8.2 \text{ milliseconds}$</p>	1+4 M
2.	<p>Consider the page reference string 7,0,1,2,0,3,0,4,2,3,0,3,2,1,2,0,1,7,0,1 for a memory with 3 frames, Solve and determine the number of page faults using Optimal replacement algorithm</p>	5M



No. of Page faults = 9

3(a) Identify the difference between single level directory and two level directory? Mention its advantages and disadvantages.

Ans:



The simplest directory structure is the single-level directory. All files are contained in the same directory, which is easy to support and understand.

Directory structure is single, uniqueness of file name has to be maintained, which is difficult when there are multiple users.

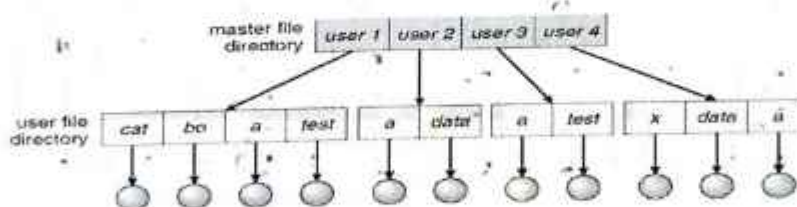
Even a single user on a single-level directory may find it difficult to remember

The names of all the files as the number of files increases.

It is not uncommon for a user to have hundreds of files on one computer system and an equal number of additional files on another system. Keeping track of so many files is a daunting task.

1. Two-Level Directory

- In the two-level directory structure, each user has its own **user file directory (UFD)**. The UFDs have similar structures, but each lists only the files of a single user.
- When a user refers to a particular file, only his own UFD is searched. 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.
- 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.



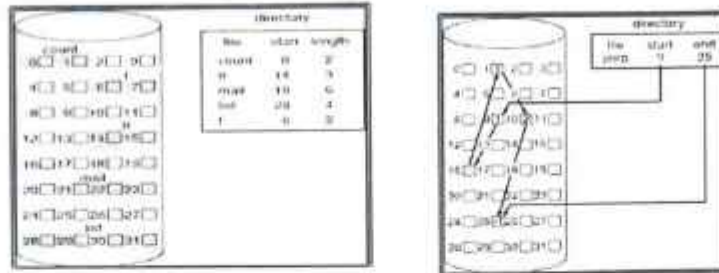
- Advantage:
 - No filename-collision among different users.
 - Efficient searching.
- Disadvantage:
 - Users are isolated from one another and can't cooperate on the same task.

3(b) **Make use of** contiguous and Linked disk space allocation methods and explain it with a neat diagram

Ans: **Contiguous allocation:**

1. Requires that each file occupy a set of contiguous blocks on the disk
2. Accessing a file is easy – only need the starting location (block #) and length (number of blocks)

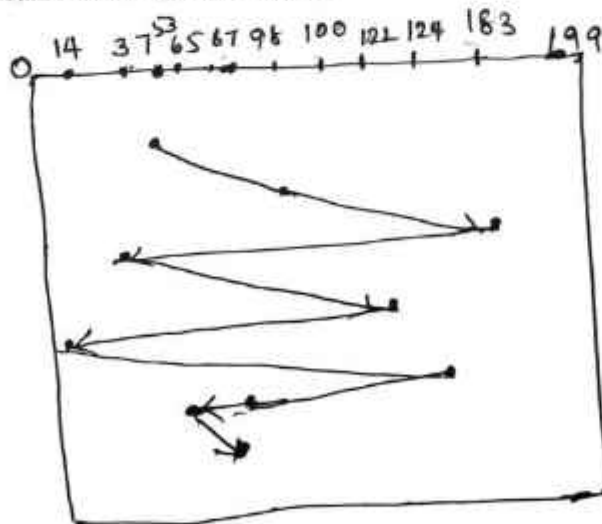
3. Contiguous allocation of a file is defined by the disk address and length (in block units) of the first block. If the file is n blocks long and starts at location b , then it occupies blocks $b, b + 1, b + 2, \dots, b + n - 1$. The directory entry for each file indicates the address of the starting block and the length of the area allocated for this file.
4. Accessing a file that has been allocated contiguously is easy. For sequential access, the file system remembers the disk address of the last block referenced and when necessary, reads the next block. For direct access to block i of a file that starts at block b , we can immediately access block $b + i$. Thus, both sequential and direct access can be supported by contiguous allocation.



Linked Allocation:

- Solves the problems of contiguous allocation
- Each file is a linked list of disk blocks: blocks may be scattered anywhere on the disk
- The directory contains a pointer to the first and last blocks of a file
- Creating a new file requires only creation of a new entry in the directory

30) A drive has 200 cylinders 0 to 199. Head starts at 53 to serve the request queue. 98, 183, 37, 122, 14, 124, 65, 67. Solve and Draw disk head schedule diagram and explain for FCFS, SSTF, C-SCAN and C-LOOK.



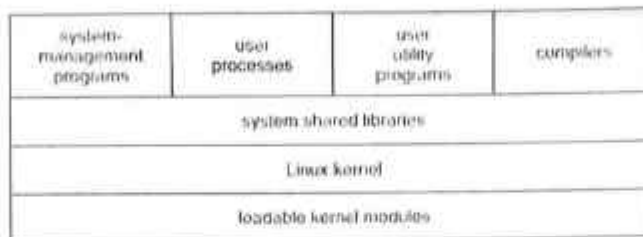
SSTF: Based on Shortest seek time

CSCAN: It services the request to one end and moves towards another end.

4(a)

Identify the components of a Linux system
Components of a Linux System

- Like most UNIX implementations, Linux is composed of three main bodies of code; the most important distinction between the kernel and all other components.
1. The **kernel** is responsible for maintaining the important abstractions of the operating system.
 - Kernel code executes in kernel mode with full access to all the physical resources of the computer.
 - All kernel code and data structures are kept in the same single address space.



2. The **system libraries** define a standard set of functions through which applications interact with the kernel, and which implement much of the operating-system functionality that does not need the full privileges of kernel code.
3. The **system utilities** perform individual specialized management

4(b)

Identify the usage of fork() and exec() system calls in Linux and a program to implement these system calls.

Ans

Fork() is used to create a new process
Exec() is used to execute a new program under that process.

7. Develop a C program to implement process system calls fork(), exec(), process and terminate process.

```

#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <sys/wait.h>

int main() {
    pid_t child_pid;
    int status;

    // Create a child process using fork()
    child_pid = fork();

    if (child_pid == -1) {
        perror("Fork failed");
        return 1;
    }

    if (child_pid == 0) {
        // This code is executed by the child process
        printf("Child process: My PID is %d\n", getpid());

        // Execute a program by the child process using exec()
        char *argv[] = {"ls", "-l", "/etc/passwd"};
        if (execvp(argv[0], argv) == -1) {
            perror("exec failed");
            return 1;
        }
    } else {
        // This code is executed by the parent process
        printf("Parent process: My PID is %d\n", getpid());

        // Wait for the child process to terminate using wait()
        wait(&status);

        if (WEXITSTATUS(status) != 0) {
            printf("Child process terminated with status %d\n", WEXITSTATUS(status));
        } else if (WTERMSIG(status) != 0) {
            printf("Child process terminated due to signal %d\n", WTERMSIG(status));
        }
    }

    return 0;
}

```

40

Identify the usage of disk formatting and explain the following terms

- (i) Boot block (ii) Bad block with a diagram

The process of dividing the disk into sectors and filling the disk with a special data structure is called low-level formatting. Sector is the smallest unit of area that is read/written by the disk controller. The data structure for a sector typically consists of a header, a data area (usually 512 bytes in size) and a trailer. The header and trailer contain information used by the disk controller, such as a sector number and an error-correcting code (ECC).

When the controller writes a sector of data during normal I/O, the ECC is updated with a value calculated from all the bytes in the data area. When a sector is read, the ECC is recalculated and is compared with the stored value. If the stored and calculated numbers are different, this mismatch indicates that the data area of the sector has become corrupted and that the disk sector may be bad.

Most hard disks are low-level-formatted at the factory as a part of the manufacturing process. This formatting enables the manufacturer to test the disk and to initialize the mapping from logical block numbers to defect-free sectors on the disk.

When the disk controller is instructed for low-level-formatting of the disk, the size of data block of all sector sit can also be told how many bytes of data space to leave between the header and trailer of all sectors. It is of sizes, such as 256, 512, and 1,024 bytes. Formatting a disk with a larger sector size means that fewer sectors can fit on each track; but it also means that fewer headers and trailers are written on each track and more space is available for user data.

The operating system needs to record its own data structures on the disk. It does so in two steps, i.e., Partition and logical formatting.

1. **Partition** – is to partition the disk into one or more groups of cylinders. The operating system can treat each partition as though it were a separate disk. For instance, one partition can hold a copy of the operating system's executable code, while another holds user files.
2. **Logical formatting (or creation of a file system)** - Now, the operating system stores the initial file-system data structures onto the disk. These data structures may include maps of free and allocated space (a FAT or modes) and an initial empty directory.

Boot block:

When a computer is switched on or rebooted, it must have an initial program to run. This is called the bootstrap program.

The bootstrap program –

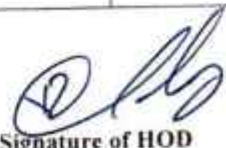
- Initializes the CPU registers, device controllers, main memory, and then starts the operating system.
- Locates and loads the operating system from the disk
- Jumps to beginning the operating-system execution.

Bad Blocks

Disks are prone to failure of sectors due to the fast movement of r/w head. Sometimes, the whole disk will be changed. Such group of sectors that are defective are called as **bad blocks**.



Signature of Course Incharge



Signature of HOD