Enable | Enlighten | Enrich
(Deemed to be University)
(Under Section 3 of UGC Act 1956)

KARPAGAM ACADEMY OF HIGHER EDUCATION

(Deemed to be University Established Under Section 3 of UGC Act 1956)

Coimbatore – 641 021.

LECTURE PLAN
DEPARTMENT OF COMPUTER APPLICATIONS

STAFF NAME: Dr.K.PRATHAPCHANDRAN
SUBJECT NAME: Operating System:Linux **SUB.CODE:**16MMU0404B
SEMESTER: II **CLASS:** II B.Sc Maths

S.No	Lecture Duration Period	Topics to be Covered	Support Material/Page Nos
UNIT-I			
1	1	Introduction -Mainframe systems Desktop Systems – Multiprocessor systems	T1:3-13
2	1	Distributed systems – real time systems.	T1:14-19
3	1	Tutorial –I	-
4	1	Process: - Process concepts – Operation on process – cooperation process	T1:95-109
5	1	Inter process Communication - Mutual Exclusion	T1:19-125
6	1	Tutorial –II	-
7	1	Critical sections	T1:191
8	1	Primitives	W1
9	1	Tutorial –III	-
10	1	Semaphores	T1:201-205
11	1	Deadlock: System Model, Deadlock characterization	T1:243-247
12	1	Tutorial –IV	-
13	1	Deadlock prevention	T1:250-253
14	1	Avoidance,Detection	T1:253-264
15	1	Tutorial –V	-
16	1	Recovery from deadlock	T1:264,W1

17	1	Tutorial –V	-
18	1	Recapitulation and Discussion of Important questions	-
Total No of Hours Planned For Unit I=18			
UNIT-II			
1	1	Storage management: Memory Management	T1:274-279
2	1	Swapping	T1:280-283
3	1	Tutorial I	-
4	1	Contiguous memory allocation	T1:283-286
5	1	Paging	T1:287-300
6	1	Tutorial II	-
7	1	segmentation – segmentation with paging	T1:303-311
8	1	Virtual memory	T1:317
9	1	Tutorial III	-
10	1	Virtual storage organization	T1:318
11	1	Demand Paging	T1:320-327
12	1	Tutorial IV	-
13	1	Process Creation	T1:328-330
14	1	Tutorial V	-
15	1	Page replacement	T1:330-340
16	1	Tutorial VI	-
17	1	Thrashing	T1:348-350
18	1	Recapitulation and Discussion of Important questions	
Total No of Hours Planned For Unit II=18			
UNIT – III			
1	1	Processor Scheduling	T1:99-100
2	1	preemptive scheduling	W1
3	1	Tutorial I	-

4	1	Scheduling Criteria	T1:155-156
5	1	Scheduling Algorithms	T1:157-160
6	1	Tutorial II	-
7	1	FCFS- SJF	T1:160-162
8	1	Priority – RoundRobin	T1:162,163
9	1	Tutorial III	-
10	1	Multilevel Queue – Multilevel Feedback Queue	W1
11	1	Algorithm evaluation	T1:172-177
12	1	Tutorial IV	-
13	1	Multiprocess schedule: Real time schedule	W1
14	1	Tutorial V	-
15	1	Deterministic Modeling	W1
16	1	Tutorial VI	-
17	1	Queue Model, Simulation	W1
18	1	Recapitulation and Discussion of Important questions	-
Total No of Hours Planned For Unit III=18			
UNIT-IV			
1	1	File systems	T1:371-375
2	1	File System Concepts	T1:375-378
3	1	Tutorial I	-
4	1	Access Methods – Directory structure	T1:379-392
5	1	File Sharing, Allocation Methods	T1:395-400
6	1	Tutorial II	-
7	1	Free space management	T1:430-432
8	1	Efficiency and performance	T1:433-437
9	1	Tutorial III	-
10	1	Recovery Disk Performance Optimization	T1:437-438

11	1	Introduction – Disk structure	T1:491
12	1	Tutorial IV	-
13	1	Disk structure	T1:491-492
14	1	Disk scheduling	T1:492-498
15	1	Tutorial V	-
16	1	Disk management.	T1:498-502
17	1	Tutorial VI	-
18	1	Recapitulation and Discussion of Important questions	-
Total No of Hours Planned For Unit III=18			
UNIT-IV			
1	1	Linux-The Operating System: Linux History, Linux features, Linux distributions	T1:695-700
2	1	Linux's relationship to Unix, Overview of Linux Architecture	T1:702-712
3	1	Tutorial I	-
4	1	Installation, Start up scripts, system process (an overview), Linux Security	T1:721-735
5	1	The Ext2 and Ext3 File Systems: General characteristics of the Ext3 File System, File permissions	T1:735-740
6	1	Tutorial II	-
7	1	User Management: Types of users, the powers of Root, Managing users (adding and deleting) : using the command line and GUI Tools	T1:740-755
8	1	Tutorial III	-
9	1	File and Directory management, system calls for files process management	T1:755-765
10	1	Tutorial IV	-
11		IPC:Pipes, FIFOs, System V IPC, Message Queues	R1:625-640
12	1	Tutorial V	-

13	1	Memory Management, Library	R1:645-661
14	1	Tutorial VI	-
15	1	Recapitulation and Discussion of Important questions	-
16	1	Discussion of Previous ESE Questions	-
17	1	Discussion of Previous ESE Questions	-
18	1	Discussion of Previous ESE Questions	-
Total No of Hours Planned for unit V=12			
Total Planned Hours:**60**			

Text Book:

1. Silberschatz Galvin Gagne. (2012). Operating system concepts, Ninth Edition, Wiley India (pvt), Ltd, New Delhi.

References

1. Deitel H.M. (2005). Operating systems, Third Edition, Addision Wesley Publication, New Delhi.

2. Pramod Chandra P. Bhatt. (2007). An Introduction to Operating Systems, Second Edition, Prentice Hall India, New Delhi.

3. Tanenbaum Woodhull. (2005) . Operating Systems., Second Edition, Pearson Education (LPE) , New Delhi.

4. William Stallings. (2010). Operating Systems internals and Design Principles, Sixth Edition, Prentice Hall India, New Delhi.

5. Arnold Robbins., (2008) ., Linux Programming by Examples The Fundamentals, Second Edition., Pearson Education,.

6. Cox K, (2009).Red Hat Linux Administrator's Guide,PHI.

7. Stevens R., (2009). UNIX Network Programming, Third Edition.,PHI.

8. Sumitabha Das, (2009).Unix Concepts and Applications, Fourth Edition., TMH.

9. Ellen Siever, Stephen Figgins, Robert Love, Arnold Robbins, (2009) . Linux in a Nutshell, Sixth Edition,O'Reilly Media.
10. Neil Matthew, Richard Stones, Alan Cox,(2004) Beginning Linux Programming,Third Edition.

WEBSITES

1. www.cs.columbia.edu/~nieh/teaching/e6118_s00/
2. www.clarkson.edu/~jnm/cs644
3. pages.cs.wisc.edu/~remzi/Classes/736/Fall2002/

Karpagam Academy of Higher Education

Department of Mathematics

Academic Year : 2016-2019

Semester : III | **Class : II B.Sc Maths**

Subject : Operating Systems: Linux | **Subject Code: 16MMU404B**

UNIT -I

S.NO	Question	Opt1	Opt2	Opt3	Opt4	Answer
1	The queue has maximum length 0; thus, the link ca	Zero capacity	Bounded capacity	Unbounded capacity	synchronous	Zero capacity
2	A page fault occurs	when the page is not in the memory	when the pageis the memory	when the process entered the blocked state	when the process is in the ready state	when the page is not in the memory
3	Let S and Q be two semaphores initialized to 1,where Po and p1 processes the following statements wait(S);wait(Q);--:signal(S) and signal(Q) and wait(Q);wait(S);_---;signal(Q);signal(S);respectively.The above sitaution depicts a	semaphore	deadlock	signal	interrupt	deadlock
4	computer hardware and acts as an intermediary between the computer user and the computer hardware.	hardware acceleration	Operating System	compiler	logical transcation	Operating System
5	_____manages the execution of user programs t	resource allocator	work station	main frame	control program	control program
6	system is the one program running at all times on the computer usually called___________________	bootstrap	firmware	kernel	read-only memory	kernel
7	computer system can be divided into ___________	2	3	4	5	2
8	____were the first computers used to tackle ma	Mainframe computer system	Mainframe computer service	multiframe computer system	multiframe computer service	Mainframe computer system

9	____________system is the collection of computer that act,work and appear as one large computer size of a distributed system.	distributed	symmetric	asymmetric	multiple	distributed
10	________________contains the address of an instruction to be fetched from memory	Program counter (]	Instruction register	Control registers	Status registers	Instruction register (IR)
11	____________________contains the instruction mo	Program counter (]	Instruction register	Control registers	Status registers	Program counter (PC)
12	The kernel is a____________________	memory manager	resource manager	file manager	directory manager	resource manager
13	Main function of shared memory is________________________	to use primary memory efficently	to do intra process communication	to do inter process communication	to do other process communication	to do inter process communication
14	Disk scheduling includes deciding_____________________	which should be accessed next	order in which disk access requests must be	the physical location of the file	the logical location of the file	order in which disk access requests must be serviced
15	Memory protection is normally done by _____________________	the processor and the associated	the operating system	the compiler	the user program	the processor and the associated hardware
16	_____________controls the nodes hardware	ubiquitous minimal kernel	ubiquitous maximal kernel	ubiquitous normal kernel	high level system management	ubiquitous minimal kernel
17	In Banker's Algorithm _____________________ co	mutual-exclusion	Time-sharing	race condition	cooperating processes	mutual-exclusion
18	_____________is the process of actually determinin	Deadlock detection	Deadlock prevention	fault tolerant	process synchronization	Deadlock detection
19	Once a system has become ____________ the deadlock must be broken by removing one or more of the necessary conditions	deadlocked	race condition	mutual-exclusion	cooperating processes	deadlocked
20	________________ is also known as parallel system	Multiprocessor systems	desktop systems	Time sharing systems	Multiprogrammed systems	Multiprocessor systems
21	_______________operating systems are even more (	Time-sharing	desktop systems	Multiprogrammed systems	Multiprocessor systems	Time-sharing
22	___________ operating system keeps several jobs in memory simultaneously.	Time-sharing	desktop systems	Multiprogrammed systems	Multiprocessor systems	Multiprogrammed systems

23	____________can save more money than multiple s	Multiprocessor systems	desktop systems	Time sharing systems	Multiprogrammed systems	Multiprocessor systems
24	This ability to continue providing service proportio	fault tolerant.	graceful degradation	Economy of scale	Increased throughput	graceful degradation
25	Systems designed for graceful degradation are also	graceful degradation	Economy of scale	fault tolerant	Increased throughput	fault tolerant
26	The most common multiple-processor systems now	symmetric multiprocessing	asymmetric multiprocessing	multithreading	multiprogramming	symmetric multiprocessing
27	Another form of a special-purpose operating systen	real-time system	distributed operating system	Process states	multiframe computer system	real-time system
28	The assignment of the CPU to the first process on t	graceful degradation	Time-sharing	dispatching	Multiprocessor systems	dispatching
29	The manifestation of a process in an operating syste	Process state transitions	process control block	child process	cooperating processes	process control block
30	A process may spawn a new process. If it does, the creating process is called the parent process and the created process is called the	child process	Process state transitions	Process state transitions	process control block	child process
31	The communication is direct or indirect, messages exchanged by communicating processes reside in a temporary queue known as	Buffering	synchronization	asynchronization	communication link	Buffering
32	The message-passing facility in Windows 2000 is c	MUTUAL EXCLUSION	Buffering	local procedure call facility	CRITICAL SECTIONS	local procedure call facility
33	______________can assume only the value 0 or the	Binary semaphores	Counting semaphores	semaphore operations	normal semaphores	Binary semaphores
34	Semophores are used to solve the problem of	race condition	process synchronization	mutual exclusion	belady problem	mutual exclusion
35	For multiprogramming operating system	special support from processor is essential	special support from processor is not essential	cache memory is essential	cache memory is not essential	special support from processor is not essential
36	Which is single user operating system	MS-DOS	UNIX	XENIX	LINUX	MS-DOS

37	Which operating system reacts in the actual time	Batch system	Quick response system	Real time system	Time sharing system	Real time system
38	In real time OS, which is most suitable scheduling scheme	round robin	FCFS	pre-emptive scheduling	random scheduling	pre-emptive scheduling
39	Dispatcher function is to	put tasks in I/O wait	schedule tasks in processor	change task priorities	Multitasking	put tasks in I/O wait
40	Multiprogramming systems	Are easier to develop than single programming	Execute each job faster	Execute more jobs in the same time	Are used only on large main frame computers	Execute more jobs in the same time
41	Operating system is	A collection of hardware components	A collection of input output devices	A collection of software routines	last entered the queue	A collection of software routines
42	Semaphores function is to	synchronize critical resources to prevent	synchronize processes for better CPU utilization	used for memory management	may cause a high I/O rate	synchronize critical resources to prevent deadlock
43	Which operating system use write through catches	UNIX	XENIX	ULTRIX	DOS	DOS
44	Which process is known for initializing a microcomputer with its OS	cold booting	boot recording	booting	warm booting	booting
45	Four necessary conditions for deadlock are non pre-emption, circular wait, hold and wait and	mutual exclusion	race condition	buffer overflow	multiprocessing	mutual exclusion
46	Remote computing services involves the use of timesharing and	multiprocessing	interactive processing	batch processing	real time processing	batch processing
47	A series of statements explaining how the data is to be processed is called	instruction	compiler	program	interpretor	program
48	Banker's algorithm deals with	deadlock prevention	deadlock avoidance	deadlock recovery	mutual exclusion	deadlock avoidance
49	Which is non pre-emptive	Round robin	FIFO	MQS	MQSF	FIFO
50	A hardware device which is capable of executing a sequence of instructions, is known as	CPU	ALU	CU	Processor	Processor

51	Distributed systems should	high security	have better resource sharing	better system utilization	low system overhead	have better resource sharing
52	Which of the following is always there in a computer	Batch system	Operating system	Time sharing system	Controlling system	Operating system
53	Which of following is not an advantage of multiprogramming	increased throughput	shorter response time	ability to assign priorities of jobs	decreased system overload	decreased system overload
54	In which of the following usually a front end processor is used	Virtual storage	Timesharing	Multiprogamming	Multithreading	Timesharing
55	Remote computing services involves the use of	multiprocessing	multiprogramming	batch processing	real time processing	batch processing
56	Banker's algorithm for resource allocation deals with	deadlock prevention	deadlock aviodance	deadlock recovery	circular wait	deadlock aviodance
57	When did IBM released the first version of its disk operating system DOS version 1.0	1981	1982	1983	1984	1981
58	The queue has finite length n; thus, at most n mess	Zero capacity	Bounded capacity	Unbounded capacity	multiprocessing	Bounded capacity
59	The queue has potentially infinite length; thus, any	Zero capacity	Bounded capacity	Unbounded capacity	multiprocessing	Unbounded capacity
60	all others from doing so simultaneously and this is called	mutual exclusion	multiprocessing	real time processing	multiprogramming	mutual exclusion

Karpagam Academy of Higher Education

Department of Mathematics
Academic Year : 2016-2019

Semester : III **Class : II B.Sc Maths**
Subject : Operating Systems: Linux **Subject Code: 16MMU404B**

S.NO	Question	Opt1	Opt2	Opt3	Opt4	Answer
UNIT -II						
1	Memory is array of ____________	bytes	circuits	ics	ram	bytes
2	CPU fetches instructions from _______	memory	pendrive	dvd	cmos	memory
3	Program must be in _________________	memory	pendrive	dvd	cmos	memory
4	Collection of process in disk forms______	input queue	output queue	stack	circle	input queue
5	Address space of computer starts at __________	0000	4444	3333	2222	0000
6	If process location is found during compile time	absolute	relative	approximate	more or less	absolute
7	Address generated by CPU is _______________	logical	physical	direct	indirect	logical
8	Logical address can be also called as _________	physical	virtual	direct	indirect	virtual
9	Run time mapping is done using _____________	MMU	CPU	CU	IU	MMU
10	In address binding base register is also called as	relocation register	memory register	hard disk	pendrive	relocation register
11	Better memory space is utilized using _________	dynamic loading	dynamic linking	registers	array of words	dynamic loading
12	__________________ routine is never loaded in	unused	used	regular	recursive	unused
13	Some operating systems support only _________	static	dynamic	temporary	interruptive	static
14	________________ is a code that locates library ro	stub	dll	recursive routine	exe file	stub
15	_____________ can be used to manage large mem	overlays	swapping	roll in and out	libraries	overlays
16	________________ error is raised in memory	addressing	swapping	dynamic	index	addressing
17	Set of ____________ are scattered throughout the	holes	gaps	free space	words	holes
18	______________ can be internal and external	fragmentation	merging	grouping	fixing	fragmentation
19	_____________ is used to divide a process into f	paging	segmentation	sp	swapping	paging
20	In paging physical memory is divided into _____	frames	pages	segments	bytes	frames
21	In paging virtual memory is divided into ______	frames	pages	segments	bytes	pages
22	______________ is first of virtual address in pag	page number	segment number	frame number	offset	page number

23	______________ is second part of virtual address	page number	segment number	frame number	offset	offset
24	Page mapping entries are found in ___________	page table	segment table	hash table	pointing table	page table
25	Page size is defined by _______________	hardware	software	os	kernel	hardware
26	______________ is first in mapping of virtual to	direct	associate	direct & associativ	pointing	direct
27	______________ is second in mapping of virtual	direct	associate	direct & associativ	pointing	associate
28	______________ is third in mapping of virtual to	direct	associate	direct & associativ	pointing	direct & associative
29	_____________ is used to divide a process into v	paging	segmentation	sp	swapping	segmentation
30	In segmentation virtual memory is divided into _	frames	pages	segments	bytes	segments
31	____________ view is supported in segmentation	user	system	cpu	manager	user
32	_____________ is format for segmentation virtua	(s,d)	(p,d)	(v,d)	(k,d)	(s,d)
33	_____________ is the first element in segment ta	limit	base	offset	page number	limit
34	_____________ is the second element in segmen	limit	base	offset	page number	base
35	Addressing in segmentation is similar as ______	direct	associate	direct & associativ	pointing	direct
36	How many elements are there in segmentation a	1	2	3	4	3
37	_____________ is organization in physical memo	frames	pages	segments	bytes	frames
38	________________ memory is used to manage in	virtual	physical	rom	eprom	virtual
39	virtual memory abstracts _________ memory	virtual	eerom	main	eprom	main
40	_______________ reasons are there for existence	1	2	3	4	3
41	_______________ benefits are there from virtual	1	2	3	4	3
42	Virtual memory is commonly implemented by _	demand	bargain	quarrel	order	demand
43	________________ fault occurs when desired pa	page	segment	pages	segments	page
44	_______________ table is used in demand pagin	page	segment	pages	segments	page
45	_______________ methods are there for process	1	2	3	4	2
46	_______________ method implements partial sh	copy on write	memory mapping	paging	segmentation	copy on write
47	_______________ is done for page fault	replacement	swapping	logging	locking	replacement
48	_______________ is unrealizable page replacem	optimal	FIFO	LRU	NRU	optimal
49	_______________ is first page replacement algo	optimal	FIFO	LRU	NRU	FIFO
50	_______________ is second page replacement al	optimal	FIFO	LRU	NRU	optimal
51	_______________ is third page replacement algo	optimal	FIFO	LRU	NRU	NRU
52	_______________ is associated with each page i	label	index	number	identity	label
53	_______________ labelled page replaced in opti	highest	lowest	moderate	below average	highest

54	_______________ end page is removed in fifo al	rear	head	top	bottom	head
55	Modified version of fifo algorithm gives ______	1	2	3	4	2
56	______________ is called as high paging activity	thrashing	smashing	mocking	breaking	thrashing
57	______________ occurs frequently during thrash	page fault	segment fault	memory fault	address fault	page fault
58	_______________ strategy is used to solve thrasl	working set	pff	lpr algorithm	gpl algorithm	working set
59	_______________ algorithm is used to solve thra	working set	pff	lpr algorithm	gpl algorithm	lpr algorithm
60	_______________ is a basic solution for thrashin	working set	pff	lpr algorithm	gpl algorithm	pff

Karpagam Academy of Higher Education

Department of Mathematics

Academic Year : 2016-2019

Semester : III **Class : II B.Sc Maths**

Subject : Operating Systems: Linux **Subject Code: 16MMU404B**

S.NO	Question	Opt1	Opt2	Opt3	Opt4	Answer
			UNIT-III			
1	_______ is the basis of multiprogrammed operating system.	RR scheduling	Self Scheduling	CPU sceduling	throughput	CPU sceduling
2	A ______ is executed until it must wait, typically for the completion of some i/o request	reverse	deadlock avoidance	deadlock	process	process
3	_____ is a fundamental operating system function.	RR	CPU	Scheduling	nonpreemptive	Scheduling
4	Process execution begins with a_____	CPU burst	RR scheduling	SJF scheduling	SRT scheduling	CPU burst
5	The operating system must select one of the processes in the ready queue to be executed by the _________	nonpreemptive	short term scheduler	long term scheduler	low level	short term scheduler
6	When scheduling takes place only under circumstances 1 and 4 called _______	variable class	real time class	priority class	nonpreemptive	nonpreemptive
7	Another component involved in the CPU scheduling function is the ______	central edge	dispatcher	claim edge	graph edge	dispatcher
8	One measure of work is the number of processes completed per time unit called _____	throughput	variable class	real time class	priority class	throughput
9	Which of the following is the simplest scheduling discipline?	FCFS scheduling	RR scheduling	SJF scheduling	SRT scheduling	FCFS scheduling
10	In which scheduling, processes are dispatched according to their arrival time on the ready queue?	FCFS scheduling	RR scheduling	SJF scheduling	SRT scheduling	FCFS scheduling
11	In which scheduling, processes are dispatched FIFO but are given a limited amount of CPU time?	FIFO scheduling	RR scheduling	SJF scheduling	SRT scheduling	RR scheduling

12	Which scheduling is effective in time sharing environments	FIFO scheduling	RR scheduling	SJF scheduling	SRT scheduling	RR scheduling
13	Variable size blocks are called	Pages	Segments	Tables	None	Segments
14	Which scheduling is effective in time sharing environments	FIFO scheduling	RR scheduling	SJF scheduling	SRT scheduling	RR scheduling
15	Which of the following is non-preemptive scheduling?	RR scheduling	SJF scheduling	SRT scheduling	None	SJF scheduling
16	The interval from the time of submission of a process to the time of completion is the ____	Queues	Processor Sharing	Sharing resources	turaround time	turaround time
17	The simplest CPU sceduling algorithm is the	FCS	SJS	FCFS	DFG	FCFS
18	Multiprogramming was made possible by _____________.	input/output units that operate independently of the CPU	operating systems	both (a) & (b) above	neither (a) or (b) above	both (a) & (b) above
19	The SJF algorithm is a special case of the general _______ algorithm	FCS	SJS	Roundrobin	FCSC	Roundrobin
20	_______ scheduling algorithm is designed especially for time sharing systems.	CFS	FSCS	priority	Round Robin	Round Robin
21	The seek optimization strategy in which there is no reordering of the queue is called ___________.	FCFS	SSTF	SCAN	C-SCAN	FCFS
22	A major problem with priority scheduling algorithms is _____	tail	Starvation	time first	time quantum	Starvation
23	If the time quantum is very small the RR aproach is called _________	Queues	Processor Sharing	Sharing resources	Context switching	Processor Sharing
24	The seek optimization strategy in which the disk arm is positioned next at the request (inward or outward) that minimizes arm movement is called _________.	FCFS	SSTF	SCAN	C-SCAN	SSTF
25	If several identical processors are available then _____ can occur.	heterogeneous	homogeneous	load sharing	UMA	load sharing

26	The high priority process would be waiting for a lower priority one to finish is called _____	resources inversion	Priority inversion	priority	Priority inheritance	Priority inversion
27	_____ systems are required to complete a critical task within a guaranteed amount of time.	hard real time	Priority inversion	load sharing	Priority inheritance	hard real time
28	The scheduler than either admits a process guarenteeing that the process will complete on time known as _____	Priority inversion	resources reservation	load sharing	Sharing resources	resources reservation
29	_______ uses the the given algorithm and the system workload to produce a formula.	deterministic modelling	scheduling process	Analaytic evaluation	Queuing model	Analaytic evaluation
30	If no thread is found the dispatcher will execute a special thread called______	variable class	real time class	priority class	idle thread	idle thread
31	Deadlocks can be described more precisely in terms of a directed graph called	resource graph	system graph	system resources allocation graph	request graph	system resources allocation graph
32	______ is th set of methods for ensuring that at atleast one of the necessary condition.	Deadlock prevention	deadlock avoidance	handling deadlock	resource deadlock	Deadlock prevention
33	_____ is possible to construct an algorithm that ensures that the system will never enter the deadlock state.	Deadlock prevention	deadlock avoidance	handling deadlock	resource deadlock	deadlock avoidance
34	A system is in a safe state only if there exists a ______	Safe state	unsafe state	normal	deadlock	Safe state
35	A critical section is a program segment ____________.	which should run in a certain specified amount	which avoids deadlocks	where shared resources are accessed	which must be enclosed by a pair of semaphore operations, P and V	where shared resources are accessed
36	In addition to the request and assignment edges,we introduce a new type of edge called _____	central edge	graph edge	claim edge	system edge	claim edge
37	The deadlock avoidance algorithm are described in next system but is less efficient than the resource allocation graph called ______	Deadlock prevention	deadlock avoidance	bankers algorithm	bankers allocation	bankers algorithm

38	CPU Scheduling is the basis of _________ operating system.	single programmed	multi programmed	multi system	multi disks	multi programmed
39	Scheduling is a fundamental ___________ function.	computer	operating system	system resource	disk	operating system
40	Another component involved in the CPU _______ function is the dispatcher	processing	mathematical	arithmetic	scheduling	scheduling
41	A major problem for priority __________ is starvation.	sort algorithms	scheduling algoriuthms	search algorithms	manage algorithms	scheduling algoriuthms
42	The seek _______ strategy in which there is no reordering of the queue is called SSTF	processing	scheduling	optimization	implementation	optimization
43	The high ___________ would be waiting for a lower priority one to finish is called priority inversion	performance	priority	patent	graph edge	priority
44	A __________ is a program segment where shared resources are accessed.	critical section	sub section	cross section	class section	critical section
45	If no thread is found, the ________ will execute a special thread called idle thread.	degrader	scheduler	dispatcher	redeemer	dispatcher
46	_______ execution begins with a CPU Burst.	Process	Performance	Purge	Put	Process
47	SJF Scheduling is an example for _________ scheduling.	non- preemptive	preemptive	emptive	prescheduling	non- preemptive
48	_______ is the simplest CPU sceduling algorithm.	FCFS	LCFS	FCLS	LCFS	FCFS
49	Segments are called ________ blocks.	equal size	variable class	variable size	big size	variable size
50	________ can be described more precisely in terms of a directed graph.	semaphore	deadlocks	dumplocks	starvation	deadlocks
51	The interval from the time of submission of a process to the time of completion is the ____	Queues	Processor Sharing	Sharing resources	turaround time	turaround time

Karpagam Academy of Higher Education

Department of Mathematics
Academic Year : 2016-2019

Semester : III **Class : II B.Sc Maths**
Subject : Operating Systems: Linux **Subject Code: 16MMU404B**

S.NO	Question	Opt1	Opt2	Opt3	Opt4	Answer
			UNIT -IV			
1	The file system consist of -------------- Distinct parts	2	3	4	5	2
2	A --------------- File is a sequence of character organized into lines	Source	Object	Text	Executable	Text
3	A --------------- File is a sequence of subroutines and functions	Source	Object	Text	Executable	Source
4	The operating system keeps a small table called the --------------- ,containing information about all open files	Show file table	Visible file table	Open file ta	Manage file table	Open file table
5	A file is executed in --------------- extension	External structure	.bat	.mdb	.in	.bat
6	The .bat file is a ----------------containing in ANCII format,command to the operating system	Binary file	Batch file	Text file	Word file	Batch file
7	The file type is used to indicate the ---------------- of the file	.txt	Internal structure	Block structure	Outer structure	Internal structure
8	Information in the file is processed in the order called------------------	Direct access	Sequence access	Dynamic access	Random access	Sequence access
9	A file is made up of fixed length that allows the program to read and write record rapidly in no particular order is called -------------------	Direct access	Sequence access	Dyanamic access	Random access	Direct access

10	Data cannot be written in secondary storage unless written with in a ---------------------	File	Swap space	Directory	Text format	File
11	File attribute consist of ----------------	Name,Type,Content	Name,type,Size	Seperate directory system	Name,identifier	Name,Size,Type,identifier
12	The information about all files is kept in -----------------	swap space	operating system	Name,Size,Type,identifier	Hard disk	Seperate directory system
13	A file is a -------------- type	Abstract	Primitive	Public	Private	Abstract
14	In UNIX Open system call returns -----------------	pointer to the entry in the open file table	pointer to the entry in the system wide table	A file to the process calling it	pointer to the entry in the close file	pointer to the entry in the open file table
15	The open file table has a ------------------- Associated with each file	File content	File permission	open count	Close count	open count
16	The file name is generaly split into which of the two parts -----------------	Name and type	Name and identifier	Name and extension	Extension and type	Name and extension
17	In the sequential access method, information in the file is processed	One disk after the other	One record after the other	One text document after the other	One name after the other	One record after the other
18	Sequential access method ----------------,on random access devices	Works well	Dosen't works well	Works slow	Works normal	Works well
19	The direct access method is based on a ------------- model of a file as---------------- allow random access to any file block	Magnetic tape,magnetic tapes	Tape,Tapes	Disk,Disks	Tape,Disk	Disk,Disks
20	A relative block number is an index relative to ---------------------	The beginning of the file	The end of the file	The last written position in file	Middle of the file	The beginning of the file
21	The index contains ------------------------	Name of all content of file	Pointer to each page	Pointers to the various blocks	Pointer to same page	Pointers to the various blocks
22	The directory can be viewed as a --------------,that translate the file name into their directory entries	Symbol table	Partition	Swap space	Cache	Symbol table

23	In the single level directory: ------------------------	All files are contain in different directories	All files are contained in the same directory	Depend on the operating system	Depend on the file name	All files are contained in the same directory
24	In the single level directory --------------	All directory must have a unique name	All files must have a unique name	All files must have a unique owner	All files must have a different names	All files must have a unique name
25	In the two level directory structure -----------------	Each user has its own user file directory	The system has its own master file directory	)Both a and b	Each user has its different file directory	Both a and b
26	When a user refers to a particular file --------------------	System MFD is searched	His own UFD is searched	Both MFD and UFD are searched	Every directory is searched	Both MFD and UFD are searched
27	The disadvantage of the two level directory structure is that	It does not solve the name collision problem	It solve the name collision problem	It does not isolate users from one another	It isolates users from one another	It isolates users from one another
28	In the tree structure directory -------------------	The tree has the same directory	The tree has the leaf directory	The tree has the root directory	The tree has no directory	The tree has the root directory
29	The three major methods of allocating disk space that are in wide use are ---------------	Contiguous,Linked ,Hashed	Contiguous,Linked,Indexed	Linked,Hashed,Indexed	Contiguous ,Linked	Contiguous,Linked,Indexed
30	In Contiguous allocation ----------------	each file must occupy a set of contiguous block on the disk	Each file is a linked list of disk blocks	All the pointers to scattered	All the files are blocked	each file must occupy a set of contiguous block on the disk
31	In linked allocation ------------------	Each file must occupy a set of contiguous block on the disk	Each file is a linked list of disk blocks	All the pointers to scattered	All the files are blocked	Each file is a linked list of disk blocks

32	In indexed allocation ------------	Each file must occupy a set of contiguous block on the disk	Each file is a linked list of disk blocks	All the pointers to scattered blocks are placed together in one location	All the files are blocked	All the pointers to scattered blocks are placed together in one location
33	One system where there are multiple operating system, the decision to load a particular one is done by ----------------	Boot loader	Boot strap	Process control block	File control block	Boot loader
34	The VFS refers to -----------	Virtual File System	Valid File System	Virtual Font System	Virtual Function System	Virtual File System
35	The disadvantage of a linear list of directory entries is the ---------------------	Size of the linear list in the memory	Linear search to find a file	It is not reliable	It is not valid	Linear search to find a file
36	One difficulty of contiguous allocation is ---------------	Finding space for a new file	Ineffecient	Costly	Time taking	Finding space for a new file
37	To solve the problem of external fragmentation ----------------- needs to be done periodically	Compaction	Check	Formatting	Replacing memory	Compaction
38	If too little space is allocated to a file ----------------	The file will not work	There will not be any space	The file cannot be extended	file cannot be opened	The file cannot be extended
39	A system program such as fsck ------------------ is a consistency checker	UNIX	Windows	Macintosh	Solaris	UNIX
40	Each set of operations for performing a specific task is a ----------------------	Program	Code	Transaction	Method	Transaction
41	Once the changes are written to the log, they are considered to be ---------------	Committed	Aborted	Completed	Finished	Committed
42	When an entire command transaction is completed,-------------------	It is stored in the memory	It is removed from the log file	It is redone	It is deleted from the memory	It is removed from the log file

43	A circular buffer is ----------------	Write to the end of its space	Overwrite older value as it goes	both A and B	overwrite new values	both A and B
44	In --------------- information is recorded magnetically on platters	Magnetic disk	Electrical disk	Assemblies	Cylinders	Magnetic disk
45	The head of the magnetic disk are attached to a --------------- that moves all the head as unit	Spindle	Disk arm	Track	Pointer	Disk arm
46	The set of tracks that are at one arm position make up a -----------	Magnetic disk	Electrical disk	Assemblies	Cylinders	Cylinders
47	The time taken to move a disk arm to the desired cylinder is called as---------------	Positioning time	Random access ti	Seek time	Rotational latency	Seek time
48	When a head damages the magnetic surface, it is known as ---------------------	Disk crash	Head crash	Magnetic damage	All of these	Head crash
49	A flopy disk is designed to rotate ------------- as compared to a hard disk drive	Faster	Slower	At the same speed	Normal speed	Slower
50	The host controller is -------------------	Controller built at the end of each disk	Controller at the computer end of the bus	Both a and b	Controller at the system side	Controller at the computer end of the bus
51	The process of dividing a disk into sectors that the disk controller can read and write, before a disk can store data is known as-----------------	Partitioning	Swap space creation	Low-level formatting	Physical formatting	Low-level formatting ,Physical formatting
52	the data structure for a sector typically contains -------------------	Header	Data area	Trailer	Main section	Header ,Data area ,Trailer
53	The header and trailer of a sector contains information used by the disk controller such as ______________.	Main section	Error corecting codes	Sector number	Disk identifier	Sector number

54	The two steps that the operating system takes to use a disk to hold its files are ---------------- and --------------	partitioning	Swap space creation	Catching	Logical formatting	partitioning
55	The -------------- program initializes all aspects of the system, from CPU registers to device controllers and the content of main memory, and then starts the operating system	Main	Boot loader	Boot strap	ROM	Boot strap
56	For most computers the boot strap is stored in--------------------	RAM	ROM	Cache	Tertiary storage	ROM
57	A disk that has a boot partition is called a ---------------	Start disk	Destroyed blocks	Boot disk	Format disk	System disk,boot disk
58	Defective sectors on disks are often known as -----------------	Good blocks	System disk	Bad blocks	Semi blocks	Bad blocks
59	Bad blocks are called as ___________	Good Sectors	Defective Sectors	boot disks	boot strap	Defective Sectors
60	ROM got __________ file	boot strap	Data area	head data	random data	boot strap

Karpagam Academy of Higher Education

Department of Mathematics
Academic Year : 2016-2019

Semester : III **Class : II B.Sc Maths**
Subject : Operating Systems: Linux **Subject Code: 16MMU404B**

S.NO	Question	Opt1	Opt2	Opt3	Opt4	Answer
			UNIT V			
1	_____________ developed the Linux Operating System.	Linus Torvalds	Lion Torvalds	Lamp Torvalds	Lin Torvals	Linus Torvalds
2	The first Linux kernel run only on _________ Microprocessors.	80586 Intel	80386 Intel	80186 AMD	8085 Intel	80386 Intel
3	Linux looks and feels much like ___________ Operating System.	Windows	Lindows	UNIX	Apple Mac	UNIX
4	Linux is an _____________ Operating System.	Close Source	Open Source	Program Source	None of the above	Open Source
5	The Linux programming interface adheres to the _________ semantics	SVR4 UNIX	SOR4 UNIX	SXR4 UNIX	SYR4 UNIX	SVR4 UNIX
6	Linux is designed to be compliant with the _________ documents	Purge	Prefix	Postfix	POSIX	POSIX
7	Linux supports _____________ Operating System.	Multitasking	Multi User	Multiuser and Multitasking	Single User	Multiuser and Multitasking
8	_________ reads input command and translates it to the operating system.	Vinix Shell	Unix Hell	Unix Shell	Cinix Shell	Unix Shell
9	Linux was originally programmed with a _________ compatible filesystem	Minimum	Maximum	Minix	DOS	Minix
10	VFS stands for __________ File System.	Vertical	Virtual	vector	VINIX	Virtual

11	The _________is a low-level systems software whose main role is to manage hardware resources	Windows GUI	Linux Kernel	Minix System	Vinix System	Linux Kernel
12	_______ is a boot loader for Linux.	LILO	LIXO	LIFO	LIMO	LILO
13	CLI stands for ____________ Interface.	Close Line	Command Line	Common Line	Cinix Line	Command Line
14	Linux Virtual File System is designed around _________ principles.	Object based	Object Oriented	Structured	none	Object oriented
15	A ____________ is an interpreter for the information from the terminal device.	line discipline	Lane discipline	Character discipline	normalization	line discipline
16	Linux's security model is closely related to _______ security mechanism.	Relational	Transactional	Object Oriented	Text	Text
17	PAM stands for ________ Authentication Module.	Pure	Purge	Pipline	Pluggable	Pluggable
18	The standard on-disk file system used by Linux is called _________.	ex2fs	expfs	ex67fp	extfp	ex2fs
19	Linux implements dynamic linking through a spacial ________ library .	Locker	linker	liquid	Lava	linker
20	The __________ service allocates memory on demand in Linux.	calloc	kmalloc	dalloc	qalloc	kmalloc
21	Linux memory management uses a _________ algorithm to manage physical memory.	Bubble sort	Sort-heap	buddy-heap	heap	buddy-heap
22	Linux kernel supports symmetric __________ hardware.	multi processing	single processing	Image processing	Data processing	multi processing
23	A ________ mechanism allows device drivers to reserve hardware resources.	module management	driver management	conflict-resolution	memory management	conflict-resolution
24	The _______ management allows modules to be loaded into memory.	module	memory	processor	file	module
25	The Linux kernel is distributed under General _________.	Public License.	Private Lincense	Corporate License	Domain License	Public License.

26	_____________ is a popular distribution of commercial Linux.	White Hat	Black Hat	Grey Hat	Red Hat	Red Hat
27	The ___________ is an array of pointers to kernel file structures.	memory table	file table	module table	schedule table	file table
28	________are a potential problem in kernel routine.	Page faults	Processor faults	Pivot faults	Pipeline faults	Page faults
29	Linux interrupts are a problem only if ____________ exist.	Cross Sections	Middle sections	Critical Sections	Dummy Table	Critical sections
30	Linux uses __________ algorithm for time-sharing process.	Sorting	Swapping	Searching	credit-based	credit-based
31	Windows 2000 is ________ bit operating system.	34	35	32	64	32
32	Windows 2000 is ________ operating system.	multi kernel	multi tasking	multi threading	multi OS	multi tasking
33	Windows 2000 provides better ___________ support.	OS	CUI	networking	platform	networking
34	Windows 2000 supports upto ________ of RAM.	4 gb	8 gb	16 gb	64 gb	64 gb
35	DLL stands ________ Link Library.	Databse	Distributed	Dynamic	DoS	Dynamic
36	HAL stands for Hardware _________ Layer.	Absolute	Abstraction	Ambient	Apple Mac	Abstraction
37	LPC stands for __________ Procedure Call.	Lower	Local	Less	Little	Local
38	Windows 2000 supports many languages because it has __________ API.	NLS	NPI	NIT	NQT	NLS
39	The kernel of Windows 2000 is ______________ oriented.	Procedure	Object	Function	Structure	Object
40	A ______ object acts as gate to control the number of threads.	semaphore	Super	Semiphore	Mutual Exclusion	semaphore

41	___________ objects control dispatching and synchronization.	Databse	Distributed	Dispatcher	Scheduler	Dispatcher
42	APC stands for ___________ Procedure Call.	Absolute	Abstraction	Asynchronous	Synchronous	Asynchronous
43	__________ can occur when a thread enters the ready or wait state.	Symmetric	Scheduling	Sampling	Superphore	Scheduling
44	The ____________ dispatcher creates an exception record.	Extreme	Exception	Error	extfp	Exception
45	The kernel uses an ________ dispatch table to bind each interrupt.	Interrupt	Interior	Inter	Inner	Interrupt
46	DPC stands for ___________ Procedure Call.	Database	Deferred	Dispatcher	Distributed	Deferred
47	The job of the ___________ manager is to supervise the use of all objects.	Object	memory	processor	Printer	Object
48	___________ are a standardized interface to all kinds of objects.	Hampers	Huckles	Handles	Heaps	Handles
49	The design of _________Memory manager supports a paging mechanism.	Virtue	Virtual	vector	Vento	Virtual
50	___________ is responsible for cache management and network drivers.	Network	Driver	I/O Manager	Interrupt	I/O Manager
51	VDM stands for Virtual _________ Machine.	Databse	DOS	Drive	Distributed	DOS
52	________ routines call the appropriate Win32 subroutines, converting 16-bit addresses into 32-bit ones	Skeleton	Kernel	Stub	GID function	Stub
53	________ FAT file system has solved the size and fragmentation problems	16-bit	8-bit	64-bit	32-bit	32-bit
54	NTFS uses _______ cluster numbers as disk addresses.	Logical	Physical	Virtual	Static	Logical

55	In MS-DOS and UNIX, each directory uses a datastructure called a ________ Tree	AVL	BinarySearch	P	B+	B+
56	NTFS divides the file's data into _______ units which are blocks of 16 contiguous clusters	Single	Compression	Two	Compaction	Compression
57	The ____ is a protocol provided by Windows 2000 to communicate between Windows 2000 server and other clients over the Internet.	TCP/IP	DLC	AppleTalk	PPTP	PPTP
58	The Common Object Model is a mechanism for interprocess communication that was developed for ________	IBM	UNIX	LINUX	Windows	Windows
59	________ protocol was introduced by IBM in 1985 as networking protocol for up to 254 machines	WebDAV	NetBIOS	Novell Netware	DHCP	NetBIOS
60	Winsock is a _________ layer interface that is largely compatible with UNIX sockets	Applications	Network	Session	Data-link	Session

Register Number____________
[16MMU404B]

KARPAGAM ACADEMY OF HIGHER EDUCATION
(Deemed to be university)
Coimbatore - 641021.
B.SC Degree Examinations
Fourth Semester
First Internal
Operating System: Linux

Date & Session:		Class	: II B.SC (Maths)
Duration: 2 Hours		Maximum	: 50 Marks

PART-A (20 X 1 = 20 Marks)
Answer ALL the Questions

1.____________is a program that manages the computer hardware and acts as an intermediary between the computer user and the computer hardware.
a.hardware acceleration **b.Operating System** c.compiler d.logical transcation
2._____manages the execution of user programs to prevent errors and improper use of the computer
a.resource allocator b.work station **c.main frame** d.control program
3.____were the first computers used to tackle many commercial & scientific application.
a.Mainframe computer system b.Mainframe computer service
c.multiframe computer system d.multiframe computer service
4._______________contains the address of an instruction to be fetched from memory
a.Program counter (PC) **b.Instruction register (IR)**
c.Control registers d.Status registers
5._______________ is also known as parallel systems or tightly coupled systems
a.Multiprocessor systems b.desktop systems c.Time sharing systems
d.Multiprogrammed systems
6.______________operating systems are even more complex than multi programmed operating systems.
a.Time-sharing b.desktop systems c.Multiprogrammed systems
d.Multiprocessor systems
7. __________ operating system keeps several jobs in memory simultaneously.
a.Time-sharing b.desktop systems c.Multiprogrammed systems
d.Multiprocessor systems
8.___________can save more money than multiple single-processor systems
a.Multiprocessor systems b.desktop systems c.Time sharing systems
d.Multiprogrammed systems
9.The most common multiple-processor systems now use
a.symmetric multiprocessing b.asymmetric multiprocessing
c.multithreading d.multiprogramming
10.Another form of a special-purpose operating system is the
a.real-time system b.distributed operating system c.Process states
d.multiframe computer system
11.The assignment of the CPU to the first process on the ready list is called
a.graceful degradation b.Time-sharing **c.dispatching** d.Multiprocessor systems
12.The manifestation of a process in an operating system is a
a.Process state transitions **b.process control block** c.child process
d.cooperating processes

13.For multiprogramming operating system
a.special support from processor is essential **b.special support from processor is not essential**
c.cache memory is essential d.cache memory is not essential
14.Which operating system reacts in the actual time
a.Batch system b.Quick response system **c.Real time system**
d.Time sharing system
15.The primary job of an OS is to ________
a.command resource b.manage resource c.provide utilities **d.Be user friendly**
16.The term " Operating System " means ________
a.A set of programs which controls computer working
b.The way a computer operator works
c.Conversion of high-level language in to machine level language
d.The way a floppy disk drive operates
17.With .... more than one process can be running simultaneously each on a different processer.
a.Multiprogramming b.Uniprocessing **c.Multiprocessing** d.Uniprogramming
18.The two central themes of modern operating system are
a.Multiprogramming and Distributed processing
b.Multiprogramming and Central Processing
c.Single Programming and Distributed processing
d.None of above
19..……………….. isa example of an operating system that support single user process and single thread
a.UNIX **b.MS-DOS** c.OS/2 d.Windows 2000
20.The operating system of a computer serves as a software interface between the user and the .
a.Hardware b.Peripheral c.Memory d.Screen

PART-B (3 X 2 = 6 Marks)

(Answer ALL the Questions)

21. Define: Operating Systems

- An operating system (**OS**) is the program that, after being initially loaded into the computer by a boot program, manages all the other programs in a computer. The other programs are called applications or application programs.

22. What is process?

- In computing, a **process** is an instance of a computer program that is being executed. It contains the program code and its current activity. Depending on the operating system (**OS**), a **process** may be made up of multiple threads of execution that execute instructions concurrently.

23. What is swapping?

- To replace pages or segments of data in memory. **Swapping** is a useful technique that enables a computer to execute programs and manipulate data files larger than main memory.

PART-C (3 X 8 = 24 Marks)

(Answer ALL the Questions)

24. A. Discuss the Desktop Operating System in detail (OR)

- The control program in a user's machine (desktop or laptop). Also called a "client operating system," Windows is the overwhelming majority while the Macintosh comes second. There are also several versions of Linux for the desktop.
- A technician might be asked to choose and install an OS for a customer. There are two distinct types of operating systems: desktop and network. A desktop operating system is intended for use in a small office, home office (SOHO) environment with a limited number of users. A network operating system (NOS) is designed for a corporate environment serving multiple users with a wide range of needs.
- A desktop OS has the following characteristics:
- Supports a single user
- Runs single-user applications
- Shares files and folders on a small network with limited security
- In the current software market, the most commonly used desktop operating systems fall into three groups: Microsoft Windows, Apple Mac OS, and Linux. This chapter focuses on Microsoft operating systems.

Microsoft Windows

Windows is one of the most popular operating systems today. The following versions of Windows are available:

- **Windows 7 Starter** - Used on netbook computers to make networking easy
- **Windows 7 Home Premium** - Used on home computers to easily share media
- **Windows 7 Professional** - Used on small business computers to secure critical information and to make routine tasks easier to complete
- **Windows 7 Enterprise** - Used on large business computers to provide more enhanced productivity, security, and management features
- **Windows 7 Ultimate** - Used on computers to combine the ease of use of Windows 7 Home Premium with the business capabilities of Windows 7 Professional and provide added data security
- **Windows Vista Home Basic** - Used on home computers for basic computing
- **Windows Vista Home Premium** - Used on home computers to expand personal productivity and digital entertainment beyond the basics
- **Windows Vista Business** - Used on small business computers for enhanced security and enhanced mobility technology
- **Windows Vista Enterprise** - Used on large business computers to provide more enhanced productivity, security, and management features

- **Windows Vista Ultimate** - Used on computers to combine all the needs of both home and business users

- **Windows XP Professional** - Used on most computers that connect to a Windows Server on a network

- **Windows XP Home** - Used on home computers and has limited security

- **Windows XP Media Center** - Used on entertainment computers for viewing movies and listening to music

- **Windows XP 64-bit Professional** - Used for computers with 64-bit processors
- **Apple Mac OS**
- Apple computers are proprietary and use an operating system called Mac OS. Mac OS is designed to be a user-friendly GUI operating system. Current versions of Mac OS are based on a customized version of UNIX.
- **Linux**
  Linux is based on UNIX, which was introduced in the late 1960s and is one of the oldest operating systems. Linus Torvalds designed Linux in 1991 as an open-source OS. Open-source programs allow the source code to be distributed and changed by anyone as a free download or by developers at a much lower cost than other operating systems.

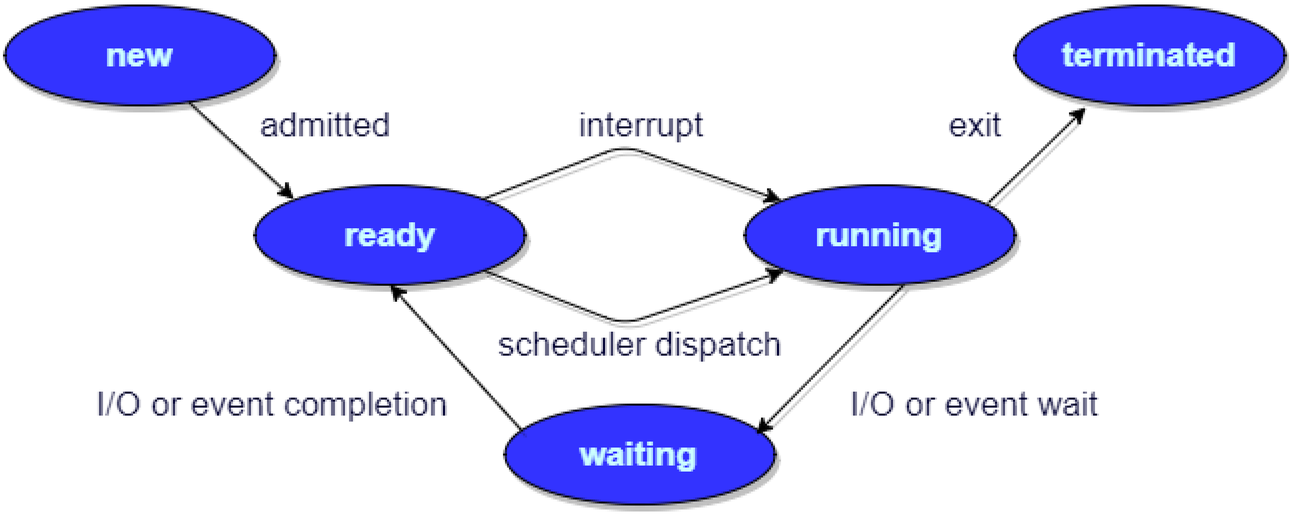
B. Write short notes on process states.

- **What is a Process?**
- A process is a program in execution. Process is not as same as program code but a lot more than it. A process is an 'active' entity as opposed to program which is considered to be a 'passive' entity. Attributes held by process include hardware state, memory, CPU etc.
- Process memory is divided into four sections for efficient working :
- The text section is made up of the compiled program code, read in from non-volatile storage when the program is launched.
- The data section is made up the global and static variables, allocated and initialized prior to executing the main.
- The heap is used for the dynamic memory allocation, and is managed via calls to new, delete, malloc, free, etc.
- The stack is used for local variables. Space on the stack is reserved for local variables when they are declared.

PROCESS STATE

Processes can be any of the following states :

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



25. A. How do you avoid deadlock explain? (OR)

- **Deadlock Avoidance**
- Deadlock avoidance can be done with Banker's Algorithm.
- **Banker's Algorithm**
- Bankers's Algorithm is resource allocation and deadlock avoidance algorithm which test all the request made by processes for resources, it check for safe state, if after granting request system remains in the safe state it allows the request and if their is no safe state it don't allow the request made by the process.
- Inputs to Banker's Algorithm
  1. Max need of resources by each process.
  2. Currently allocated resources by each process.
  3. Max free available resources in the system.
- Request will only be granted under below condition.
  1. If request made by process is less than equal to max need to that process.
- 2. If request made by process is less than equal to freely availbale resource in the system.

B. Explain the concept of Inter Process Communication.

- A process can be of two type:
- Independent process.
- Co-operating process.
- An independent process is not affected by the execution of other processes while a co-operating process can be affected by other executing processes. Though one can think that those processes, which are running independently, will execute very efficiently but in practical, there are many situations when co-operative nature can be utilised for increasing computational speed, convenience and modularity. Inter process communication (IPC) is a mechanism which allows processes to communicate each other and synchronize their actions. The communication between these processes can be seen as a method of co-operation between them. Processes can communicate with each other using these two ways:
- Shared Memory
- Message passing
- The Figure 1 below shows a basic structure of communication between processes via shared memory method and via message passing.

- An operating system can implement both method of communication. First, we will discuss the shared memory method of communication and then message passing. Communication between processes using shared memory requires processes to share some variable and it completely depends on how programmer will implement it. One way of communication using shared memory can be imagined like this: Suppose process1 and process2 are executing simultaneously and they share some resources or use some information from other process, process1 generate information about certain computations or resources being used and keeps it as a record in shared memory. When process2 need to use the shared information, it will check in the record stored in shared memory and take note of the information generated by process1 and act accordingly. Processes can use shared memory for extracting information as a record from other process as well as for delivering any specific information to other process. Let's discuss an example of communication between processes using shared memory method.

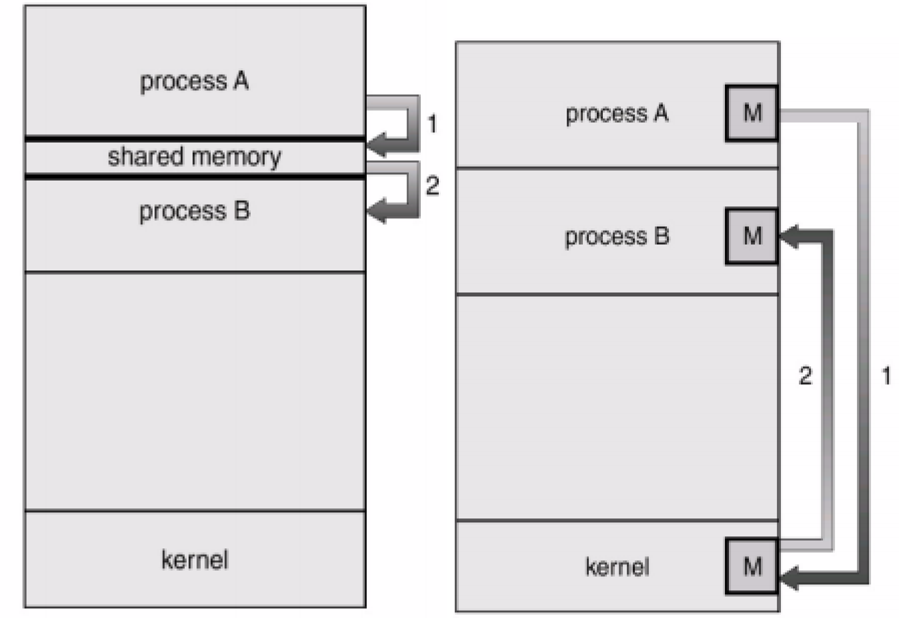


Figure 1: Shared Memory and Message Passing (Image Taken from "Operating System Cncepts" by Galvin et al.)

26. A.Write short notes on memory management. (OR)

- Memory management is the functionality of an operating system which handles or manages primary memory and moves processes back and forth between main memory and disk during execution. Memory management keeps track of each and every memory location, regardless of either it is allocated to some process or it is free. It checks how much memory is to be allocated to processes. It decides which process will get memory at what time. It tracks whenever some memory gets freed or unallocated and correspondingly it updates the status.
- This tutorial will teach you basic concepts related to Memory Management.

Process Address Space

- The process address space is the set of logical addresses that a process references in its code. For example, when 32-bit addressing is in use, addresses can range from 0 to 0x7fffffff; that is, 2^31 possible numbers, for a total theoretical size of 2 gigabytes.

- The operating system takes care of mapping the logical addresses to physical addresses at the time of memory allocation to the program. There are three types of addresses used in a program before and after memory is allocated −

S.N.	Memory Addresses & Description
1	**Symbolic addresses** The addresses used in a source code. The variable names, constants, and instruction labels are the basic elements of the symbolic address space.
2	**Relative addresses** At the time of compilation, a compiler converts symbolic addresses into relative addresses.
3	**Physical addresses** The loader generates these addresses at the time when a program is loaded into main memory.

Virtual and physical addresses are the same in compile-time and load-time address-binding schemes. Virtual and physical addresses differ in execution-time address-binding scheme.

The set of all logical addresses generated by a program is referred to as a **logical address space**. The set of all physical addresses corresponding to these logical addresses is referred to as a **physical address space.**

The runtime mapping from virtual to physical address is done by the memory management unit (MMU) which is a hardware device. MMU uses following mechanism to convert virtual address to physical address.

- The value in the base register is added to every address generated by a user process, which is treated as offset at the time it is sent to memory. For example, if the base register value is 10000, then an attempt by the user to use address location 100 will be dynamically reallocated to location 10100.
- The user program deals with virtual addresses; it never sees the real physical addresses.

B. What is paging? What is the usage of paging explain.

- Paging is a memory management scheme that eliminates the need for contiguous allocation of physical memory. This scheme permits the physical address space of a process to be non – contiguous.
- Logical Address or Virtual Address (represented in bits): An address generated by the CPU
- Logical Address Space or Virtual Address Space(represented in words or bytes): The set of all logical addresses generated by a program
- Physical Address (represented in bits): An address actually available on memory unit
- Physical Address Space (represented in words or bytes): The set of all physical addresses corresponding to the logical addresses
- **Example:**
- If Logical Address = 31 bit, then Logical Address Space = 2^{31} words = 2 G words (1 G = 2^{30})
- If Logical Address Space = 128 M words = $2^7 * 2^{20}$ words, then Logical Address = $\log_2 2^{27}$ = 27 bits
- If Physical Address = 22 bit, then Physical Address Space = 2^{22} words = 4 M words (1 M = 2^{20})

- If Physical Address Space = 16 M words = $2^4 * 2^{20}$ words, then Physical Address = $\log_2 2^{24}$ = 24 bits
- The mapping from virtual to physical address is done by the memory management unit (MMU) which is a hardware device and this mapping is known as paging technique.
- The Physical Address Space is conceptually divided into a number of fixed-size blocks, called **frames**.
- The Logical address Space is also splitted into fixed-size blocks, called **pages**.
- Page Size = Frame Size
- Let us consider an example:
- Physical Address = 12 bits, then Physical Address Space = 4 K words
- Logical Address = 13 bits, then Logical Address Space = 8 K words
- Page size = frame size = 1 K words (assumption)

Address generated by CPU is divided into

- **Page number(p):** Number of bits required to represent the pages in Logical Address Space or Page number
- **Page offset(d):** Number of bits required to represent particular word in a page or page size of Logical Address Space or word number of a page or page offset.

Physical Address is divided into

- **Frame number(f):** Number of bits required to represent the frame of Physical Address Space or Frame number.
- **Frame offset(d):** Number of bits required to represent particular word in a frame or frame size of Physical Address Space or word number of a frame or frame offset.

****

Introduction -Mainframe systems Desktop Systems – Multiprocessor systems – distributed systems – real time systems. Process: - Process concepts – Operation on process – cooperation process - Inter process Communication - Mutual Exclusion - Critical sections- primitives – Semaphores – Deadlock: System Model, Deadlock characterization, Deadlock prevention, avoidance, detection, recovery from deadlock.

OPERATING SYSTEMS

INTRODUCTION

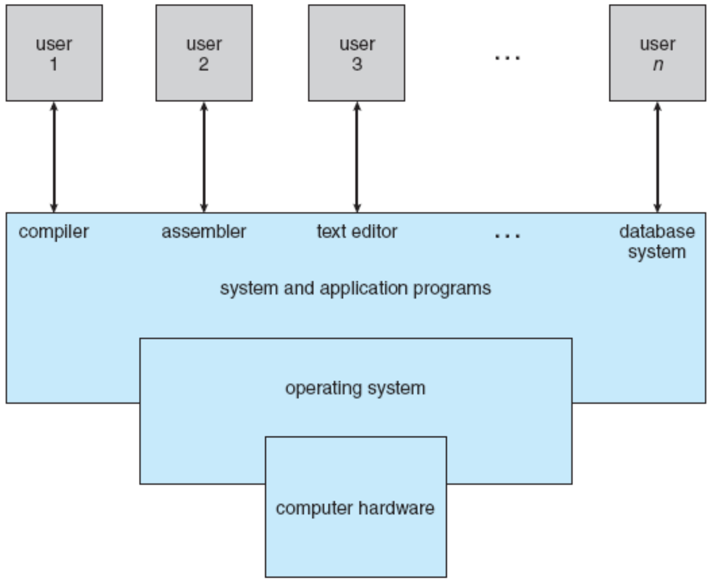
An Operating System (OS) is an interface between computer user and computer hardware. An operating system is software which performs all the basic tasks like file management, memory management, process management, handling input and output, and controlling peripheral devices such as disk drives and printers. Some popular Operating Systems include Linux, Windows, OS X, VMS, OS/400, AIX, z/OS, etc.

- **Definition**

An operating system is a program that acts as an interface between the user and the computer hardware and controls the execution of all kinds of programs.

What is an OS?

A computer system can be divided roughly into four components: the hardware, the operating system, the application programs, and the users. The hardware provides the basic computing resources for the system. The application programs define the ways in which these resources are used to solve users' computing problems.



System Components

The operating system controls the hardware and coordinates its use among the various application programs for the various users. OS cannot be defined exactly because, it differs in perspective.

- **User View**

The user's view of the computer varies according to the interface being used. In a personal Computing environment the goal of OS is "ease to use" with some attention paid for "resource-sharing ". In Computing environment like mainframes and minicomputer "Resource utilization" is maximized for computer availability and prevent user from sharing other's fair time. In environment like client server "Individual Usability" and "Resource sharing" are compromised in designing. In latest technologies like mobile and touch-pads, lap-tops the work of OS is to improve "battery-life" for better efficiency. In some systems like embedded system user's interaction is needed at initial phases only. The design principles of user view differ, so defining the work of OS cannot be made on their perspective.

- **System View**

In system (Computer) point of view, the work of OS is involved with the efficiency of handling hardware or software resources. In context, an OS can be viewed as a "Resource allocator". A computer system has many resources that may be required to solve a problem: CPU time, memory space, file-storage space, I/O devices, and so on. The operating system acts as the manager of these resources. Facing numerous and possibly conflicting requests for resources, the operating system must decide how to allocate them to specific programs and users so that it can operate the computer system efficiently and fairly.

An operating system can be viewed as a "Control Program" that manages the execution of user programs to prevent errors and improper use of the computer.

BASIC OS FUNCTION

Following are some of important functions of an operating System.

- Memory Management
- Processor Management
- Device Management
- File Management
- Security
- Control over system performance
- Job accounting
- Error detecting aids
- Coordination between other software and users

Memory Management

Memory management refers to management of Primary Memory or Main Memory. Main memory is a large array of words or bytes where each word or byte has its own address.

Main memory provides a fast storage that can be accessed directly by the CPU. For a program to be executed, it must in the main memory. An Operating System does the following activities for memory management −

- Keeps tracks of primary memory, i.e., what part of it are in use by whom, what part are not in use.
- In multiprogramming, the OS decides which process will get memory when and how much.
- Allocates the memory when a process requests it to do so.
- De-allocates the memory when a process no longer needs it or has been terminated.

Processor Management

In multiprogramming environment, the OS decides which process gets the processor when and for how much time. This function is called process scheduling. An Operating System does the following activities for processor management −

- Keeps tracks of processor and status of process. The program responsible for this task is known as traffic controller.
- Allocates the processor (CPU) to a process.
- De-allocates processor when a process is no longer required.

Device Management

An Operating System manages device communication via their respective drivers. It does the following activities for device management −

- Keeps tracks of all devices. Program responsible for this task is known as the I/O controller.
- Decides which process gets the device when and for how much time.
- Allocates the device in the efficient way.
- De-allocates devices.

File Management

A file system is normally organized into directories for easy navigation and usage. These directories may contain files and other directions.

An Operating System does the following activities for file management −

- Keeps track of information, location, uses, status etc. The collective facilities are often known as file system.
- Decides who gets the resources.
- Allocates the resources.
- De-allocates the resources.

Other Important Activities

Following are some of the important activities that an Operating System performs −

- Security − By means of password and similar other techniques, it prevents unauthorized access to programs and data.
- Control over system performance − Recording delays between request for a service and response from the system.
- Job accounting − Keeping track of time and resources used by various jobs and users.
- Error detecting aids − Production of dumps, traces, error messages, and other debugging and error detecting aids.
- Coordination between other softwares and users − Coordination and assignment of compilers, interpreters, assemblers and other software to the various users of the computer systems.

The operating system is the core software component of your computer. It performs many functions and is, in very basic terms, an interface between your computer and the outside world. In the section about hardware, a computer is described as consisting of several component parts including your monitor, keyboard, mouse, and other parts. The operating system provides an interface to these parts using what is referred to as "drivers". This is why sometimes when you install a new printer or other piece of hardware, your system will ask you to install more software called a driver.

An operating system has three main functions: (1) manage the computer's resources, such as the central processing unit, memory, disk drives, and printers, (2) establish a user interface, and (3) execute and provide services for applications software.

Other Operating System Functions

The operating system provides for several other functions including:

- System tools (programs) used to monitor computer performance, debug problems, or maintain parts of the system.
- A set of libraries or functions which programs may use to perform specific tasks especially relating to interfacing with computer system components.
- The operating system makes these interfacing functions along with its other functions operate smoothly and these functions are mostly transparent to the user.
- The operating system underpins the entire operation of the modern computer.

RESOURCE ABSTRACTION

- Resource abstraction is the process of "hiding the details of how the hardware operates, thereby making computer hardware relatively easy for an application programmer to use"

OPERATING SYSTEM TYPES

- There are many types of operating systems. The most common is the Microsoft suite of operating systems. They include from most recent to the oldest:
- Windows XP Professional Edition - A version used by many businesses on workstations. It has the ability to become a member of a corporate domain.
- Windows XP Home Edition - A lower cost version of Windows XP which is for home use only and should not be used at a business.

- Windows 2000 - A better version of the Windows NT operating system which works well both at home and as a workstation at a business. It includes technologies which allow hardware to be automatically detected and other enhancements over Windows NT.
- Windows ME - A upgraded version from windows 98 but it has been historically plagued with programming errors which may be frustrating for home users.
- Windows 98 - This was produced in two main versions. The first Windows 98 version was plagued with programming errors but the Windows 98 Second Edition which came out later was much better with many errors resolved.
- Windows NT - A version of Windows made specifically for businesses offering better control over workstation capabilities to help network administrators.
- Windows 95 - The first version of Windows after the older Windows 3.x versions offering a better interface and better library functions for programs.

There are other worthwhile types of operating systems not made by Microsoft. The greatest problem with these operating systems lies in the fact that not as many application programs are written for them. However if you can get the type of application programs you are looking for, one of the systems listed below may be a good choice.

- Unix - A system that has been around for many years and it is very stable. It is primary used to be a server rather than a workstation and should not be used by anyone who does not understand the system. It can be difficult to learn. Unix must normally run an a computer made by the same company that produces the software.
- Linux - Linux is similar to Unix in operation but it is free. It also should not be used by anyone who does not understand the system and can be difficult to learn.
- Apple MacIntosh - Most recent versions are based on Unix but it has a good graphical interface so it is both stable (does not crash often or have as many software problems as other systems may have) and easy to learn. One drawback to this system is that it can only be run on Apple produced hardware.

TYPES OF OPERATING SYSTEM

- Types of operating system which are commonly used

MULTI-PROGRAMMING SYSTEM

- The work of the server is to execute the job in sequence assigned by the users at their fair intervals. This is the first time the OS are programmed (Control Program or Handler) to handle the users with the required resources. The switching between the users and the allocation of same resources to multiple processes was the difficult task. There was plenty of algorithm design for this by various research sectors in this time which paved a new way for multi-processing.

- *Multiprogramming* is a rudimentary form of parallel processing in which several programs are run at the same time on a uniprocessor. Since there is only one processor, there can be no true simultaneous execution of different programs.

BATCH OPERATING SYSTEM

- The tasks are grouped as batch based on the priority specified by the user. Once the tasks are grouped they are executed as a batch by the machine. The duration of execution may be a week or even months. The tasks are grouped manually by a person and after proper execution the results are given to them by that person. The processing of OS is to just execute the task and not on scheduling.
- The users of a batch operating system do not interact with the computer directly. Each user prepares his job on an off-line device like punch cards and submits it to the computer operator. To speed up processing, jobs with similar needs are batched together and run as a group. The programmers leave their programs with the operator and the operator then sorts the programs with similar requirements into batches.

The problems with Batch Systems are as follows −

- Lack of interaction between the user and the job.
- CPU is often idle, because the speed of the mechanical I/O devices is slower than the CPU.
- Difficult to provide the desired priority.

TIME-SHARING OPERATING SYSTEMS

- Time-sharing is a technique which enables many people, located at various terminals, to use a particular computer system at the same time. Time-sharing or multitasking is a logical extension of multiprogramming. Processor's time which is shared among multiple users simultaneously is termed as time-sharing.
- The main difference between Multiprogrammed Batch Systems and Time-Sharing Systems is that in case of Multiprogrammed batch systems, the objective is to maximize

processor use, whereas in Time-Sharing Systems, the objective is to minimize response time.

- Multiple jobs are executed by the CPU by switching between them, but the switches occur so frequently. Thus, the user can receive an immediate response. For example, in a transaction processing, the processor executes each user program in a short burst or quantum of computation.
- That is, if n users are present, then each user can get a time quantum. When the user submits the command, the response time is in few seconds at most. The operating system uses CPU scheduling and multiprogramming to provide each user with a small portion of a time. Computer systems that were designed primarily as batch systems have been modified to time-sharing systems.

Advantages of Timesharing operating systems are as follows −

- Provides the advantage of quick response.
- Avoids duplication of software.
- Reduces CPU idle time.

Disadvantages of Time-sharing operating systems are as follows −

- Problem of reliability.
- Question of security and integrity of user programs and data.
- Problem of data communication.

REAL TIME OPERATING SYSTEM

- A real-time system is defined as a data processing system in which the time interval required to process and respond to inputs is so small that it controls the environment. The time taken by the system to respond to an input and display of required updated information is termed as the response time. So in this method, the response time is very less as compared to online processing.
- Real-time systems are used when there are rigid time requirements on the operation of a processor or the flow of data and real-time systems can be used as a control device in a

dedicated application. A real-time operating system must have well-defined, fixed time constraints, otherwise the system will fail. For example, Scientific experiments, medical imaging systems, industrial control systems, weapon systems, robots, air traffic control systems, etc.

There are two types of real-time operating systems.

Hard real-time systems

- Hard real-time systems guarantee that critical tasks complete on time. In hard real-time systems, secondary storage is limited or missing and the data is stored in ROM. In these systems, virtual memory is almost never found.

Soft real-time systems

- Soft real-time systems are less restrictive. A critical real-time task gets priority over other tasks and retains the priority until it completes. Soft real-time systems have limited utility than hard real-time systems. For example, multimedia, virtual reality, Advanced Scientific Projects likes undersea exploration and planetary rovers, etc.

DISTRIBUTED OPERATING SYSTEM

- Distributed systems use multiple central processors to serve multiple real-time applications and multiple users. Data processing jobs are distributed among the processors accordingly.
- The processors communicate with one another through various communication lines (such as high-speed buses or telephone lines). These are referred as loosely coupled systems or distributed systems. Processors in a distributed system may vary in size and function. These processors are referred as sites, nodes, computers, and so on.

The advantages of distributed systems are as follows −

- With resource sharing facility, a user at one site may be able to use the resources available at another.
- Speedup the exchange of data with one another via electronic mail.
- If one site fails in a distributed system, the remaining sites can potentially continue operating.
- Better service to the customers.
- Reduction of the load on the host computer.
- Reduction of delays in data processing.

Distributed Systems

- A distributed system is a collection of physically separate, possibly heterogeneous, computer systems that are networked to provide users with access to the various resources that the system maintains. Access to a shared resource increases computation speed, functionality, data availability, and reliability. Some operating systems generalize network access as a form of file access, with the details of networking contained in the network interface's device driver. Distributed systems depend on networking for their functionality.

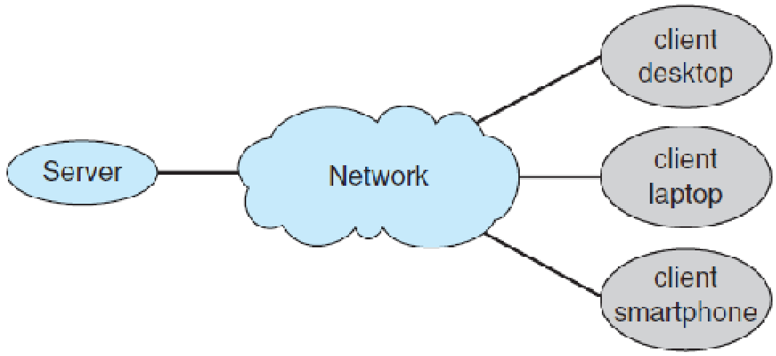
- Networks vary by the protocols used, the distances between nodes, and the transport media. TCP/IP is the most common network protocol, and it provides the fundamental architecture of the Internet. Most operating systems support TCP/IP, including all general-purpose ones. The media to carry networks are equally varied. They include copper wires, fiber strands, and wireless transmissions between satellites, microwave dishes, and radios.
- A network operating system is an operating system that provides features such as file sharing across the network, along with a communication scheme that allows different processes on different computers to exchange messages. A computer running a network operating system acts autonomously from all other computers on the network, although it is aware of the network and is able to communicate with other networked computers.

Client-Server Computing: As PCs have become faster, more powerful, and cheaper, designers have shifted away from centralized system architecture. Terminals connected to centralized systems are now being supplanted by PCs and mobile devices. Correspondingly, user-interface functionality once handled directly by centralized systems is increasingly being handled by PCs, quite often through a web interface. As a result, many of today's systems act as server systems to satisfy requests generated by client systems. This form of specialized distributed system, called a client–server system

Server systems can be broadly categorized as compute servers and file servers:

• The compute-server system provides an interface to which a client can send a request to perform an action (for example, read data). In response, the server executes the action and sends the results to the client. A server running a database that responds to client requests for data is an example of such a system.

• The file-server system provides a file-system interface where clients can create, update, read, and delete files. An example of such a system is a web server that delivers files to clients running web browsers.

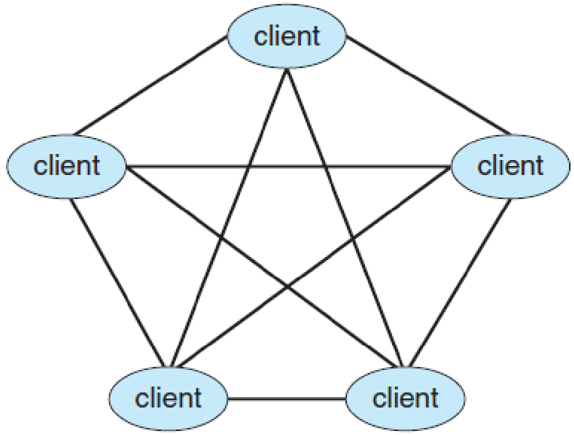


Client-Server Model

Peer to peer Systems

Another structure for a distributed system is the peer-to-peer (P2P) system model. In this model, clients and servers are not distinguished from one another. Instead, all nodes within the system are considered peers, and each may act as either a client or a server, depending on whether it is requesting or providing a service. Peer-to-peer systems offer an advantage over traditional client-server systems. In a client-server system, the server is a bottleneck; but in a peer-to-peer system, services can be provided by several nodes distributed throughout the network. Determining what services are available is accomplished in one of two general ways:

- When a node joins a network, it registers its service with a centralized lookup service on the network. Any node desiring a specific service first contacts this centralized lookup service to determine which node provides the service. The remainder of the communication takes place between the client and the service provider.
- An alternative scheme uses no centralized lookup service. Instead, a peer acting as a client must discover what node provides a desired service by broadcasting a request for the service to all other nodes in the network. The node (or nodes) providing that service responds to the peer making the request. To support this approach, a discovery protocol must be provided that allows peers to discover services provided by other peers in the network.



Peer-Peer with no-centralized Machine

Skype is another example of peer-to-peer computing. It allows clients to make voice calls and video calls and to send text messages over the Internet using a technology known as voice over IP (VoIP). Skype uses a hybrid peer- to-peer approach. It includes a centralized login server, but it also incorporates decentralized peers and allows two peers to communicate.

Network operating System

- A Network Operating System runs on a server and provides the server the capability to manage data, users, groups, security, applications, and other networking functions. The primary purpose of the network operating system is to allow shared file and printer access among multiple computers in a network, typically a local area network (LAN), a private network or to other networks.
- Examples of network operating systems include Microsoft Windows Server 2003, Microsoft Windows Server 2008, UNIX, Linux, Mac OS X, Novell NetWare, and BSD.

The advantages of network operating systems are as follows −

- Centralized servers are highly stable.
- Security is server managed.
- Upgrades to new technologies and hardware can be easily integrated into the system.
- Remote access to servers is possible from different locations and types of systems.

The disadvantages of network operating systems are as follows −

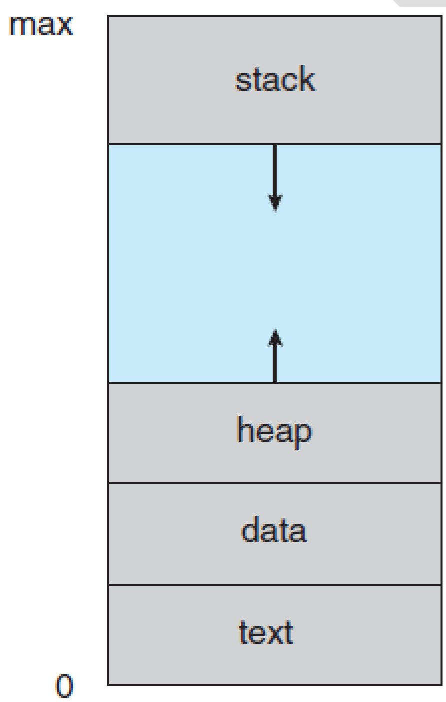
- High cost of buying and running a server.
- Dependency on a central location for most operations.
- Regular maintenance and updates are required.

Process Concept

Process is a program that is in execution. It is defined as unit of work in modern systems. A batch system executes jobs, whereas a time-shared system has user programs, or tasks. Even on a single-user system, a user may be able to run several programs at one time: a word processor, a Web browser, and an e-mail package. And even if a user can execute only one program at a time, such as on an embedded device that does not support multitasking, the operating system may need to support its own internal programmed activities, such as memory management. In many respects, all these activities are similar, so we call all of them processes.

Process in memory

A process is more than the program code, which is sometimes known as the text section.
It also includes the current activity, as represented by the value of the program counter and the contents of the processor's registers. A process generally also includes the process stack, which contains temporary data (such as function parameters, return addresses, and local variables), and a data section, which contains global variables. A process may also include a heap, which is memory that is dynamically allocated during process run time.



Process in Memory

A program is a passive entity, such as a file containing a list of instructions stored on disk (often called an executable file). In contrast, a process is an active entity, with a program counter specifying the next instruction to execute and a set of associated resources. A program becomes a process when an executable file is loaded into memory.

Process State

As a process executes, it changes state. The state of a process is defined in part by the current activity of that process. A process may be in one of the following states:

• New. The process is being created.

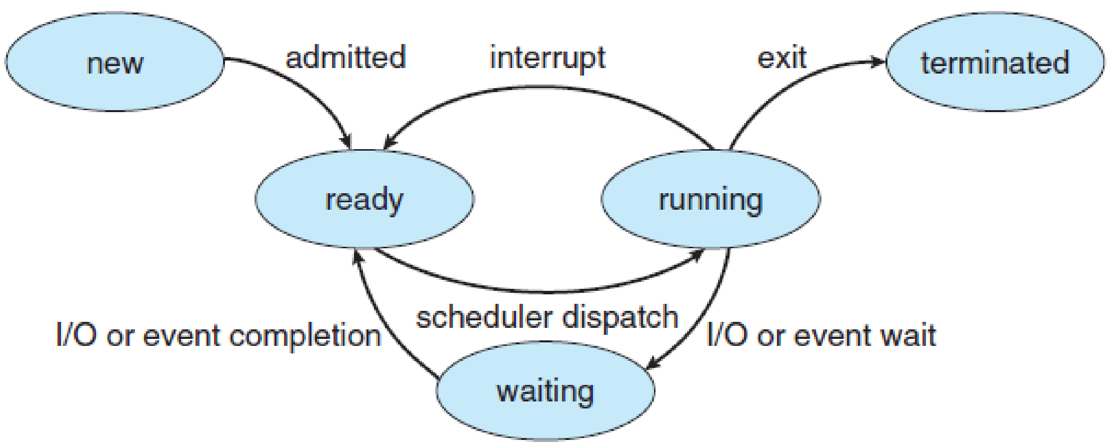
• Running. Instructions are being executed.

• Waiting. The process is waiting for some event to occur (such as an I/O completion or reception of a signal).

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

• Terminated. The process has finished execution.

These names are arbitrary, and they vary across operating systems. The states that they represent are found on all systems, however. Certain operating systems also more finely delineate process states. It is important to realize that only one process can be running on any processor at any instant. Many processes may be ready and waiting, however. The state diagram corresponding to these states is presented in the following Figure.



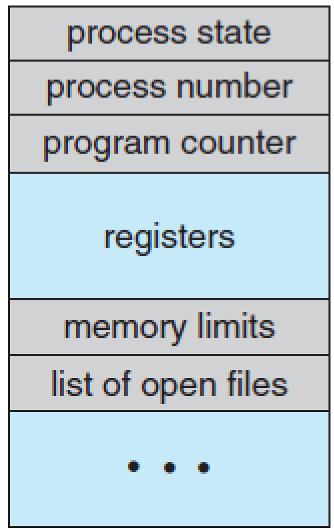
Process State Diagram

Process Control Block (PCB)

Each process is represented in the operating system by a process control block (PCB)—also called a task control block. It contains many pieces of information associated with a specific process, including these: Process state. The state may be new, ready, running, and waiting, halted, and so on.

- **Program counter.** The counter indicates the address of the next instruction to be executed for this process.
- **CPU registers.** The registers vary in number and type, depending on the computer architecture. They include accumulators, index registers, stack pointers, and general-purpose registers, plus any condition-code information. Along with the program counter, this state information must be saved when an interrupt occurs, to allow the process to be continued correctly afterward.

- **CPU-scheduling information.** This information includes a process priority, pointers to scheduling queues, and any other scheduling parameters.

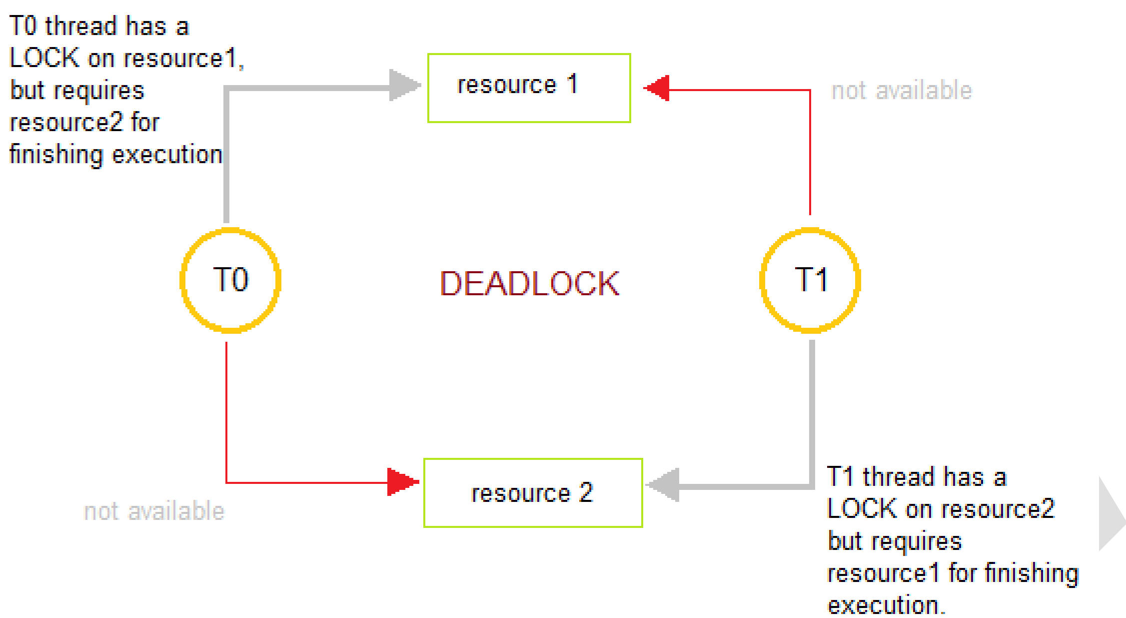


Process Control Block (PCB)

- **Memory-management information**. This information may include such items as the value of the base and limit registers and the page tables, or the segment tables, depending on the memory system used by the operating system
- **Accounting information**. This information includes the amount of CPU and real time used, time limits, account numbers, job or process numbers, and so on.
- **I/O status information**. This information includes the list of I/O devices allocated to the process, a list of open files, and so on.

What is a Deadlock?

Deadlocks are a set of blocked processes each holding a resource and waiting to acquire a resource held by another process.



How to avoid Deadlocks

Deadlocks can be avoided by avoiding at least one of the four conditions, because all this four conditions are required simultaneously to cause deadlock.

1. **Mutual Exclusion**

   Resources shared such as read-only files do not lead to deadlocks but resources, such as printers and tape drives, requires exclusive access by a single process.

2. **Hold and Wait**

   In this condition processes must be prevented from holding one or more resources while simultaneously waiting for one or more others.

3. **No Preemption**

   Preemption of process resource allocations can avoid the condition of deadlocks, where ever possible.

4. **Circular Wait**

   Circular wait can be avoided if we number all resources, and require that processes request resources only in strictly increasing(or decreasing) order.

Handling Deadlock

The above points focus on preventing deadlocks. But what to do once a deadlock has occured. Following three strategies can be used to remove deadlock after its occurrence.

1. **Preemption**

   We can take a resource from one process and give it to other. This will resolve the deadlock situation, but sometimes it does causes problems.

2. **Rollback**

   In situations where deadlock is a real possibility, the system can periodically make a record of the state of each process and when deadlock occurs, roll everything back to the last checkpoint, and restart, but allocating resources differently so that deadlock does not occur.

3. **Kill one or more processes**

   This is the simplest way, but it works.

Deadlock Prevention

Deadlock prevention algorithms ensure that at least one of the necessary conditions (Mutual exclusion, hold and wait, no preemption and circular wait) does not hold true. However most prevention algorithms have poor resource utilization, and hence result in reduced throughputs.

Mutual Exclusion

Not always possible to prevent deadlock by preventing mutual exclusion (making all resources shareable) as certain resources are cannot be shared safely.

Hold and Wait

We will see two approaches, but both have their disadvantages.

A resource can get all required resources before it start execution. This will avoid deadlock, but will result in reduced throughputs as resources are held by processes even when they are not needed. They could have been used by other processes during this time.

Second approach is to request for a resource only when it is not holing any other resource. This may result in a starvation as all required resources might not be available freely always.

No preemption

We will see two approaches here. If a process request for a resource which is held by another waiting resource, then the resource may be preempted from the other waiting resource. In the second approach, if a process request for a resource which are not readily available, all other resources that it holds are preempted.

The challenge here is that the resources can be preempted only if we can save the current state can be saved and processes could be restarted later from the saved state.

Circular wait

To avoid circular wait, resources may be ordered and we can ensure that each process can request resources only in an increasing order of these numbers. The algorithm may itself increase complexity and may also lead to poor resource utilization.

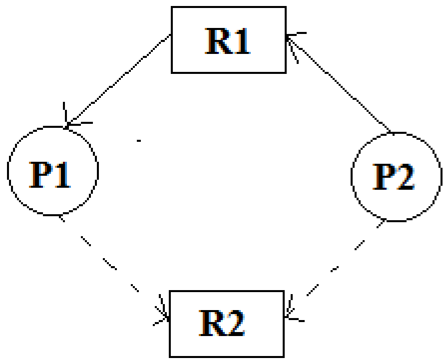
Deadlock avoidance

As you saw already, most prevention algorithms have poor resource utilization, and hence result in reduced throughputs. Instead, we can try to avoid deadlocks by making use prior knowledge about the usage of resources by processes including resources available, resources allocated, future requests and future releases by processes. Most deadlock avoidance algorithms need every process to tell in advance the maximum number of resources of each type that it may need. Based on all these info we may decide if a process should wait for a resource or not, and thus avoid chances for circular wait.

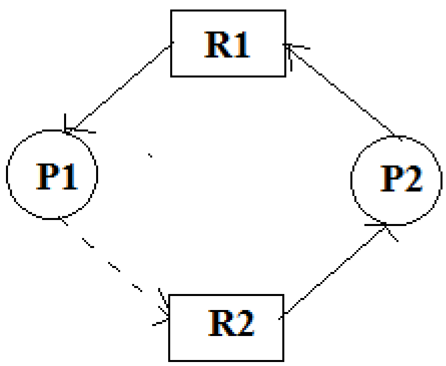
If a system is already in a safe state, we can try to stay away from an unsafe state and avoid deadlock. Deadlocks cannot be avoided in an unsafe state. A system can be considered to be in safe state if it is not in a state of deadlock and can allocate resources upto the maximum available. A safe sequence of processes and allocation of resources ensures a safe state. Deadlock avoidance algorithms try not to allocate resources to a process if it will make the system in an unsafe state. Since resource allocation is not done right away in some cases, deadlock avoidance algorithms also suffer from low resource utilization problem.

A resource allocation graph is generally used to avoid deadlocks. If there are no cycles in the resource allocation graph, then there are no deadlocks. If there are cycles, there may be a deadlock. If there is only one instance of every resource, then a cycle implies a deadlock. Vertices of the resource allocation graph are resources and processes. The resource allocation graph has request edges and assignment edges. An edge from a process to resource is a request edge and an edge from a resource to process is an allocation edge. A calm edge denotes that a request may be made in future and is represented as a dashed line. Based on calm edges we can see if there is a chance for a cycle and then grant requests if the system will again be in a safe state.

Consider the image with calm edges as below:



If R2 is allocated to p2 and if P1 request for R2, there will be a deadlock.

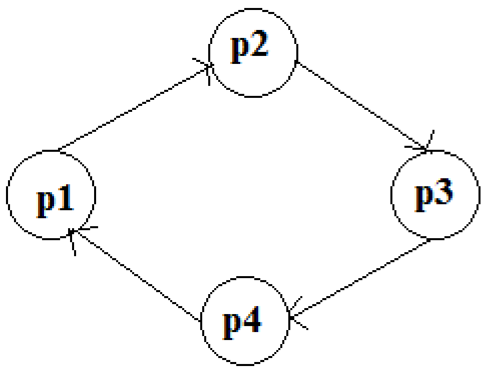


The resource allocation graph is not much useful if there are multiple instances for a resource. In such a case, we can use Banker's algorithm. In this algorithm, every process must tell upfront the maximum resource of each type it need, subject to the maximum available instances for each type. Allocation of resources is made only, if the allocation ensures a safe state; else the processes need to wait. The Banker's algorithm can be divided into two parts: Safety algorithm if a system is in a safe state or not. The resource request algorithm make an assumption of allocation and see if the system will be in a safe state. If the new state is unsafe, the resources are not allocated and the data structures are restored to their previous state; in this case the processes must wait for the resource. *You can refer to any operating system text books for details of these algorithms.*

Deadlock Detection

If deadlock prevention and avoidance are not done properly, as deadlock may occur and only things left to do is to detect the recover from the deadlock.

If all resource types has only single instance, then we can use a graph called wait-for-graph, which is a variant of resource allocation graph. Here, vertices represent processes and a directed edge from P1 to P2 indicate that P1 is waiting for a resource held by P2. Like in the case of resource allocation graph, a cycle in a wait-for-graph indicate a deadlock. So the system can maintain a wait-for-graph and check for cycles periodically to detect any deadlocks.



The wait-for-graph is not much useful if there are multiple instances for a resource, as a cycle may not imply a deadlock. In such a case, we can use an algorithm similar to Banker's algorithm to detect deadlock. We can see if further allocations can be made on not based on current allocations. *You can refer to any operating system text books for details of these algorithms.*

Deadlock Recovery

Once a deadlock is detected, you will have to break the deadlock. It can be done through different ways, including, aborting one or more processes to break the circular wait condition causing the deadlock and preempting resources from one or more processes which are deadlocked.

POSSIBLE QUESTIONS

PART –B

(Each Question carries 2 Marks)

1. What is Process?
2. What is Process Control?
3. List the Basic OS Functions?
4. What is an Operating System?
5. What is critical section?
6. What is semaphore?
7. What are all basic conditions for deadlock?
8. Define: Software
9. Define: Hardware
10. What is IPC?

PART –C

(Each Question carries 6 Marks)

1. Write banker's algorithm and explain.
2. Write a short notes on inter process communication.
3. Explain the Types of Operating System.
4. Explain Basic OS Functions
5. Explain in detail about Multiprogramming Systems
6. Discuss about Process Control Block in detail
7. Explain in detail about Distributed System
8. Discuss about the overview of Operating System in detail
9. Discuss about (i) Batch System (ii) Real time System
10. Explain in detail about the Resource Abstraction.

Storage management: Memory Management - swapping- Contiguous memory allocation – paging, segmentation – segmentation with paging – Virtual memory :Virtual storage organization – Demand Paging, Process Creation – Page replacement – Thrashing.

Storage management

Memory Management

Main Memory refers to a physical memory that is the internal memory to the computer. The word main is used to distinguish it from external mass storage devices such as disk drives. Main memory is also known as RAM. The computer is able to change only data that is in main memory. Therefore, every program we execute and every file we access must be copied from a storage device into main memory.

All the programs are loaded in the main memeory for execution. Sometimes complete program is loaded into the memory, but some times a certain part or routine of the program is loaded into the main memory only when it is called by the program, this mechanism is called **Dynamic Loading**, this enhance the performance.

Also, at times one program is dependent on some other program. In such a case, rather than loading all the dependent programs, CPU links the dependent programs to the main executing program when its required. This mechanism is known as **Dynamic Linking**.

Swapping

A process needs to be in memory for execution. But sometimes there is not enough main memory to hold all the currently active processes in a timesharing system. So, excess process are kept on disk and brought in to run dynamically. Swapping is the process of bringing in each process in main memory, running it for a while and then putting it back to the disk.

Contiguous Memory Allocation

In contiguous memory allocation each process is contained in a single contiguous block of memory. Memory is divided into several fixed size partitions. Each partition contains exactly one process. When a partition is free, a process is selected from the input queue and loaded into it. The free blocks of memory are known as *holes*. The set of holes is searched to determine which hole is best to allocate.

Memory Protection

Memory protection is a phenomenon by which we control memory access rights on a computer. The main aim of it is to prevent a process from accessing memory that has not been allocated to it. Hence prevents a bug within a process from affecting other processes, or the operating system itself, and instead results in a segmentation fault or storage violation exception being sent to the disturbing process, generally killing of process.

Memory Allocation

Memory allocation is a process by which computer programs are assigned memory or space. It is of three types :

1. **First Fit**

   The first hole that is big enough is allocated to program.

2. **Best Fit**

   The smallest hole that is big enough is allocated to program.

3. **Worst Fit**

   The largest hole that is big enough is allocated to program.

Fragmentation

Fragmentation occurs in a dynamic memory allocation system when most of the free blocks are too small to satisfy any request. It is generally termed as inability to use the available memory.

In such situation processes are loaded and removed from the memory. As a result of this, free holes exists to satisfy a request but is non contiguous i.e. the memory is fragmented into large no. Of small holes. This phenomenon is known as **External Fragmentation.**

Also, at times the physical memory is broken into fixed size blocks and memory is allocated in unit of block sizes. The memory allocated to a space may be slightly larger than the requested memory. The difference between allocated and required memory is known as **Internal fragmentation** i.e. the memory that is internal to a partition but is of no use.

Paging

A solution to fragmentation problem is Paging. Paging is a memory management mechanism that allows the physical address space of a process to be non-contagious. Here physical memory is divided into blocks of equal size called **Pages**. The pages belonging to a certain process are loaded into available memory frames.

Page Table

A Page Table is the data structure used by a virtual memory system in a computer operating system to store the mapping between *virtual address* and *physical addresses.*

Virtual address is also known as Logical address and is generated by the CPU. While Physical address is the address that actually exists on memory.

Segmentation

Segmentation is another memory management scheme that supports the user-view of memory. Segmentation allows breaking of the virtual address space of a single process into segments that may be placed in non-contiguous areas of physical memory.

Segmentation with Paging

Both paging and segmentation have their advantages and disadvantages, it is better to combine these two schemes to improve on each. The combined scheme is known as 'Page the Elements'. Each segment in this scheme is divided into pages and each segment is maintained in a page table. So the logical address is divided into following 3 parts :

- Segment numbers(S)
- Page number (P)
- The displacement or offset number (D)

Virtual Memory

Virtual Memory is a space where large programs can store themselves in form of pages while their execution and only the required pages or portions of processes are loaded into the main memory. This technique is useful as large virtual memory is provided for user programs when a very small physical memory is there.

In real scenarios, most processes never need all their pages at once, for following reasons :

- Error handling code is not needed unless that specific error occurs, some of which are quite rare.
- Arrays are often over-sized for worst-case scenarios, and only a small fraction of the arrays are actually used in practice.
- Certain features of certain programs are rarely used.

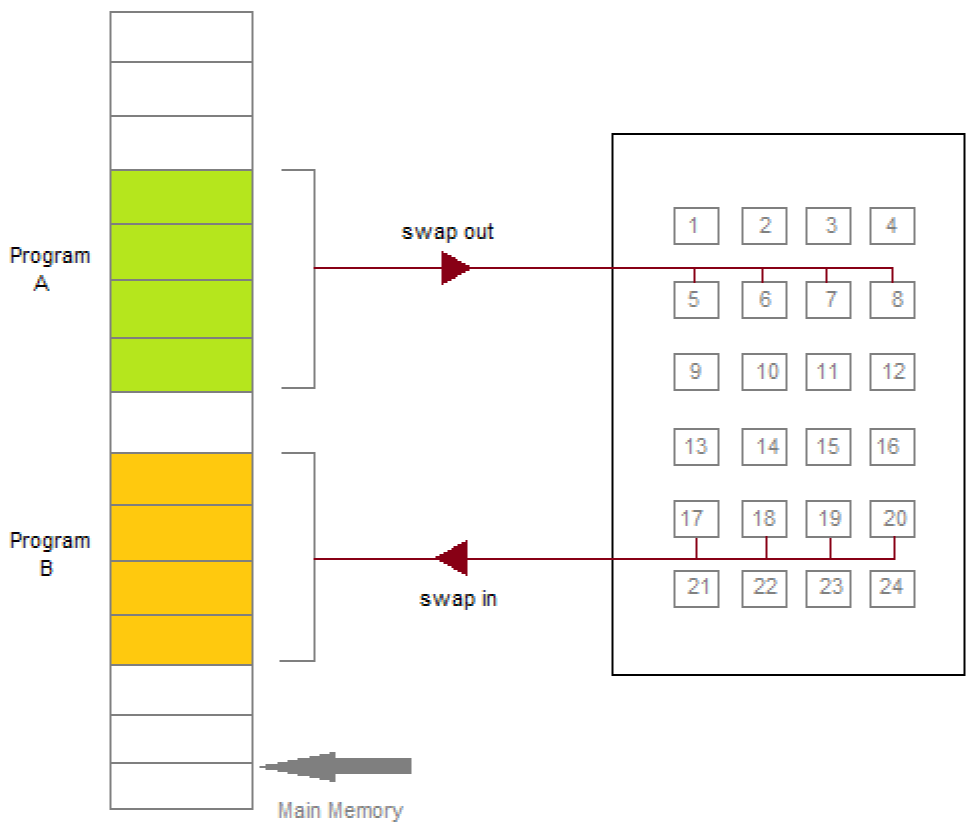
Benefits of having Virtual Memory :

1. Large programs can be written, as virtual space available is huge compared to physical memory.
2. Less I/O required, leads to faster and easy swapping of processes.

3. More physical memory available, as programs are stored on virtual memory, so they occupy very less space on actual physical memory.

Demand Paging

The basic idea behind demand paging is that when a process is swapped in, its pages are not swapped in all at once. Rather they are swapped in only when the process needs them(On demand). This is termed as lazy swapper, although a pager is a more accurate term.



Initially only those pages are loaded which will be required the process immediately.

The pages that are not moved into the memory, are marked as invalid in the page table. For an invalid entry the rest of the table is empty. In case of pages that are loaded in the memory, they are marked as valid along with the information about where to find the swapped out page.

When the process requires any of the page that is not loaded into the memory, a page fault trap is triggered and following steps are followed,

1. The memory address which is requested by the process is first checked, to verify the request made by the process.
2. If its found to be invalid, the process is terminated.

3. In case the request by the process is valid, a free frame is located, possibly from a free-frame list, where the required page will be moved.
4. A new operation is scheduled to move the necessary page from disk to the specified memory location. (This will usually block the process on an I/O wait, allowing some other process to use the CPU in the meantime.)
5. When the I/O operation is complete, the process's page table is updated with the new frame number, and the invalid bit is changed to valid.
6. The instruction that caused the page fault must now be restarted from the beginning.

There are cases when no pages are loaded into the memory initially, pages are only loaded when demanded by the process by generating page faults. This is called **Pure Demand Paging**.

The only major issue with Demand Paging is, after a new page is loaded, the process starts execution from the beginning. Its is not a big issue for small programs, but for larger programs it affects performance drastically.

Page Replacement

As studied in Demand Paging, only certain pages of a process are loaded initially into the memory. This allows us to get more number of processes into the memory at the same time. but what happens when a process requests for more pages and no free memory is available to bring them in. Following steps can be taken to deal with this problem :

1. Put the process in the wait queue, until any other process finishes its execution thereby freeing frames.
2. Or, remove some other process completely from the memory to free frames.
3. Or, find some pages that are not being used right now, move them to the disk to get free frames. This technique is called **Page replacement** and is most commonly used. We have some great algorithms to carry on page replacement efficiently.

Basic Page Replacement

- Find the location of the page requested by ongoing process on the disk.
- Find a free frame. If there is a free frame, use it. If there is no free frame, use a page-replacement algorithm to select any existing frame to be replaced, such frame is known as **victim frame**.
- Write the victim frame to disk. Change all related page tables to indicate that this page is no longer in memory.
- Move the required page and store it in the frame. Adjust all related page and frame tables to indicate the change.
- Restart the process that was waiting for this page.

FIFO Page Replacement

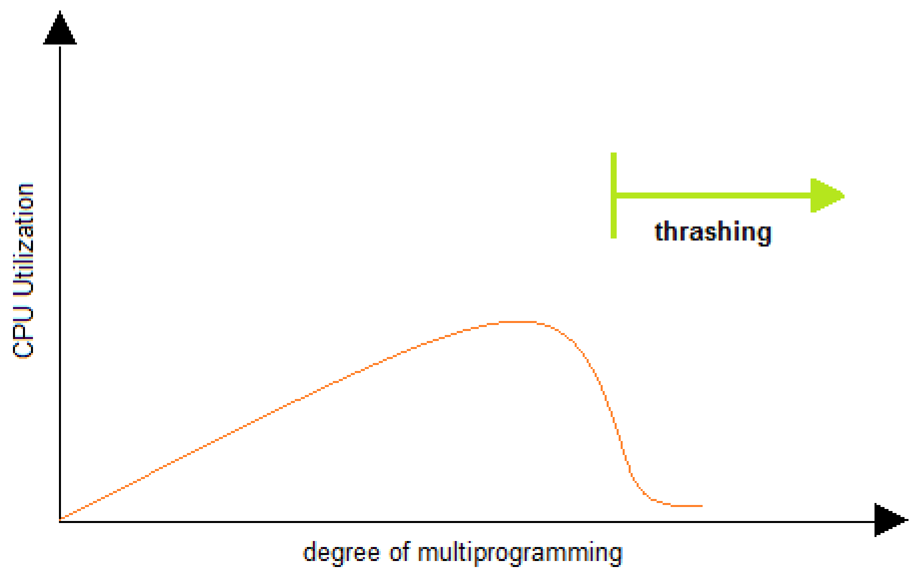
- A very simple way of Page replacement is FIFO (First in First Out)

- As new pages are requested and are swapped in, they are added to tail of a queue and the page which is at the head becomes the victim.
- Its not an effective way of page replacement but can be used for small systems.

Thrashing

A process that is spending more time paging than executing is said to be thrashing. In other words it means, that the process doesn't have enough frames to hold all the pages for its execution, so it is swapping pages in and out very frequently to keep executing. Sometimes, the pages which will be required in the near future have to be swapped out.

Initially when the CPU utilization is low, the process scheduling mechanism, to increase the level of multiprogramming loads multiple processes into the memory at the same time, allocating a limited amount of frames to each process. As the memory fills up, process starts to spend a lot of time for the required pages to be swapped in, again leading to low CPU utilization because most of the proccesses are waiting for pages. Hence the scheduler loads more processes to increase CPU utilization, as this continues at a point of time the complete system comes to a stop.



To prevent thrashing we must provide processes with as many frames as they really need "right now".

POSSIBLE QUESTIONS

PART –B

(Each Question carries 2 Marks)

1.What is memory?

2.Define: Swapping

3.What is paging?

4.Define:Segmentation

5.Define:word in memory

6.What is thrashing?

7.Define: Semaphore

8.How can you achieve page replacement?

9.What is virtual storage organization?

10. Define: Virtual memory

PART –C

(Each Question carries 6 Marks)

1. Explain about Memory Allocation Strategies.
2. Explain about Fixed and Variable partition.
3. Explain the Comparison between paging and Fragmentation
4. Difference between Physical address space and Virtual Address space
5. Discuss about swapping of two processes
6. Difference between Paging and Segmentation
7. Explain the concept of Physical address space in detail.
8. Explain in detail about Segmentation.
9. Explain about Virtual address space.
10. Discuss about Paging in detail

Processor Scheduling : preemptive scheduling : - Scheduling Criteria – Scheduling Algorithms – FCFS-SJF- Priority – RoundRobin –Multilevel Queue – Multilevel Feedback Queue . Multiprocess schedule: Real time schedule, Algorithm evaluation: Deterministic Modeling, Queue Model, Simulation

Process Scheduling

The act of determining which process in the ready state should be moved to the running state is known as Process Scheduling.

The prime aim of the process scheduling system is to keep the CPU busy all the time and to deliver minimum response time for all programs. For achieving this, the scheduler must apply appropriate rules for swapping processes IN and OUT of CPU.

Schedulers fell into one of the two general categories :

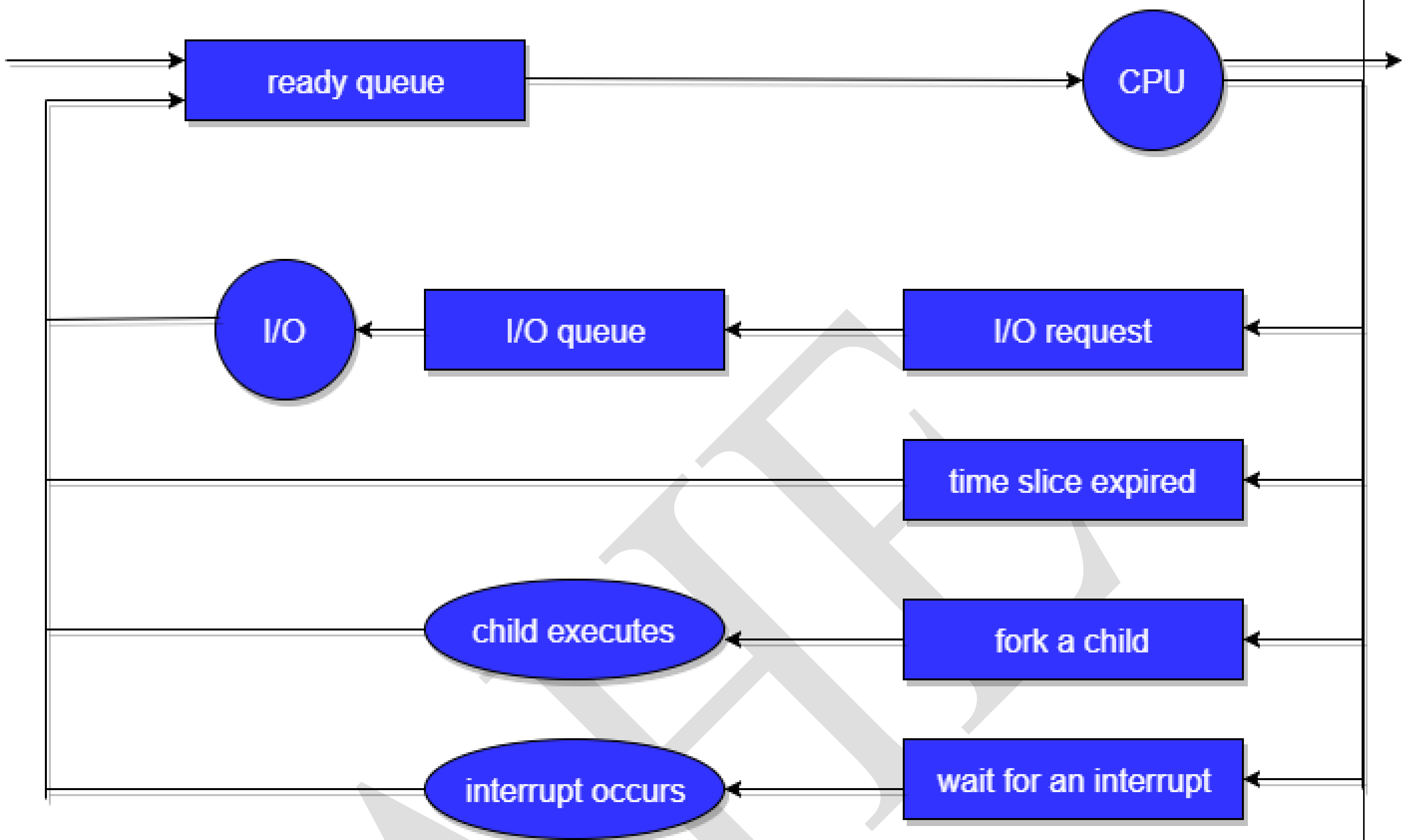
- **Non pre-emptive scheduling.** When the currently executing process gives up the CPU voluntarily.
- **Pre-emptive scheduling.** When the operating system decides to favour another process, pre-empting the currently executing process.

Scheduling Queues

- All processes when enters into the system are stored in the **job queue**.
- Processes in the Ready state are placed in the **ready queue**.
- Processes waiting for a device to become available are placed in **device queues**. There are unique device queues for each I/O device available.

A new process is initially put in the ready queue. It waits in the ready queue until it is selected for execution(or dispatched). Once the process is assigned to the CPU and is executing, once of several events could occur.

- The process could issue an I/O request, and then be placed in an I/O queue.
- The process could create a new subprocess and wait for its termination.
- The process could be removed forcibly from the CPU, as a result of an interrupt, and be put back in the ready queue.



In the first two cases, the process eventually switches from the waiting state to the ready state, and is then put back in the ready queue. A process continues this cycle until it terminates, at which time it is removed from all queues and has its PCB and resources deallocated.

Types of Schedulers

There are three types of schedulers available :

1. **Long Term Scheduler** :

   Long term scheduler runs less frequently. Long Term Schedulers decide which program must get into the job queue. From the job queue, the Job Processor, selects processes and loads them into the memory for execution. Primary aim of the Job Scheduler is to maintain a good degree of Multiprogramming. An optimal degree of Multiprogramming means the average rate of process creation is equal to the average departure rate of processes from the execution memory.

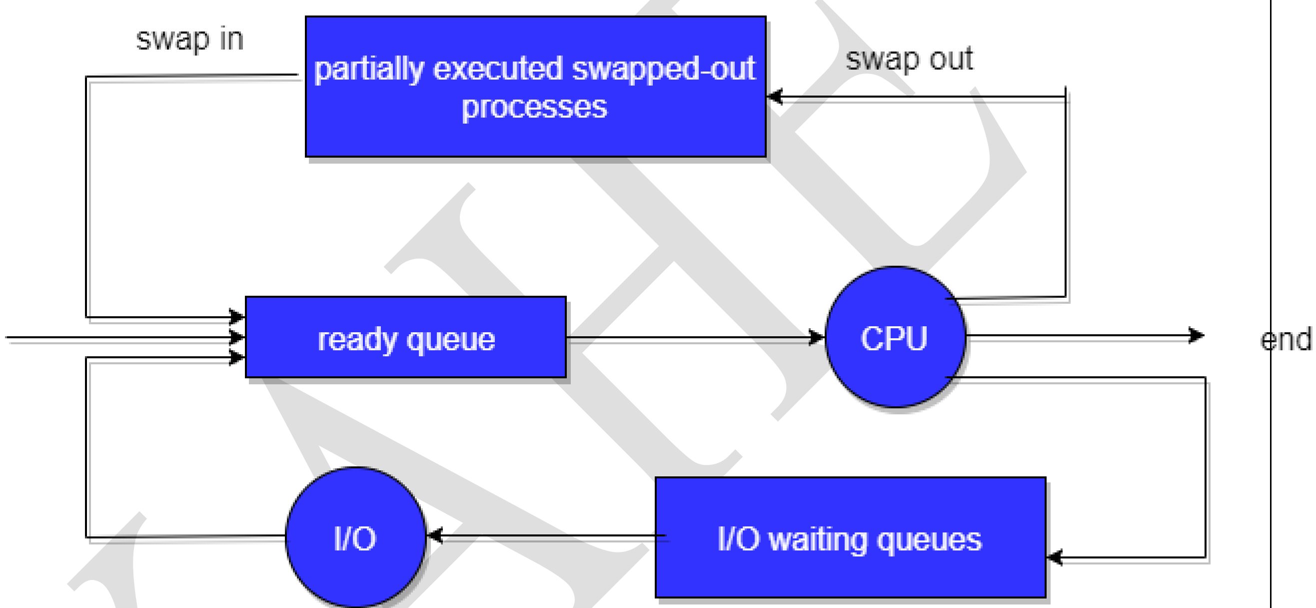
2. **Short Term Scheduler** :

   This is also known as CPU Scheduler and runs very frequently. The primary aim of this scheduler is to enhance CPU performance and increase process execution rate.

3. **Medium Term Scheduler** :

This scheduler removes the processes from memory (and from active contention for the CPU), and thus reduces the degree of multiprogramming. At some later time, the process can be reintroduced into memory and its execution van be continued where it left off. This scheme is called **swapping**. The process is swapped out, and is later swapped in, by the medium term scheduler.

Swapping may be necessary to improve the process mix, or because a change in memory requirements has overcommitted available memory, requiring memory to be freed up. This complete process is descripted in the below diagram:



Addition of medium-term scheduling to the queueing diagram.

Context Switch

- Switching the CPU to another process requires **saving** the state of the old process and **loading** the saved state for the new process. This task is known as a **context switch**.
- The **context** of a process is represented in the **Process Control Block(PCB)** of a process; it includes the value of the CPU registers, the process state and memory-management information. When a context switch occurs, the Kernel saves the context of the old process in its PCB and loads the saved context of the new process scheduled to run.
- Context switch time is **pure overhead**, because the **system does no useful work while switching**. Its speed varies from machine to machine, depending on the memory speed, the number of registers that must be copied, and the existence of special instructions(such as a single instruction to load or store all registers). Typical speeds range from 1 to 1000 microseconds.
- Context Switching has become such a performance **bottleneck** that programmers are using new structures(threads) to avoid it whenever possible.

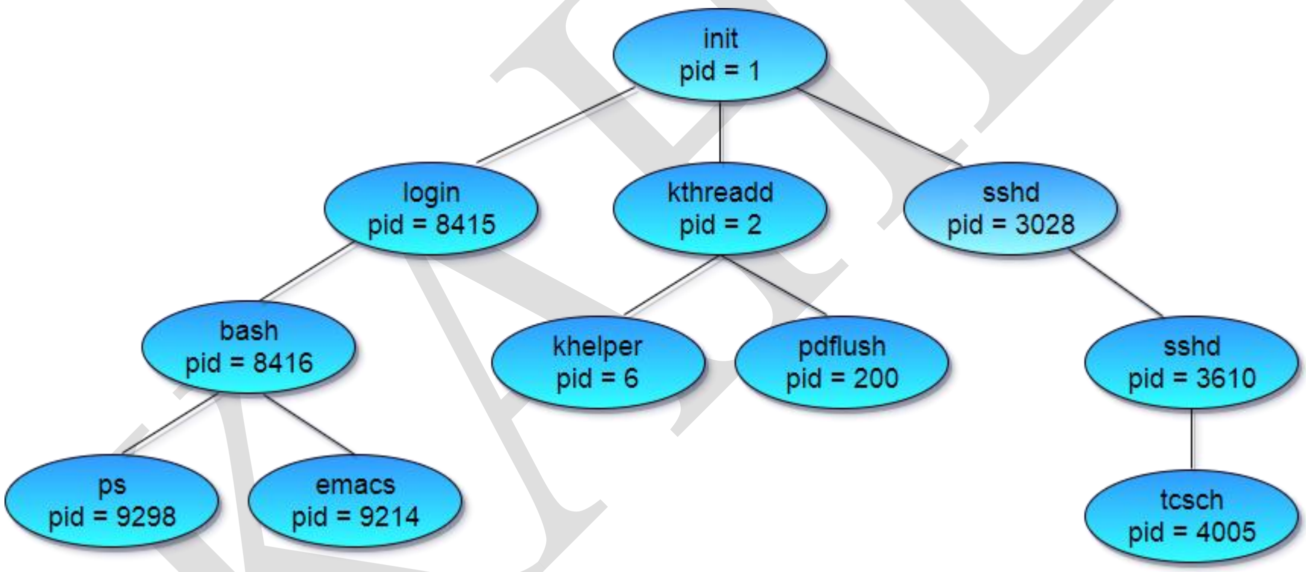
Operations on Process

Process Creation

Through appropriate system calls, such as fork or spawn, processes may create other processes. The process which creates other process, is termed the **parent** of the other process, while the created sub-process is termed its **child**.

Each process is given an integer identifier, termed as process identifier, or PID. The parent PID (PPID) is also stored for each process.

On a typical UNIX systems the process scheduler is termed as sched, and is given PID 0. The first thing done by it at system start-up time is to launch init, which gives that process PID 1. Further Init launches all the system daemons and user logins, and becomes the ultimate parent of all other processes.



A child process may receive some amount of shared resources with its parent depending on system implementation. To prevent runaway children from consuming all of a certain system resource, child processes may or may not be limited to a subset of the resources originally allocated to the parent.

There are two options for the parent process after creating the child :

- Wait for the child process to terminate before proceeding. Parent process makes a wait() system call, for either a specific child process or for any particular child process, which causes the parent process to block until the wait() returns. UNIX shells normally wait for their children to complete before issuing a new prompt.
- Run concurrently with the child, continuing to process without waiting. When a UNIX shell runs a process as a background task, this is the operation seen. It is also possible for the parent to run for a while, and then wait for the child later, which might occur in a sort of a parallel processing operation.

There are also two possibilities in terms of the address space of the new process:

1. The child process is a duplicate of the parent process.
2. The child process has a program loaded into it.

To illustrate these different implementations, let us consider the **UNIX** operating system. In UNIX, each process is identified by its **process identifier**, which is a unique integer. A new process is created by the **fork** system call. The new process consists of a copy of the address space of the original process. This mechanism allows the parent process to communicate easily with its child process. Both processes (the parent and the child) continue execution at the instruction after the fork system call, with one difference: **The return code for the fork system call is zero for the new(child) process, whereas the(non zero) process identifier of the child is returned to the parent.**

Typically, the **execlp system call** is used after the fork system call by one of the two processes to replace the process memory space with a new program. The execlp system call loads a binary file into memory - destroying the memory image of the program containing the execlp system call – and starts its execution. In this manner the two processes are able to communicate, and then to go their separate ways.

Below is a C program to illustrate forking a separate process using UNIX(made using Ubuntu):

```
#include<stdio.h>

void main(int argc, char *argv[])
{

    int pid;

    /* Fork another process */
    pid=fork();

    if(pid<0)
    {
        //Error occurred
        fprintf(stderr, "Fork Failed");
        exit(-1);
    }
    else if (pid == 0)
    {
        //Child process
        execlp("/bin/ls","ls",NULL);
    }
    else
    {
        //Parent process
        //Parent will wait for the child to complete
        wait(NULL);
        printf("Child complete");
        exit(0);
    }

}
```

Gate Numerical Tip: *if fork is called for **n** times, the number of child process or new process created are: **2^n – 1**.*

Process Termination

By making the exit(system call), typically returning an int, processes may request their own termination. This int is passed along to the parent if it is doing a wait(), and is typically zero on successful completion and some non-zero code in the event of any problem.

Processes may also be terminated by the system for a variety of reasons, including :

- The inability of the system to deliver the necessary system resources.
- In response to a KILL command or other unhandled process interrupts.
- A parent may kill its children if the task assigned to them is no longer needed i.e. if the need of having a child terminates.
- If the parent exits, the system may or may not allow the child to continue without a parent (In UNIX systems, orphaned processes are generally inherited by init, which then proceeds to kill them.)

When a process ends, all of its system resources are freed up, open files flushed and closed, etc. The process termination status and execution times are returned to the parent if the parent is waiting for the child to terminate, or eventually returned to init if the process already became an orphan.

The processes which are trying to terminate but cannot do so because their parent is not waiting for them are termed **zombies**. These are eventually inherited by init as orphans and killed off.

Operating System Scheduling algorithms

A Process Scheduler schedules different processes to be assigned to the CPU based on particular scheduling algorithms. There are six popular process scheduling algorithms which we are going to discuss in this chapter −

- First-Come, First-Served (FCFS) Scheduling
- Shortest-Job-Next (SJN) Scheduling
- Priority Scheduling
- Shortest Remaining Time
- Round Robin(RR) Scheduling
- Multiple-Level Queues Scheduling

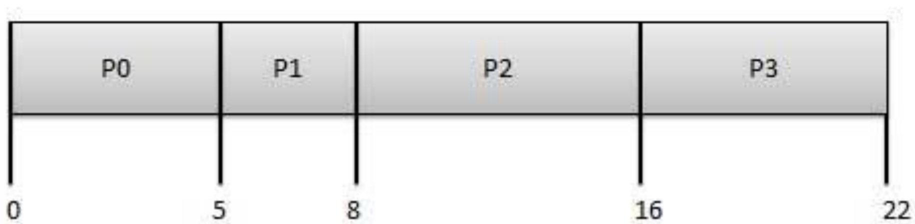
These algorithms are either **non-preemptive or preemptive**. Non-preemptive algorithms are designed so that once a process enters the running state, it cannot be preempted until it completes its allotted time, whereas the preemptive scheduling is based on priority where a scheduler may preempt a low priority running process anytime when a high priority process enters into a ready state.

First Come First Serve (FCFS)

- Jobs are executed on first come, first serve basis.
- It is a non-preemptive, pre-emptive scheduling algorithm.
- Easy to understand and implement.
- Its implementation is based on FIFO queue.

- Poor in performance as average wait time is high.

Process	Arrival Time	Execute Time	Service Time
P0	0	5	0
P1	1	3	5
P2	2	8	8
P3	3	6	16



Wait time of each process is as follows −

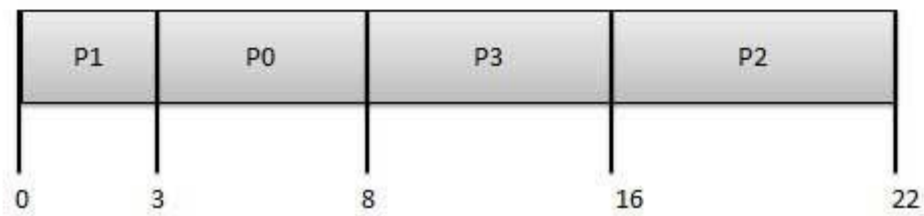
Process	**Wait Time : Service Time - Arrival Time**
P0	0 - 0 = 0
P1	5 - 1 = 4
P2	8 - 2 = 6
P3	16 - 3 = 13

Average Wait Time: (0+4+6+13) / 4 = 5.75

Shortest Job Next (SJN)

- This is also known as **shortest job first**, or SJF
- This is a non-preemptive, pre-emptive scheduling algorithm.
- Best approach to minimize waiting time.
- Easy to implement in Batch systems where required CPU time is known in advance.
- Impossible to implement in interactive systems where required CPU time is not known.
- The processer should know in advance how much time process will take.

Process	Arrival Time	Execute Time	Service Time
P0	0	5	3
P1	1	3	0
P2	2	8	16
P3	3	6	8



Wait time of each process is as follows −

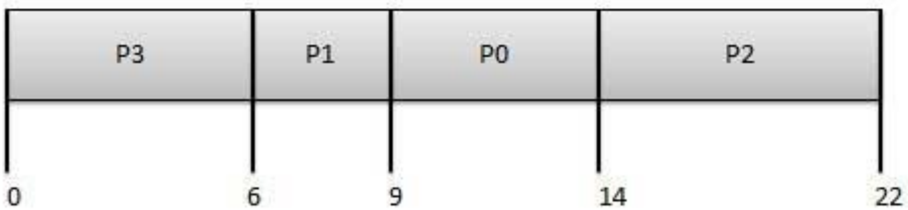
Process	**Wait Time : Service Time - Arrival Time**
P0	3 - 0 = 3
P1	0 - 0 = 0
P2	16 - 2 = 14
P3	8 - 3 = 5

Average Wait Time: (3+0+14+5) / 4 = 5.50

Priority Based Scheduling

- Priority scheduling is a non-preemptive algorithm and one of the most common scheduling algorithms in batch systems.
- Each process is assigned a priority. Process with highest priority is to be executed first and so on.
- Processes with same priority are executed on first come first served basis.
- Priority can be decided based on memory requirements, time requirements or any other resource requirement.

Process	Arrival Time	Execute Time	Priority	Service Time
P0	0	5	1	9
P1	1	3	2	6
P2	2	8	1	14
P3	3	6	3	0



Wait time of each process is as follows −

Process	Wait Time : Service Time - Arrival Time
P0	9 - 0 = 9
P1	6 - 1 = 5
P2	14 - 2 = 12
P3	0 - 0 = 0

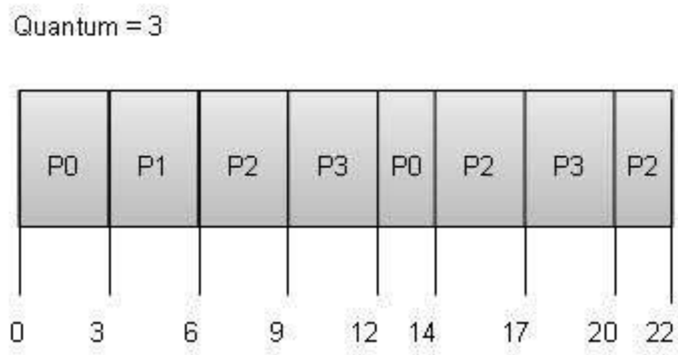
Average Wait Time: (9+5+12+0) / 4 = 6.5

Shortest Remaining Time

- Shortest remaining time (SRT) is the preemptive version of the SJN algorithm.
- The processor is allocated to the job closest to completion but it can be preempted by a newer ready job with shorter time to completion.
- Impossible to implement in interactive systems where required CPU time is not known.
- It is often used in batch environments where short jobs need to give preference.

Round Robin Scheduling

- Round Robin is the preemptive process scheduling algorithm.
- Each process is provided a fix time to execute, it is called a **quantum**.
- Once a process is executed for a given time period, it is preempted and other process executes for a given time period.
- Context switching is used to save states of preempted processes.



Wait time of each process is as follows −

Process	**Wait Time : Service Time - Arrival Time**
P0	(0 - 0) + (12 - 3) = 9
P1	(3 - 1) = 2
P2	(6 - 2) + (14 - 9) + (20 - 17) = 12
P3	(9 - 3) + (17 - 12) = 11

Average Wait Time: (9+2+12+11) / 4 = 8.5

Multiple-Level Queues Scheduling

Multiple-level queues are not an independent scheduling algorithm. They make use of other existing algorithms to group and schedule jobs with common characteristics.

- Multiple queues are maintained for processes with common characteristics.
- Each queue can have its own scheduling algorithms.
- Priorities are assigned to each queue.

For example, CPU-bound jobs can be scheduled in one queue and all I/O-bound jobs in another queue. The Process Scheduler then alternately selects jobs from each queue and assigns them to the CPU based on the algorithm assigned to the queue.

Operating System | Multiple-Processor Scheduling

1

In multiple-processor scheduling **multiple CPU's** are available and hence **Load Sharing** becomes possible. However multiple processor scheduling is more **complex** as compared to single processor scheduling. In multiple processor scheduling there are cases when the processors are identical i.e. HOMOGENEOUS, in terms of their functionality, we can use any processor available to run any process in the queue.

Approaches to Multiple-Processor Scheduling –

One approach is when all the scheduling decisions and I/O processing are handled by a single processor which is called the **Master Server** and the other processors executes only the **user code**. This is simple and reduces the need of data sharing. This entire scenario is called **Asymmetric Multiprocessing**.

A second approach uses **Symmetric Multiprocessing** where each processor is **self scheduling**. All processes may be in a common ready queue or each processor may have its own private queue for ready processes. The scheduling proceeds further by having the scheduler for each processor examine the ready queue and select a process to execute.

Processor Affinity –

Processor Affinity means a processes has an **affinity** for the processor on which it is currently running. When a process runs on a specific processor there are certain effects on the cache memory. The data most recently accessed by the process populate the cache for the processor and as a result successive memory access by the process are often satisfied in the cache memory. Now if the process migrates to another processor, the contents of the cache memory must be invalidated for the first processor and the cache for the second processor must be repopulated. Because of the high cost of invalidating and repopulating caches, most of the SMP(symmetric multiprocessing) systems try to avoid migration of processes from one processor to another and try to keep a process running on the same processor. This is known as **PROCESSOR AFFINITY**.

There are two types of processor affinity:

1. **Soft Affinity –** When an operating system has a policy of attempting to keep a process running on the same processor but not guaranteeing it will do so, this situation is called soft affinity.
2. **Hard Affinity –** Some systems such as Linux also provide some system calls that support Hard Affinity which allows a process to migrate between processors.

Load Balancing –

Load Balancing is the **phenomena** which keeps the **workload** evenly **distributed** across all processors in an SMP system. Load balancing is necessary only on systems where each processor has its own private queue of process which are eligible to execute. Load balancing is unnecessary because once a processor becomes idle it immediately extracts a runnable process from the common run queue. On SMP(symmetric multiprocessing), it is important to keep the workload balanced among all processors to fully utilize the benefits of having more than one processor else one or more processor will sit idle while other processors have high workloads along with lists of processors awaiting the CPU.

There are two general approaches to load balancing :

1. **Push Migration –** In push migration a task routinely checks the load on each processor and if it finds an imbalance then it evenly distributes load on each processors by moving the processes from overloaded to idle or less busy processors.
2. **Pull Migration –** Pull Migration occurs when an idle processor pulls a waiting task from a busy processor for its execution.

Multicore Processors –

In multicore processors **multiple processor** cores are places on the same physical chip. Each core has a register set to maintain its architectural state and thus appears to the operating system as a separate physical processor. **SMP systems** that use multicore processors are faster and consume **less power** than systems in which each processor has its own physical chip.

However multicore processors may **complicate** the scheduling problems. When processor accesses memory then it spends a significant amount of time waiting for the data to become available. This situation is called **MEMORY STALL**. It occurs for various reasons such as cache miss, which is accessing the data that is not in the cache memory. In such cases the processor can spend upto fifty percent of its time waiting for data to become available from the memory. To solve this problem recent hardware designs have implemented multithreaded processor cores in which two or more hardware threads are assigned to each core. Therefore if one thread stalls while waiting for the memory, core can switch to another thread.

There are two ways to multithread a processor :

1. **Coarse-Grained Multithreading –** In coarse grained multithreading a thread executes on a processor until a long latency event such as a memory stall occurs, because of the delay caused by the long latency event, the processor must switch to another thread to begin execution. The cost of switching between threads is high as the instruction pipeline must be terminated before the other thread can begin execution on the processor core. Once this new thread begins execution it begins filling the pipeline with its instructions.
2. **Fine-Grained Multithreading –** This multithreading switches between threads at a much finer level mainly at the boundary of an instruction cycle. The architectural design of fine grained systems include logic for thread switching and as a result the cost of switching between threads is small.

Virtualization and Threading –

In this type of **multiple-processor** scheduling even a single CPU system acts like a multiple-processor system. In a system with Virtualization, the virtualization presents one or more virtual CPU's to each of virtual machines running on the system and then schedules the use of physical CPU'S among the virtual machines. Most virtualized environments have one host operating system and many guest operating systems. The host operating system creates and manages the virtual machines and each virtual machine has a guest operating system installed and applications running within that guest.Each guest operating system may be assigned for specific use cases,applications, and users,including time sharing or even real-time operation. Any guest operating-system scheduling algorithm that assumes a certain amount of progress in a given amount of time will be negatively impacted by the virtualization. In a time sharing operating system that tries to allot 100 milliseconds to each time slice to give users a reasonable response time. A given 100 millisecond time slice may take much more than 100 milliseconds of virtual CPU time. Depending on how busy the system is, the time slice may take a second or more which results in a very poor response time for users logged into that virtual machine. The net effect of such scheduling layering is that individual virtualized operating systems receive only a portion of the available CPU cycles, even though they believe they are receiving all cycles and that they are scheduling all of those cycles.Commonly, the time-of-day clocks in virtual machines are incorrect because timers take no longer to trigger than they would on dedicated CPU's.

Virtualizations can thus undo the good scheduling-algorithm efforts of the operating systems within virtual machines.

Real Time Schedule

Real-time operating systems

If you are serious about meeting hard deadlines or designing a safety critical system, you will likely need to run a **real-time operating system** on a system that is dedicated to those tasks and minimizes all other forms of interference. For example, you may not be able to afford wasting time servicing disk interrupts and you certainly will not want to move memory pages back and forth between the disk and memory for fear of the time it will take to retrieve them if they are suddenly needed. You also will need to ensure that your operating system has preemptable system calls since you don't want a process held up because the operating system is tied up servicing a system call.

A real-time operating system has a well-specified maximum time for each action that it performs to support applications with precise timing needs. Systems that can guarantee these maximum times are called **hard real-time** systems. Those that meet these times most of the time are called **soft real-time** systems. Deploying an airbag in response to a sensor being actuated is a case where you would want a hard real-time system. Decoding video frames is an example of where a soft real-time system will suffice. Real-time systems will usually have the following characteristics:

- Priority-based scheduler
- Guaranteed maximum time to service interrupts
- Ability to ensure that processes are fully loaded into memory and stay in memory
- Consistent and efficient memory allocation
- Preemptable system calls

Process types

As we start to use terms such as *compute time* and *deadline*, it helps to see how these terms relate to different categories of processes:

1. **Terminating processes**: A terminating process may be considered as one that runs and then exits (terminates). We are interested in the amount of time that it takes it to run to completion. Its deadline is the time that at which it should complete all its work and its compute time is the amount of CPU time it needs.
2. **Nonterminating processes**: For processes such as video and audio servers as well as editors, we are not interested in the terminating time of these processes but rather in the time between events. For example, we would like our audio server to fill a 4K byte audio buffer every 500 milliseconds or we would like our video server to provide a new video frame every 33.3 milliseconds. For these processes, the compute time is the CPU time that the process needs to compute its periodic event and the deadline is the time at which it must have the results ready. Nonterminating processes may be divided into two classes:
    - **Periodic**: A periodic process has a fixed frequency at which it needs to run. For example, a video decompressor may have to run 30 times per second at 33.3 millisecond intervals.
    - **Aperiodic**: Aperiodic processes have no fixed, cyclical, frequency between events. Event interrupts may occur sporadically and event computation times may vary dramatically. For

purposes of scheduling, we use the shortest period and the longest computation time to play it safe.

How much can we do?

The CPU cannot work magic. If we want to have our system process four video streams at the same time at 30 frames per second and processing a single frame requires 40 milliseconds of CPU time, we will be forced to fail in our scheduling needs. There is just not enough CPU time to go around. If *C* represents our computation time and *D* represents the deadline, the following relation must hold:

$$C \leq D$$

This assures us that we will have enough CPU time to complete the task. Moreover, for periodic tasks, the deadline must be within the period. If the period of the process is *T*, the following relation must now hold:

$$C \leq D \leq T$$

Let's now look at a few popular algorithms for scheduling processes with real-time constraints.

Earliest deadline scheduling

Earliest deadline scheduling is simple in concept. Every process tells the operating system scheduler its absolute time deadline. The scheduling algorithm simply allows the process that is in the greatest danger of missing its deadline to run first. Generally, this means that one process will run to completion if it has an earlier deadline than another. The only time a process would be preempted would be when a new process with an even shorter deadline becomes ready to run. To determine whether all the scheduled processes are capable of having their deadlines met, the following condition must hold :

$$\sum_i \frac{C_i}{T_i} \leq 1$$

This simply tells us sum of all the percentages of CPU time used per process has to be less than or equal to 100%.

Least slack scheduling

This method is similar to shortest remaining time scheduling with the concept of a deadline thrown in. The goal is to pick the process that we can least afford to delay. This differs from earliest deadline scheduling because we're not looking only at the deadline, but at the amount of time we can procrastinate (work on something else) before we will have to put 100% of the CPU resource to finishing the task. Least slack is computed as the time to the deadline minus the amount of computation. For example, suppose that our remaining computation time, *C*, is 5 msec. and the deadline, *D*, is 20 msec. from now. The slack time is *D* - *C*, or 15 msec. The scheduler will compare this slack time with the slack times of other processes in the system and run the one with the lowest slack time.

The effect of least slack scheduling is significantly different from that of earliest deadline scheduling. With earliest deadline, we will always work on the process with the nearest deadline, delaying all the other processes. With least slack scheduling, we get a balanced result in that we attempt to keep the differences from deadlines balanced among processes. If the CPU has no problem meeting all deadlines, both scheduling policies will work satisfactorily. If there is too much of a workload and some deadlines must be missed, earliest deadline will satisfy the processes with the earliest deadlines (assuming all processes arrived at the same time) because it started working on them early. Processes with later deadlines will get delayed significantly. With least slack scheduling, all deadlines will be missed, but they all will be missed by roughly the same amount of time. Which is better? It depends on the applications. The same scheduling constraint applies to Least Slack scheduling as to Earliest Deadline First scheduling.

Rate monotonic analysis

Rate monotonic analysis is a technique for assigning static priorities to periodic processes. As such, it is not a scheduler but a mechanism for governing the behavior of a preemptive priority scheduler. A conventional priority scheduler is used with this system, where the highest priority ready process will always get scheduled, preempting any lower priority processes.

A scheduler that is aware of rate monotonic scheduling would be provided with process timing parameters (period of execution) when the process is created and compute a suitable priority for the process. Most schedulers that support priority scheduling (e.g., Windows, Linux, Solaris, FreeBSD, NetBSD) do not perform rate monotonic analysis but only allow fixed priorities, so it is up to the user to assign proper priority levels for all real-time processes on the system. To do this properly, the user must be aware of all the real-time processes that will be running at any given time and each process' frequency of execution (1/*T*, where *T* is the period). To determine whether all scheduled processes can have their real-time demands met, the system has to also know each process' compute needs per period (*C*) and check that the following condition holds:

$$\sum_i \frac{C_i}{T_i} < \ln 2$$

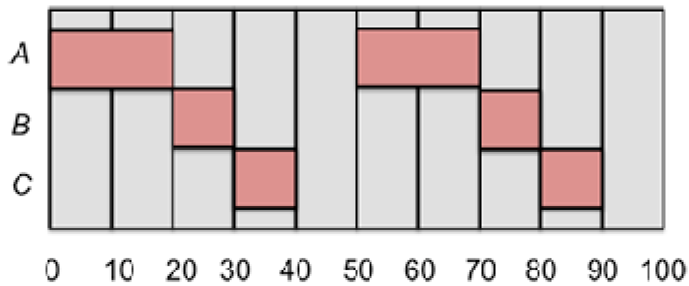
To assign a rate monotonic priority, one simply uses the frequency information for each process. If a process is an aperiodic process, the worst-case (fastest) frequency should be used. The highest frequency (smallest period) process gets the highest priority and successively lower frequency processes get lower priorities.

Scheduling is performed by a simple priority scheduler. At each quantum, the highest priority ready process gets to run. Processes at the same priority level run round-robin.

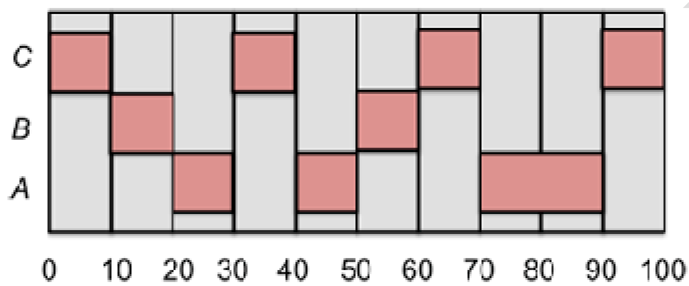
Here is an example of ratemonotonic priority assignment. Suppose we have the following processes:

A runs every 50 msec for 20 msec
B runs every 50 msec for 10 msec
C runs every 30 msec for 10 msec

Rate-monotonic assignment requires that the highest frequency process(es) (*B* and *C*) get the highest priority and *A*, having the lowest frequency of execution, gets a lower priority. If we do not follow the rules and give *A* the highest priority, *B* the next highest, and *C* the lowest, the CPU will run processes in the following order:



This does not give us the desired performance because, while processes *A* and *B* get scheduled acceptably, process *C* is late the second time it is scheduled and misses an entire period! Now let us reverse the priorities as ratemonotonic assignment would dictate:



The scheduler can now satisfactorily meet the real-time requirements these tasks. Rate monotonic priority assignment is guaranteed to be optimal. If processes cannot be scheduled using rate monotonic assignment, the processes cannot be properly scheduled with any other static priority assignment.

Deterministic Modeling

This evaluation method takes a predetermined workload and evaluates each algorithm using that workload.

Assume we are presented with the following processes, which all arrive at time zero.

Process	Burst Time
P1	9
P2	33
P3	2
P4	5
P5	14

Which of the following algorithms will perform best on this workload?

First Come First Served (FCFS), Non Preemptive Shortest Job First (SJF) and Round Robin (RR). Assume a quantum of 8 milliseconds.

Before looking at the answers, try to calculate the figures for each algorithm.

The advantages of deterministic modeling is that it is exact and fast to compute. The disadvantage is that it is only applicable to the workload that you use to test. As an example, use the above workload but assume P1 only has a burst time of 8 milliseconds. What does this do to the average waiting time?

Of course, the workload might be typical and scale up but generally deterministic modeling is too specific and requires too much knowledge about the workload.

Queuing Models

Another method of evaluating scheduling algorithms is to use queuing theory. Using data from real processes we can arrive at a probability distribution for the length of a burst time and the I/O times for a process. We can now generate these times with a certain distribution.

We can also generate arrival times for processes (arrival time distribution).

If we define a queue for the CPU and a queue for each I/O device we can test the various scheduling algorithms using queuing theory.

Knowing the arrival rates and the service rates we can calculate various figures such as average queue length, average wait time, CPU utilization etc.

One useful formula is ***Little's Formula.***

$n = \lambda w$

Where

n is the average queue length
λ is the average arrival rate for new processes (e.g. five a second)
w is the average waiting time in the queue

Knowing two of these values we can, obviously, calculate the third. For example, if we know that eight processes arrive every second and there are normally sixteen processes in the queue we can compute that the average waiting time per process is two seconds.

The main disadvantage of using queuing models is that it is not always easy to define realistic distribution times and we have to make assumptions. This results in the model only being an approximation of what actually happens.

Simulations

Rather than using queuing models we simulate a computer. A Variable, representing a clock is incremented. At each increment the state of the simulation is updated.

Statistics are gathered at each clock tick so that the system performance can be analysed.

The data to drive the simulation can be generated in the same way as the queuing model, although this leads to similar problems.

Alternatively, we can use trace data. This is data collected from real processes on real machines and is fed into the simulation. This can often provide good results and good comparisons over a range of scheduling algorithms.

However, simulations can take a long time to run, can take a long time to implement and the trace data may be difficult to collect and require large amounts of storage.

Implementation

The best way to compare algorithms is to implement them on real machines. This will give the best results but does have a number of disadvantages.

· It is expensive as the algorithm has to be written and then implemented on real hardware.
· If typical workloads are to be monitored, the scheduling algorithm must be used in a live situation. Users may not be happy with an environment that is constantly changing.
· If we find a scheduling algorithm that performs well there is no guarantee that this state will continue if the workload or environment changes.

POSSIBLE QUESTIONS

PART –B

(Each Question carries 2 Marks)

1.Define : Process

2.Define: Scheduler

3.What is preemptive scheduling?

4.What is non preemptive scheduling?

5.List the scheduling algorithms

6.What is simulation?

7.What is modeling?

8.What is deterministic modeling?

9.What is real time schedule?

10. What is queue model?

PART –C

(Each Question carries 6 Marks)

1. Write short notes on FCFS scheduling algorithm.
2. Explain the shortest job first scheduling algorithm.
3. Write a algorithm for round robin and explain
4. Write a algorithm for priority queue and explain
5. Explain the multilevel queue algorithm
6. Discuss the concept of real time schedule
7. Discuss the concept of deterministic modeling in detail
8. Explain the queue model.
9. What is simulation? Discuss in detail
10. Discuss any two scheduling algorithms.

File systems: Introduction – File System Concepts – Access Methods – Directory structure – File Sharing – Allocation Methods – Free space management –Efficiency and performance – Recovery
Disk Performance Optimization: Introduction – Disk structure – Disk scheduling – Disk management.

File System:

File

A file is a named collection of related information that is recorded on secondary storage such as magnetic disks, magnetic tapes and optical disks. In general, a file is a sequence of bits, bytes, lines or records whose meaning is defined by the files creator and user.

File Structure

A File Structure should be according to a required format that the operating system can understand.

- A file has a certain defined structure according to its type.
- A text file is a sequence of characters organized into lines.
- A source file is a sequence of procedures and functions.
- An object file is a sequence of bytes organized into blocks that are understandable by the machine.
- When operating system defines different file structures, it also contains the code to support these file structure. Unix, MS-DOS support minimum number of file structure.

File Type

File type refers to the ability of the operating system to distinguish different types of file such as text files source files and binary files etc. Many operating systems support many types of files. Operating system like MS-DOS and UNIX have the following types of files −

Ordinary files

- These are the files that contain user information.
- These may have text, databases or executable program.
- The user can apply various operations on such files like add, modify, delete or even remove the entire file.

Directory files

- These files contain list of file names and other information related to these files.

Special files

- These files are also known as device files.
- These files represent physical device like disks, terminals, printers, networks, tape drive etc.

These files are of two types −

- **Character special files** − data is handled character by character as in case of terminals or printers.
- **Block special files** − data is handled in blocks as in the case of disks and tapes.

File Access Mechanisms

File access mechanism refers to the manner in which the records of a file may be accessed. There are several ways to access files −

- Sequential access
- Direct/Random access
- Indexed sequential access

Sequential access

A sequential access is that in which the records are accessed in some sequence, i.e., the information in the file is processed in order, one record after the other. This access method is the most primitive one. Example: Compilers usually access files in this fashion.

Direct/Random access

- Random access file organization provides, accessing the records directly.
- Each record has its own address on the file with by the help of which it can be directly accessed for reading or writing.
- The records need not be in any sequence within the file and they need not be in adjacent locations on the storage medium.

Indexed sequential access

- This mechanism is built up on base of sequential access.
- An index is created for each file which contains pointers to various blocks.
- Index is searched sequentially and its pointer is used to access the file directly.

Space Allocation

Files are allocated disk spaces by operating system. Operating systems deploy following three main ways to allocate disk space to files.

- Contiguous Allocation
- Linked Allocation
- Indexed Allocation

Contiguous Allocation

- Each file occupies a contiguous address space on disk.
- Assigned disk address is in linear order.

- Easy to implement.
- External fragmentation is a major issue with this type of allocation technique.

Linked Allocation

- Each file carries a list of links to disk blocks.
- Directory contains link / pointer to first block of a file.
- No external fragmentation
- Effectively used in sequential access file.
- Inefficient in case of direct access file.

Indexed Allocation

- Provides solutions to problems of contiguous and linked allocation.
- A index block is created having all pointers to files.
- Each file has its own index block which stores the addresses of disk space occupied by the file.
- Directory contains the addresses of index blocks of files.

File Systems | Operating System

A file is a collection of related information that is recorded on secondary storage. Or file is a collection of logically related entities. From user's perspective a file is the smallest allotment of logical secondary storage.

Attributes	Types	Operations
Name	Doc	Create
Type	Exe	Open
Size	Jpg	Read
Creation data	Xis	Write
Author	C	Append
Last modified	Java	Truncate
protection	class	Delete
		Close

There are various file types with their associated functions.

File type	Usual extension	Function
Executable	exe,com,bin	Read to run machine language program
Object	obj,o	Compiled,machine language not linked
Source code	C,java,pas,asm,a	Source code in various languages
Batch	bat,sh	Commands to the command interpreter
Text	txt,doc	Textual data,documents
Word processor	Wp,tex,rrf,doc	Various word processor formats
Archive	arc,zip,tar	Related files grouped into one file compressed for storage
Multimedia	mpeg,mov,rm	File containing audio or a/v information

FILE DIRECTORIES:
Collection of files is a file directory. The directory contains information about the files, including attributes, location and ownership. Much of this information, especially that is concerned with storage, is managed by the operating system. The directory is itself a file, accessible by various file management routines.

Information contained in a device directory are:

- Name
- Type
- Address
- Current length
- Maximum length
- Date last accessed
- Date last updated
- Owner id
- Protection information

Operation performed on directory are:

- Search for a file
- Create a file
- Delete a file
- List a directory
- Rename a file
- Traverse the file system

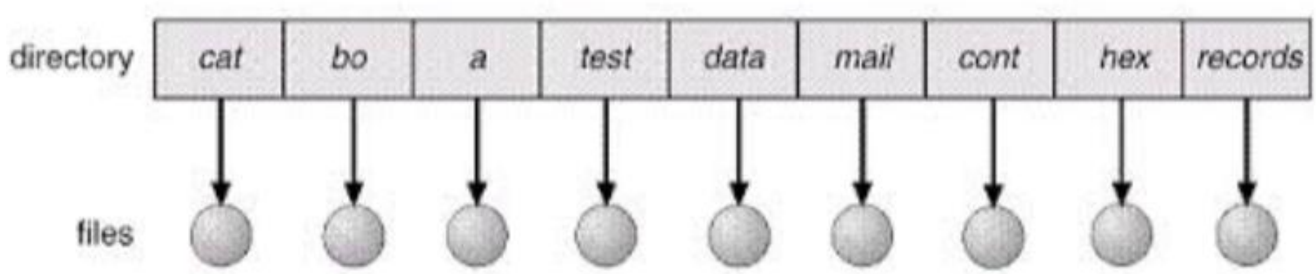
Advantages of maintaining directories are:

- **Efficiency:** A file can be located more quickly.
- **Naming:** It becomes convenient for users as two users can have same name for different files or may have different name for same file.
- **Grouping:** Logical grouping of files can be done by properties e.g. all java programs, all games etc.

SINGLE-LEVEL DIRECTORY

In this a single directory is maintained for all the users.

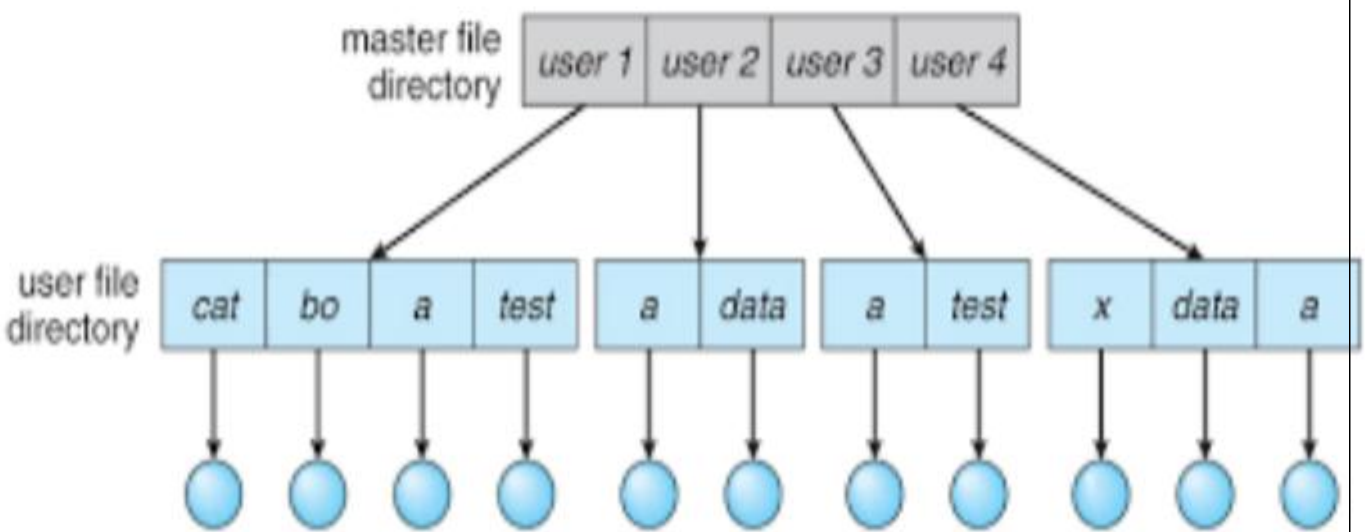
- **Naming problem:** Users cannot have same name for two files.
- **Grouping problem:** Users cannot group files according to their need.



TWO-LEVEL DIRECTORY

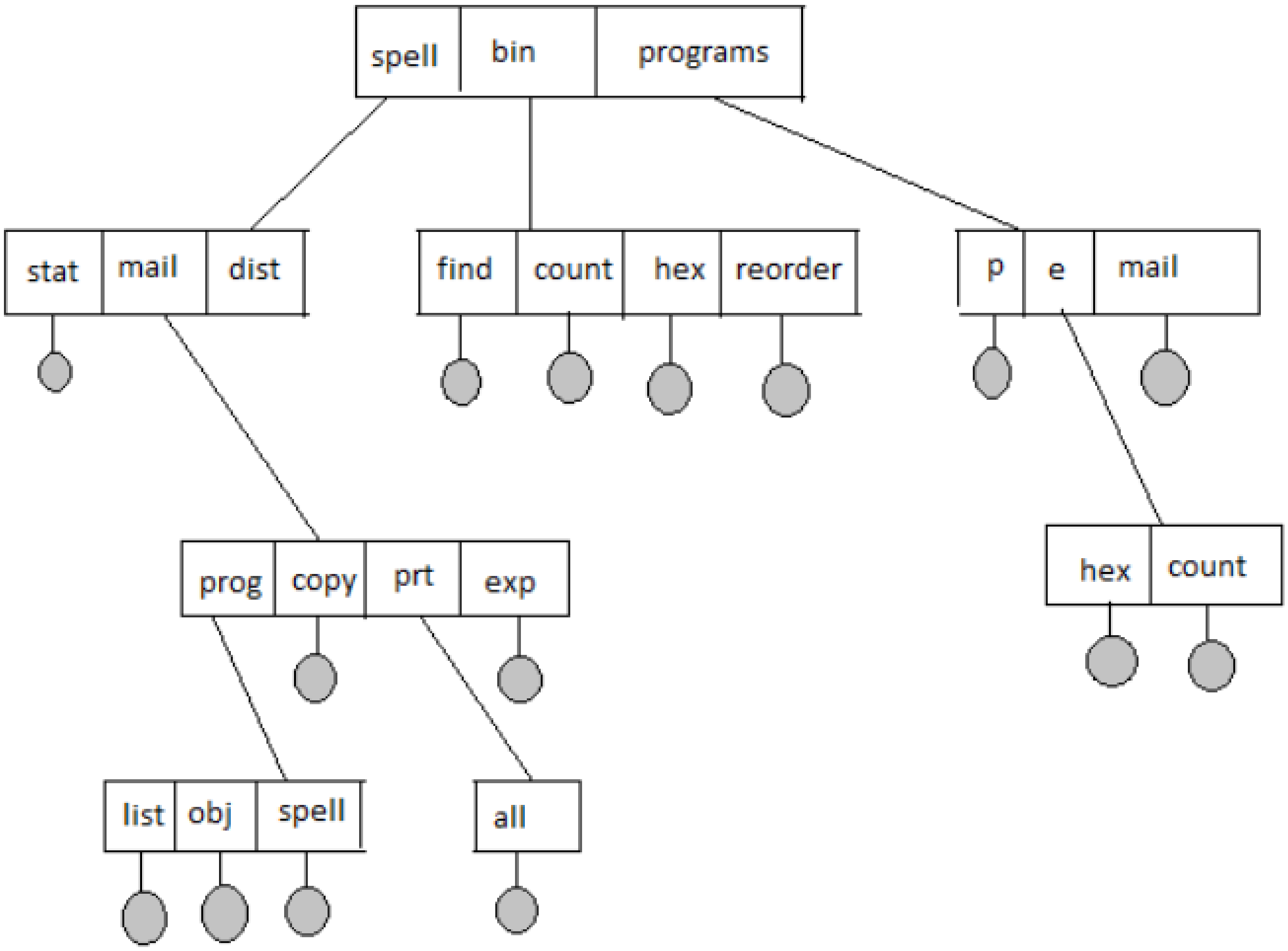
In this separate directories for each user is maintained.

- Path name:Due to two levels there is a path name for every file to locate that file.
- Now,we can have same file name for different user.
- Searching is efficient in this method.



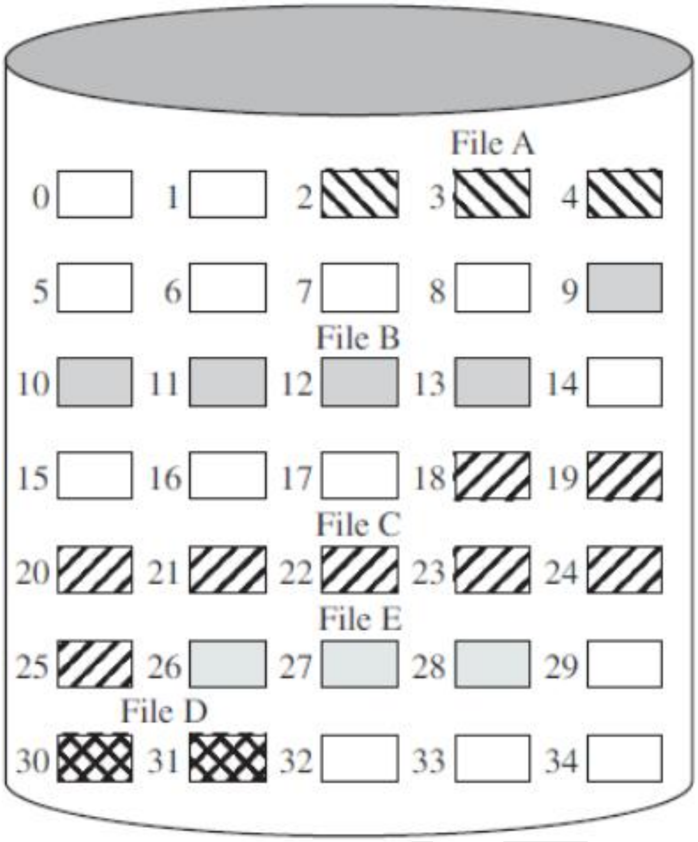
TREE-STRUCTURED DIRECTORY :

Directory is maintained in the form of a tree. Searching is efficient and also there is grouping capability. We have absolute or relative path name for a file.



FILE ALLOCATION METHODS

1. Continuous Allocation: A single continuous set of blocks is allocated to a file at the time of file creation. Thus, this is a pre-allocation strategy, using variable size portions. The file allocation table needs just a single entry for each file, showing the starting block and the length of the file. This method is best from the point of view of the individual sequential file. Multiple blocks can be read in at a time to improve I/O performance for sequential processing. It is also easy to retrieve a single block. For example, if a file starts at block b, and the ith block of the file is wanted, its location on secondary storage is simply b+i-1.



File allocation table

File name	Start block	Length
File A	2	3
File B	9	5
File C	18	8
File D	30	2
File E	26	3

Disadvantage

- External fragmentation will occur, making it difficult to find contiguous blocks of space of sufficient length. Compaction algorithm will be necessary to free up additional space on disk.
- Also, with pre-allocation, it is necessary to declare the size of the file at the time of creation.

2. Linked Allocation(Non-contiguous allocation) : Allocation is on an individual block basis. Each block contains a pointer to the next block in the chain. Again the file table needs just a single entry for each file, showing the starting block and the length of the file. Although pre-allocation is possible, it is more common simply to allocate blocks as needed. Any free block can be added to the chain. The blocks need not be continuous. Increase in file size is always possible if free disk block is available. There is no external fragmentation because only one block at a time is needed but there can be internal fragmentation but it exists only in the last disk block of file.

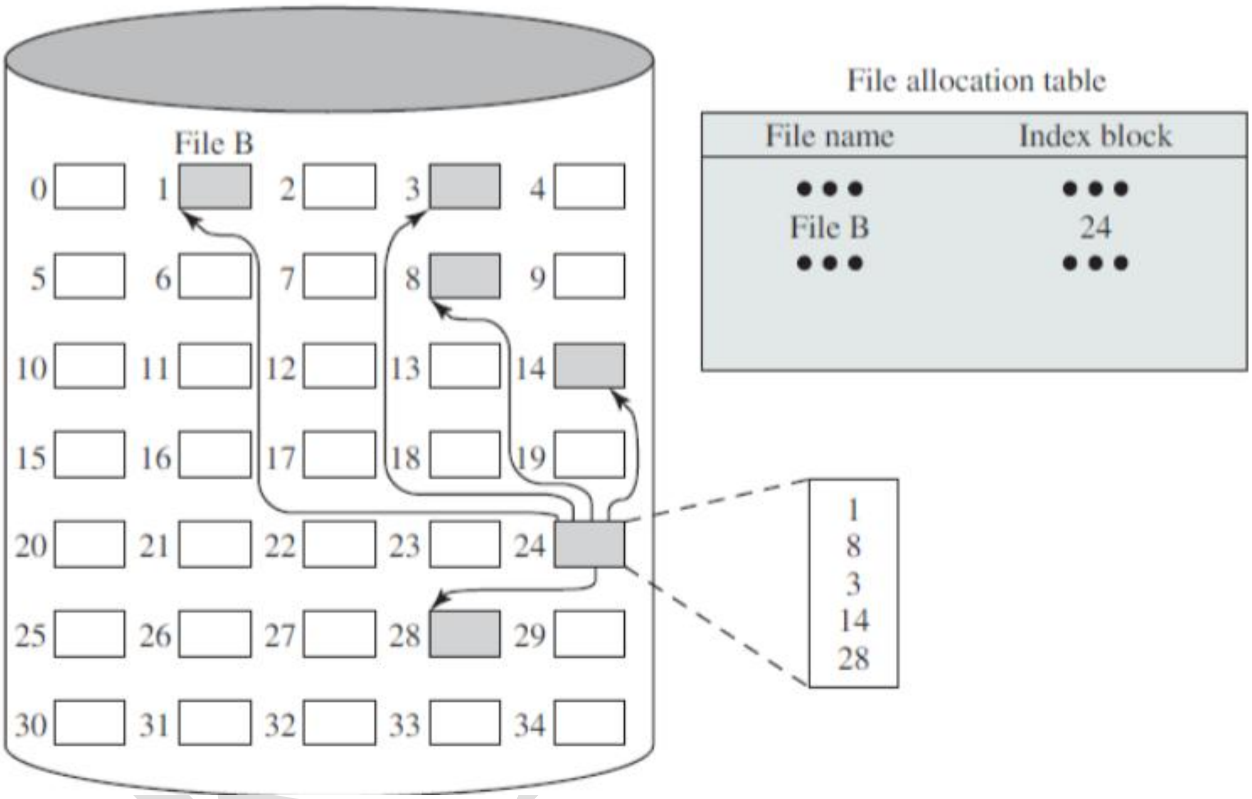
Disadvantage:

- Internal fragmentation exists in last disk block of file.
- There is an overhead of maintaining the pointer in every disk block.
- If the pointer of any disk block is lost, the file will be truncated.

- It supports only the sequencial access of files.

3. Indexed Allocation:

It addresses many of the problems of contiguous and chained allocation. In this case, the file allocation table contains a separate one-level index for each file: The index has one entry for each block allocated to the file. Allocation may be on the basis of fixed-size blocks or variable-sized blocks. Allocation by blocks eliminates external fragmentation, whereas allocation by variable-size blocks improves locality. This allocation technique supports both sequential and direct access to the file and thus is the most popular form of file allocation.



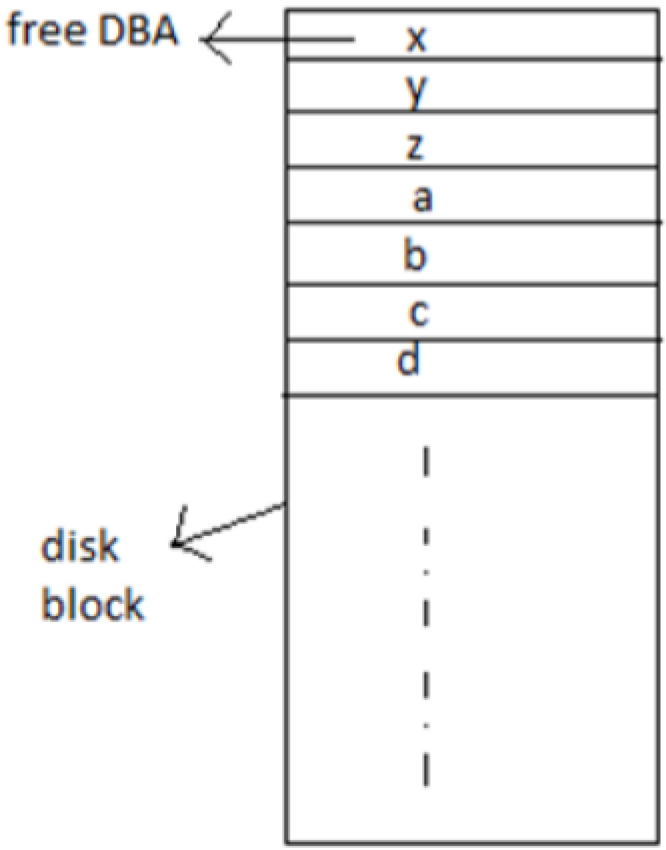
Disk Free Space Management

Just as the space that is allocated to files must be managed ,so the space that is not currently allocated to any file must be managed. To perform any of the file allocation techniques,it is necessary to know what blocks on the disk are available. Thus we need a disk allocation table in addition to a file allocation table.The following are the approaches used for free space management.

1. **Bit Tables** : This method uses a vector containing one bit for each block on the disk. Each entry for a 0 corresponds to a free block and each 1 corresponds to a block in use.
   For example: 00011010111100110001

   In this vector every bit correspond to a particular vector and 0 implies that, that particular block is free and 1 implies that the block is already occupied. A bit table has the advantage that it is relatively easy to find one or a contiguous group of free blocks. Thus, a bit table works well with any of the file allocation methods. Another advantage is that it is as small as possible.

2. **Free Block List** : In this method, each block is assigned a number sequentially and the list of the numbers of all free blocks is maintained in a reserved block of the disk.



Disk Scheduling Algorithms
2.4

Disk scheduling is is done by operating systems to schedule I/O requests arriving for disk. Disk scheduling is also known as I/O scheduling.

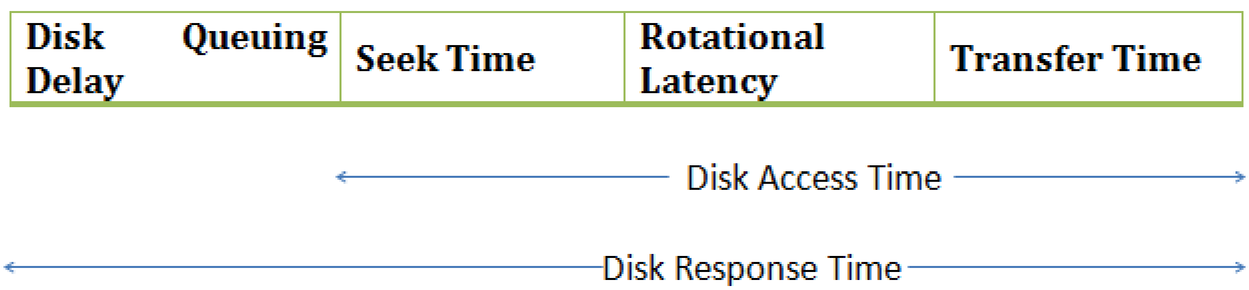
Disk scheduling is important because:

- Multiple I/O requests may arrive by different processes and only one I/O request can be served at a time by disk controller. Thus other I/O requests need to wait in waiting queue and need to be scheduled.
- Two or more request may be far from each other so can result in greater disk arm movement.
- Hard drives are one of the slowest parts of computer system and thus need to be accessed in an efficient manner.

There are many Disk Scheduling Algorithms but before discussing them let's have a quick look at some of the important terms:

- **Seek Time:** Seek time is the time taken to locate the disk arm to a specified track where the data is to be read or write. So the disk scheduling algorithm that gives minimum average seek time is better.
- **Rotational Latency:** Rotational Latency is the time taken by the desired sector of disk to rotate into a position so that it can access the read/write heads. So the disk scheduling algorithm that gives minimum rotational latency is better.
- **Transfer Time:** Transfer time is the time to transfer the data. It depends on the rotating speed of the disk and number of bytes to be transferred.
- **Disk Access Time:** Disk Access Time is:

Disk Access Time = Seek Time +
Rotational Latency +
Transfer Time



- **Disk Response Time:** Response Time is the average of time spent by a request waiting to perform its I/O operation. *Average Response time* is the response time of the all requests. *Variance Response Time* is measure of how individual request are serviced with respect to average response time. So the disk scheduling algorithm that gives minimum variance response time is better.

Disk Scheduling Algorithms

1. **FCFS:** FCFS is the simplest of all the Disk Scheduling Algorithms. In FCFS, the requests are addressed in the order they arrive in the disk queue.

Advantages:

- Every request gets a fair chance
- No indefinite postponement

Disadvantages:

- Does not try to optimize seek time

- May not provide the best possible service

2. **SSTF:** In SSTF (Shortest Seek Time First), requests having shortest seek time are executed first. So, the seek time of every request is calculated in advance in queue and then they are scheduled according to their calculated seek time. As a result, the request near the disk arm will get executed first. SSTF is certainly an improvement over FCFS as it decreases the average response time and increases the throughput of system.

Advantages:

- Average Response Time decreases
- Throughput increases

Disadvantages:

- Overhead to calculate seek time in advance
- Can cause Starvation for a request if it has higher seek time as compared to incoming requests
- High variance of response time as SSTF favours only some requests

3. **SCAN:** In SCAN algorithm the disk arm moves into a particular direction and services the requests coming in its path and after reaching the end of disk, it reverses its direction and again services the request arriving in its path. So, this algorithm works like an elevator and hence also known as **elevator algorithm.** As a result, the requests at the midrange are serviced more and those arriving behind the disk arm will have to wait.

Advantages:

- High throughput
- Low variance of response time
- Average response time

Disadvantages:

- Long waiting time for requests for locations just visited by disk arm

4. **CSCAN**: In SCAN algorithm, the disk arm again scans the path that has been scanned, after reversing its direction. So, it may be possible that too many requests are waiting at the other end or there may be zero or few requests pending at the scanned area.

These situations are avoided in *CSAN* algorithm in which the disk arm instead of reversing its direction goes to the other end of the disk and starts servicing the requests from there. So, the disk arm moves in a circular fashion and this algorithm is also similar to SCAN algorithm and hence it is known as C-SCAN (Circular SCAN).

Advantages:

- Provides more uniform wait time compared to SCAN

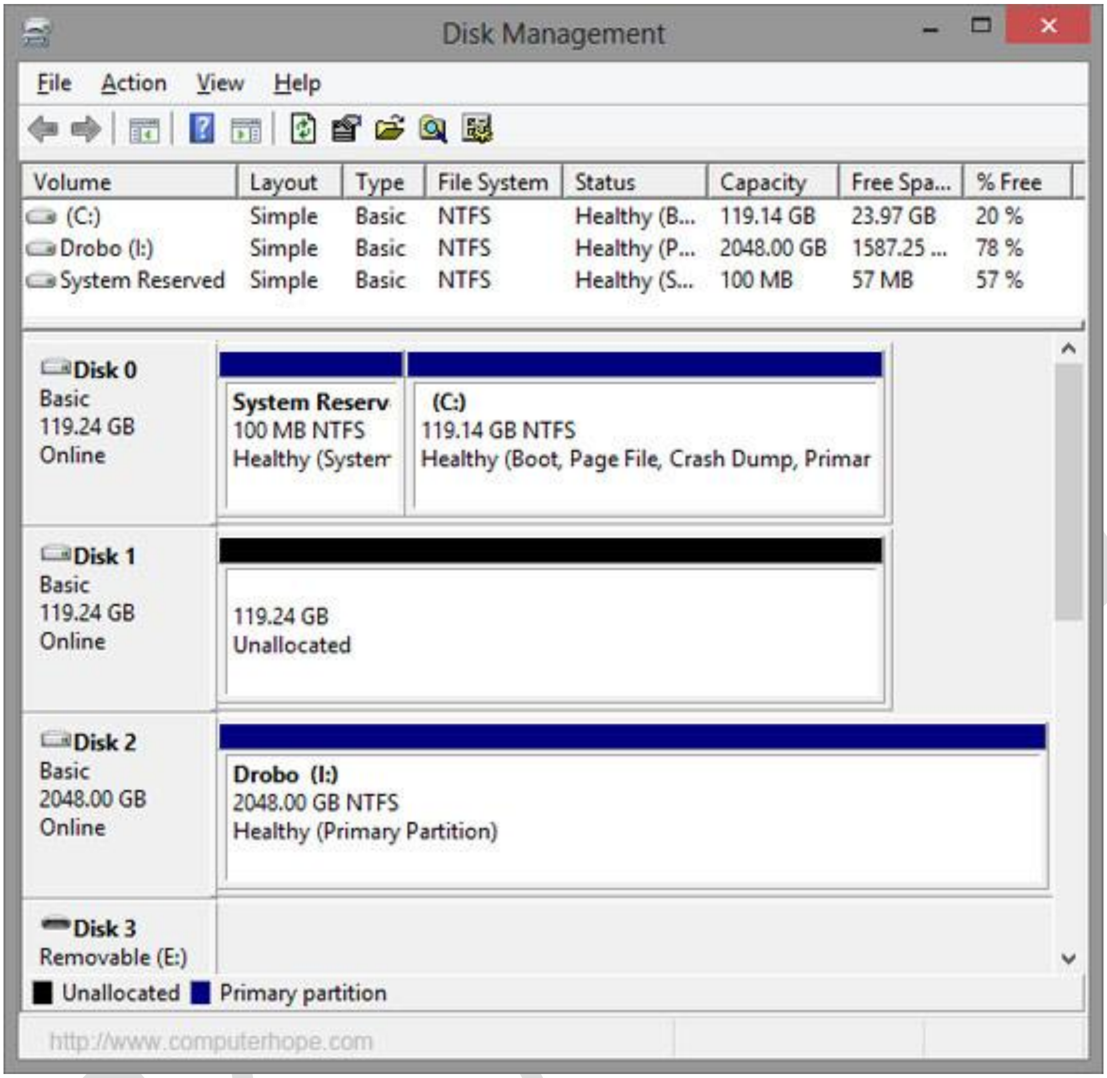
5. **LOOK:** It is similar to the SCAN disk scheduling algorithm except the difference that the disk arm in spite of going to the end of the disk goes only to the last request to be serviced in front of the head and then reverses its direction from there only. Thus it prevents the extra delay which occurred due to unnecessary traversal to the end of the disk.

6. **CLOOK:** As LOOK is similar to SCAN algorithm, in similar way, CLOOK is similar to CSCAN disk scheduling algorithm. In CLOOK, the disk arm inspite of going to the end goes only to the last request to be serviced in front of the head and then from there goes to the other end's last request. Thus, it also prevents the extra delay which occurred due to unnecessary traversal to the end of the disk.

Disk Management

Updated: 11/10/2017 by Computer Hope

Disk Management is a Microsoft Windows utility first introduced in Windows XP as a replacement for the fdisk command. It enables users to view and manage the disk drives installed in their computer and the partitions associated with those drives. As can be seen in the picture below, each drive is displayed followed by the layout, type, file system, status, capacity, free space, % free, and fault tolerance.



How to open Windows Disk Management

1. Click the Start button and access the Run option. You can also press **Windows key + R** on the keyboard to open the Run option.
2. Type **diskmgmt.msc** and press Enter.

Tip: In Windows 8, you can type "diskmgmt.msc" directly on the Start screen to access Disk Management.

or

1. Open the Control Panel.
2. Double-click on **Administrative Tools** if in Classic View or click **Performance and Maintenance** and then **Administrative Tools** if in Category View. Note: If you do not have admin rights to the computer this will not be available.
3. Once in the Administrative Tools window double-click **Computer Management** and then click **Disk Management** under the Storage section.

POSSIBLE QUESTIONS

PART –B

(Each Question carries 2 Marks)

1.What is file structure?

2.Define: Directory file

3. What is file Type?

4.List the file accessing mechanisms.

5.What is disk scheduling?

6.What is seek time?

7.Define: Rotational latency

8.What is shortest seek time first?

9.List the allocation methods

10. What is disk management?

PART –C

(Each Question carries 6 Marks)

1.Explain the various file accessing methods in detail.

2.Write short notes on file sharing.

3.Write short notes on disk structure.

4.Write short notes on disk scheduling.

5.What is disk management in detail?

6.Write short notes on free space management.

7.How do you achieve efficiency and performance of file system?

8.Write short notes on disk performance optimization.

9.Write a SCAN algorithm and explain.

10. Write a CSCAN algorithm and explain.

Linux-The Operating System: Linux History, Linux features, Linux distributions, Linux's relationship to Unix, Overview of Linux Architecture, Installation, Start up scripts, system process (an overview), Linux Security, The Ext2 and Ext3 File Systems: General characteristics of the Ext3 File System, File permissions, User Management: Types of users, the powers of Root, Managing users (adding and deleting) : using the command line and GUI Tools.
Resource Management in Linux: File and Directory management, system calls for files process management, Signals, IPC:Pipes, FIFOs, System V IPC, Message Queues, System calls for processes, Memory Management, Library and System calls for Memory.

Linux:

What Is Linux?

In the simple language Linux is an operating system (OS). We all are familiar with other operating systems like Microsoft windows, Apple Mac OS, iOS, Google android, etc, just like them linux is also an operating system.

An operating system is a software that enables communication between computer hardware and software. It conveys input to get processed by the processor and brings output to the hardware to display it. This is the basic function of an operating system. Although, it performs many other important tasks, let's not talk about that.

Linux is around us since mid 90s. It can be used from wristwatches to supercomputers. It is everywhere in our phones, laptops, PCs, cars and even in refrigerators.It is very much famous among the developers and normal computer users.

Structure Of Linux Operating System

An operating system is a collection of software, each designed for a specific function.

Linux OS has following components:

1) Kernel

kernel is the core of the operating system. It establishes communication between devices and software. Moreover, it manages the system resources. Basically it has four responsibilities:

- **device management:** A system has many devices connected to it like CPU, memory device, sound cards, graphic cards, etc. A kernel stores all the data related to all the devices in device driver (without this kernel won't be able to control the devices). Thus kernel knows what a device can do and how to manipulate it to bring out the best performance. It also manages communication between all the devices. Kernel has certain rules that has to be followed by all the devices.

- **Memory management:** Another function that kernel has to manage is the memory management. Kernel keeps a track of used and unused memory and make sure that processes shouldn't manipulate data of each other using virtual memory address.
- **Process management:** In process management kernel assign enough time and gives priorities to processes before handling CPU to other process. It also deals with security and ownership information.
- **Handling system calls:** Handling system calls means a programmer can write a query or ask the kernel to perform a task.

2) System Libraries

System libraries are special programs that helps in accessing the kernel's features. A kernel has to be triggered to perform a task and this triggering is done by the applications. But applications must know how to place a system call because each kernel has a different set of system calls. Programmers have developed standard library of procedures to communicate with kernel. Each operating system supports these standards and then these are transferred to system calls for that operating system.

Most well known system library for Linux is glibc (GNU C library).

3) System Tools

Linux OS has a set of utility tools which are usually simple commands. It is a software which GNU project has written and publish under their open source license so that software is freely available to everyone.

With the help of commands you can access your files, edit and manipulate data in your directories or files, change location of files or anything.

4) Development Tools

With the above three components your OS is running and working. But to update your system you have additional tools and libraries. These additional tools and libraries are written by the programmers and are called tool chain. A tool chain is a vital development tool used by the developers to produce a working application.

5) End User Tools

These end tools make a system unique for a user. End tools are not required for the operating system but are necessary for a user.

Some examples of end tools are graphic design tools, office suites, browsers, multimedia players, etc.

Open Source Operating System

Most OS come in a compiled format means the main source code has run through a program called compiler that translates the source code into a language which is known to the computer.

Modifying this compiled code is really a tough job.

On the other hand, open source is completely different. The source code is included with the compiled version and allows modification by anyone having some knowledge. It gives us freedom to run the program, freedom to change the code according to our use, freedom to redistribute its copies and freedom to distribute copies which are modified by us.

In short, Linux is an operating system that is "for the people, by the people".

Linux History

Evolution of Computer

In earlier days, computers were as big as houses or parks. So you can imagine how difficult it was to operate them. Moreover, every computer has a different operating system which made it completely worse to operate on them. Every software was designed for a specific purpose and was unable to operate on other computer. It was extremely costly and normal people neither can afford it nor can understand it.

Evolution of Unix

In 1969, a team of developers of Bell Labs started a project to make a common software for all the computers and named it as 'Unix'. It was simple and elegant, used 'C' language instead of assembly language and its code was recyclable. As it was recyclable, a part of its code now commonly called 'kernel' was used to develop the operating system and other functions and could be used on different systems. Also its source code was open source.

Initially, Unix was only found in large organizations like government, university, or larger financial corporations with mainframes and minicomputers (PC is a microcomputer).

Unix Expansion

In eighties, many organizations like IBM, HP and dozen other companies started creating their own Unix. It result in a mess of Unix dialects. Then in 1983, Richard Stallman developed GNU project with the goal to make it freely available Unix like operating system and to be used by everyone. But his project failed in gaining popularity. Many other Unix like operating system came into existence but none of them was able to gain popularity.

Evolution of Linux

In 1991, Linus Torvalds a student at the university of Helsinki, Finland, thought to have a freely available academic version of Unix started writing its own code. Later this project became the Linux kernel. He wrote this program specially for his own PC as he wanted to use Unix 386 Intel computer but couldn't afford it. He did it on MINIX using GNU C compiler. GNU C compiler is still the main choice to compile Linux code but other compilers are also used like Intel C compiler.

He started it just for fun but ended up with such a large project. Firstly he wanted to name it as 'Freax' but later it became 'Linux'.

He published the Linux kernel under his own license and was restricted to use as commercially. Linux uses most of its tools from GNU software and are under GNU copyright. In 1992, he released the kernel under GNU General Public License.

Linux Today

Today, supercomputers, smart phones, desktop, web servers, tablet, laptops and home appliances like washing machines, DVD players, routers, modems, cars, refrigerators, etc use Linux OS.

Linux Features

- **Multiuser capability:** Multiple users can access the same system resources like memory, hard disk, etc. But they have to use different terminals to operate.
- **Multitasking:** More than one function can be performed simultaneously by dividing the CPU time intelligently.
- **Portability:** Portability doesn't mean it is smaller in file size or can be carried in pen drives or memory cards. It means that it support different types of hardware.
- **Security:** It provides security in three ways namely authenticating (by assigning password and login ID), authorization (by assigning permission to read, write and execute) and encryption (converts file into an unreadable format).
- **Live CD/USB:** Almost all Linux distros provide live CD/USB so that users can run/try it without installing it.
- **Graphical User Interface (X Window system):** Linux is command line based OS but it can be converted to GUI based by installing packages.
- **Support's customized keyboard:** As it is used worldwide, hence supports different languages keyboards.
- **Application support:** It has its own software repository from where users can download and install many applications.
- **File System:** Provides hierarchical file system in which files and directories are arranged.
- **Open Source:** Linux code is freely available to all and is a community based development project.

Why Use Linux

Linux is completely different from other operating systems in many ways.

- It is an open source OS which gives a great advantage to the programmers as they can design their own custom operating systems.
- It gives you a lot of option of programs having some different features so you can choose according to your need.
- A global development community look at different ways to enhance its security, hence it is highly secured and robust so you don't need an anti virus to scan it regularly. Companies like Google, Amazon and Facebook use linux in order to protect their servers as it is highly reliable and stable.
- Above all you don't have to pay for software and server licensing to install Linux, its absolutely free and you can install it on as many computers as you want.
- Its completely trouble free operating system and don't have an issue with viruses, malware and slowing down your computer.

Linux Distributions (Distros)

Other operating systems like Microsoft combine each bit of codes internally and release it as a single package. You have to choose from one of the version they offer.

But Linux is different from them. Different parts of Linux are developed by different organizations.

Different parts include kernel, shell utilities, X server, system environment, graphical programs, etc. If you want you can access the codes of all these parts and assemble them yourself. But its not an easy task seeking a lot of time and all the parts has to be assembled correctly in order to work properly.

From here on distribution (also called as distros) comes into the picture. They assemble all these parts for us and give us a compiled operating system of Linux to install and use.

Linux Distributions List

There are on an average six hundred Linux distributors providing different features. Here, we'll discuss about some of the popular Linux distros today.

1) Ubuntu

It came into existence in 2004 by Canonical and quickly became popular. Canonical wants Ubuntu to be used as easy graphical Linux desktop without the use of command line. It is the most well known Linux distribution. Ubuntu is a next version of Debian and easy to use for newbies. It comes with a lots of pre-installed apps and easy to use repositories libraries.

Earlier, Ubuntu uses GNOME2 desktop environment but now it has developed its own unity desktop environment. It releases every six months and currently working to expand to run on tablets and smartphones.

2) Linux Mint

Mint is based on Ubuntu and uses its repository software so some packages are common in both.

Earlier it was an alternative of Ubuntu because media codecs and proprietary software are included in mint but was absent in Ubuntu. But now it has its own popularity and it uses cinnamon and mate desktop instead of Ubuntu's unity desktop environment.

3) Debian

Debian has its existence since 1993 and releases its versions much slowly then Ubuntu and mint.

This makes it one of the most stable Linux distributor.

Ubuntu is based on Debian and was founded to improve the core bits of Debian more quickly and make it more user friendly. Every release name of Debian is based on the name of the movie Toy Story.

4) Red Hat Enterprise / CentOS

Red hat is a commercial Linux distributor. There products are red hat enterprise Linux (RHEL) and Fedora which are freely available. RHEL is well tested before release and supported till seven years after the release, whereas, fedora provides faster update and without any support.

Red hat uses trademark law to prevent their software from being redistributed. CentOS is a community project that uses red hat enterprise Linux code but removes all its trademark and make it freely available. In other words, it is a free version of RHEL and provide a stable platform for a long time.

5) Fedora

It is a project that mainly focuses on free software and provides latest version of software. It doesn't make its own desktop environment but used 'upstream' software. By default it has GNOME3 desktop environment. It is less stable but provides the latest stuff.

Choosing a Linux Distro

Distribution	Why To Use

UBuntu	It works like Mac OS and easy to use.
Linux mint	It works like windows and should be use by new comers.
Debian	It provides stability but not recommended to a new user.
Fedora	If you want to use red hat and latest software.
Red hat enterprise	To be used commercially.
CentOS	If you want to use red hat but without its trademark.
OpenSUSE	It works same as Fedora but slightly older and more stable.
Arch Linux	It is not for the beginners because every package has to be installed by yourself.

Unix Vs Linux

Today Linux is in great demand. You can see the use of Linux everywhere. It's dominating on our servers, desktop, smartphones and even used in some electrical devices like refrigerators.

Some people think Unix and Linux as synonyms, but that's not true. Many operating systems were developed to be like Unix but none of them got the popularity as Linux. Linux is the clone of Unix. It has several features similar to Unix, still have some key differences. Before Linux and Windows, computer world was dominated by Unix. Unix is a copyrighted name and IBM AIX, HP-UX and Sun Solaris are only Unix operating system remained till date.

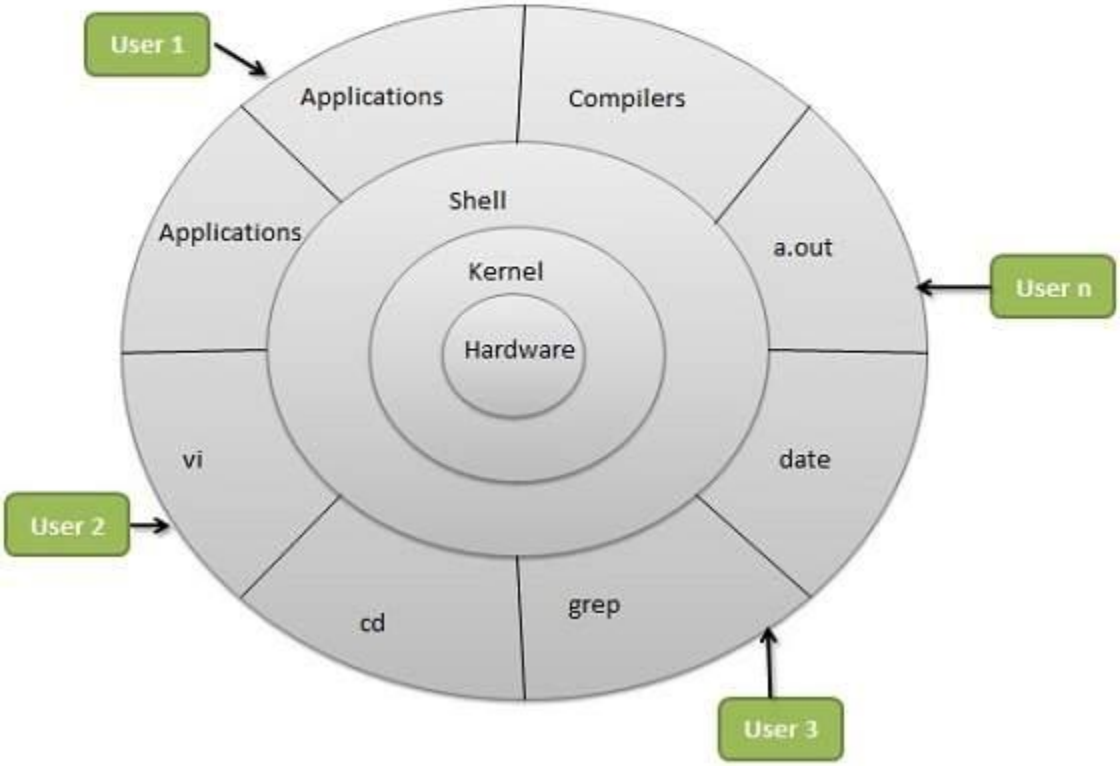
Difference between Linux and Unix

Comparison	Linux	Unix
Definition	It is an open-source operating system which is *freely available to everyone*.	It is an operating system which *can be only used by its copyrighters*.
Examples	It has different distros like Ubuntu, Redhat, Fedora, etc	IBM AIX, HP-UX and Sun Solaris.
Users	Nowadays, Linux is in great demand. Anyone can use Linux whether a home user, developer or a student.	It was developed mainly for servers, workstations and mainframes.
Usage	Linux is used everywhere from servers, PC, smartphones, tablets to mainframes and supercomputers.	It is used in servers, workstations and PCs.
Cost	Linux is freely distributed,downloaded, and distributed through magazines also. And priced distros of Linux are	Unix copyright vendors decide different costs for their respective Unix Operating

	also cheaper than Windows.	systems.
Development	As it is open source, it is developed by sharing and collaboration of codes by world-wide developers.	Unix was developed by AT&T Labs, various commercial vendors and non-profit organizations.
Manufacturer	Linux kernel is developed by the community of developers from different parts of the world. Although the father of Linux, Linus Torvalds oversees things.	Unix has three distributions IBM AIX, HP-UX and Sun Solaris. Apple also uses Unix to make OSX operating system.
GUI	Linux is command based but some distros provide GUI based Linux. Gnome and KDE are mostly used GUI.	Initially it was command based OS, but later Common Desktop Environment was created. Most Unix distributions use Gnome.
Interface	The default interface is BASH (Bourne Again SHell). But some distros have developed their own interfaces.	It originally used Bourne shell. But is also compatible with other GUIs.
File system support	Linux supports more file system than Unix.	It also supports file system but lesser than Linux.
Coding	Linux is a Unix clone,behaves like Unix but doesn't contain its code.	Unix contain a completely different coding developed by AT&T Labs.
Operating system	Linux is just the kernel.	Unix is a complete package of Operating system.
Security	It provides higher security. Linux has about 60-100 viruses listed till date.	Unix is also highly secured. It has about 85-120 viruses listed till date
Error detection and solution	As Linux is open-source,whenever a user post any kind of threat, developers from all over the world start working on it. And hence, it provides faster solution.	In Unix, users have to wait for some time for the problem to be resolved.

Architecture

The following illustration shows the architecture of a Linux system −



The architecture of a Linux System consists of the following layers −

- **Hardware layer** − Hardware consists of all peripheral devices (RAM/ HDD/ CPU etc).
- **Kernel** − It is the core component of Operating System, interacts directly with hardware, provides low level services to upper layer components.
- **Shell** − An interface to kernel, hiding complexity of kernel's functions from users. The shell takes commands from the user and executes kernel's functions.
- **Utilities** − Utility programs that provide the user most of the functionalities of an operating systems.

Introduction to Linux security principles

Introduction

Security should be one of the foremost thoughts at all stages of setting up your Linux computer. To implement a good security policy on a machine requires a good knowledge of the fundamentals of Linux as well as some of the applications and protocols that are used.

Security of Linux is a massive subject and there are many complete books on the subject. I couldn't put everything in this one tutorial, but this does give a basic introduction to security and how the techniques, and tools can be used to provide additional security on a Linux computer. Hopefully this will provide sufficient information to be able to investigate other sources of information.

Why do we need security?

Although Linux users are must less prone to viruses than some other major operating systems, there are still many security issues facing Linux users and administrators.

One of the most important steps in any task is to identify why you are doing it. Rather than just saying we need to make a system secure you need to consider what is meant by secure, what risks there are associated with any data that's available, what impact your security measures will have on your users. Without first considering any of these factors how else will you know if you've met your goal of making a system secure.

Security requirements

After establishing why security is to be implemented you should consider the aspects of security that are required. The main security requirements are:

Authorisation - Only allow those that need access to the data
Authenticity - Verifying they are who they say they are
Privacy / Confidentiality - Ensure personal information is not being compromised
Integrity - Ensuring that the data has not been tampered with
Non-repudiation - Confirmation that data is received. The ability to prove it in court
Availability - Ensure that the system can perform it's required function

Imposed requirements

Some security requirements are not ones that are directly under your control but are instead imposed upon you. These may be legal requirements (e.g. Data Protection Act 1998), compliance with standards (e.g. ISO 7984-2 International Standards Organisation Security Standard), or corporate policy. If you handle credit card transactions then you may be required to comply with minimum security standards as described by the Payment Card Industry (PCI).

Some of these standards are very vague (e.g. the Data Protection Act just specifies that appropriate security should be in place) whereas some may be more specific (e.g. a corporate policy may insist on a minimum length of passwords etc.).

Knowing the enemy

Before being able to effectively protect a computer system you need to know who it is that is trying to attack your systems and what they are trying to do. I have shown some examples by answering a few questions about those who could potentially attack a computer system.

1. Who wants to?
2. Why are they doing this?
3. What do they try and achieve?
4. How do they do it?

Hackers, crackers and phreakers

These words are commonly used when referring to security attacks, however the meanings are often misinterpreted or understood. I have taken these in order of how easy the term is to explain so as to avoid confusing these together. Note that other people may have different meanings when they use these terms.

Phreakers - Also known as Phone Phreakers, this term originates from what could be considered to be the earliest form of attacks against electronic systems. It's earliest for was to bypass the systems used in telephone systems allowing free or reduced price international phone calls. One of the earliest forms of this was when the American pay phone system used a certain frequency signal to indicate that a coin had been placed in the phone. It was discovered that the frequency of the signal was 2600 Hz, which was also the same frequency emitted from a toy whistle distributed with a popular make of cereals. By blowing the whistle into the phone when a request was made for payment the Phreaker could fool the operating into thinking that money had been deposited in the pay phone.

Crackers - These are people that gain unauthorised access to a computer. When people refer to hackers breaking into a computer then they are really referring to crackers.

Hackers - Using the traditional meaning of the word Hacker is not meant to imply any kind of illegal or immoral activities. The true meaning is of a computer enthusiast that understands the inner workings of a system and uses that knowledge to "hack" together programs etc. to perform a function. This was different to the traditional techniques or programming that are designed to follow a set structure and procedure to produce a finished piece of software. Due to incorrect use, including by the press, the word hacker has now come to take on two meanings. One is it's original meaning and the other is that of anyone who tries to penetrate a computer (crackers) or those who cause intentional disruption or damage (none-physical) to computer systems.

Throughout this tutorial I will normally refer to the perpetrator as an attacker, regardless of which of these categories she comes under, however where I do refer to a hacker I will normally mean the newer of these meanings.

Whilst some may object to my use of the word hacker, my justification is to turn to the definition held in the Oxford English dictionary which describes the popular use of the language and is considered a definitive guide to the English language:
"Hacker - computer enthusiast, esp. one gaining unauthorised access to files"
The Oxford Popular Dictionary, Parragon, 1995

The stereotypes - why be a hacker?

By understanding the reasons for the attacks gives a basis for what protection can be used to protect the data. I have therefore taken a few examples of reasons for hackers. This includes the stereotypical examples and some that you may not necessarily think about. This is by no means complete, it does however highlight that there are different reasons that someone would want to attack your system.

Just for fun - Typically someone in further or higher education that uses the college or universities computer facilities to attack another computer over the Internet. Whilst there are indeed a number of attackers that match this description it is important to recognise that these are not the only type of hackers. This person will typically have limited resources and normally does it, just for fun; or to prove their intelligence etc. However they may be part of a larger group united using the Internet. Whilst many do not intend to commit malicious damage they may discredit your company name, they may cause accidental damage, and may open the door for others.

Commercial espionage / sabotage - Whilst espionage normally congers up the image of James Bond fighting a host of bad guys the reality is much less dramatic. There is potentially a risk from competitors wanting to gain a competitive edge. For example if you are bidding for a contract and your competitor is able to find out details of your bid, they could easily undercut you and win the contract. Alternatively by putting your web page out of action, customers could be encouraged to try the competition.
This kind of attacker normally has a lot of resources, both financial and in man power, at it's disposal and

has very specific targets. If your organisation is involved in military contracts there may be a real Ernst Stavro Blofeld trying to steal the technology to take over the world.

Fighting a cause - Other groups that may wish to attack your company are those that are fighting for a cause or defending a belief. Whilst there are a number of obvious extremist groups such as terrorists or the extremist animal rights groups this could equally apply to less controversial areas where someone has a different opinion.

Disgruntled employees - So far I have mentioned attackers external to the organisation, however it is sometimes the case that the greater risk lies from employees within the organisation. These could already have authorised access to a computer, and already be inside the firewall. They could then use that access against the organisation and exploit other holes in the system. Whilst these people can have different motives one of the most obvious is for someone that has been fired, disciplined or who is not satisfied with their current standing in the organisation. Defending against the internal employee can be more challenging as methods need to be found to limit access without preventing others for performing their job. To tighten up security to the point where employees cannot do their job properly is an indirect Denial of Service.

Unintentional user error - Whilst normal users may not be trying to cause any damage to the system it's possible that they could cause some accidental damage to data. By limiting a users access user errors can be contained to a reasonable extent. This could be in the form of a programming error as well as incorrectly typing instructions into a program.

Linux File Systems: Ext2 vs Ext3 vs Ext4

by Ramesh Natarajan on May 16, 2011

Tweet

ext2, ext3 and ext4 are all filesystems created for Linux. This article explains the following:

- High level difference between these filesystems.
- How to create these filesystems.
- How to convert from one filesystem type to another.

Ext2

- Ext2 stands for second extended file system.
- It was introduced in 1993. Developed by Rémy Card.
- This was developed to overcome the limitation of the original ext file system.
- Ext2 does not have journaling feature.
- On flash drives, usb drives, ext2 is recommended, as it doesn’t need to do the over head of journaling.
- Maximum individual file size can be from 16 GB to 2 TB
- Overall ext2 file system size can be from 2 TB to 32 TB

Ext3

- Ext3 stands for third extended file system.
- It was introduced in 2001. Developed by Stephen Tweedie.
- Starting from Linux Kernel 2.4.15 ext3 was available.

- The main benefit of ext3 is that it allows journaling.
- Journaling has a dedicated area in the file system, where all the changes are tracked. When the system crashes, the possibility of file system corruption is less because of journaling.
- Maximum individual file size can be from 16 GB to 2 TB
- Overall ext3 file system size can be from 2 TB to 32 TB
- There are three types of journaling available in ext3 file system.
    - Journal – Metadata and content are saved in the journal.
    - Ordered – Only metadata is saved in the journal. Metadata are journaled only after writing the content to disk. This is the default.
    - Writeback – Only metadata is saved in the journal. Metadata might be journaled either before or after the content is written to the disk.
- You can convert a ext2 file system to ext3 file system directly (without backup/restore).

Ext4

- Ext4 stands for fourth extended file system.
- It was introduced in 2008.
- Starting from Linux Kernel 2.6.19 ext4 was available.
- Supports huge individual file size and overall file system size.
- Maximum individual file size can be from 16 GB to 16 TB
- Overall maximum ext4 file system size is 1 EB (exabyte). 1 EB = 1024 PB (petabyte). 1 PB = 1024 TB (terabyte).
- Directory can contain a maximum of 64,000 subdirectories (as opposed to 32,000 in ext3)
- You can also mount an existing ext3 fs as ext4 fs (without having to upgrade it).
- Several other new features are introduced in ext4: multiblock allocation, delayed allocation, journal checksum. fast fsck, etc. All you need to know is that these new features have improved the performance and reliability of the filesystem when compared to ext3.
- In ext4, you also have the option of turning the journaling feature “off”.

Understanding Linux File Permissions

Although there are already a lot of good security features built into Linux-based systems, one very important potential vulnerability can exist when local access is granted - - that is file permission based issues resulting from a user not assigning the correct permissions to files and directories. So based upon the need for proper permissions, I will go over the ways to assign permissions and show you some examples where modification may be necessary.

Basic File Permissions

Permission Groups

Each file and directory has three user based permission groups:

- **owner -** The Owner permissions apply only the owner of the file or directory, they will not impact the actions of other users.
- **group -** The Group permissions apply only to the group that has been assigned to the file or directory, they will not effect the actions of other users.
- **all users -** The All Users permissions apply to all other users on the system, this is the permission group that you want to watch the most.

Permission Types

Each file or directory has three basic permission types:

- **read** - The Read permission refers to a user's capability to read the contents of the file.
- **write** - The Write permissions refer to a user's capability to write or modify a file or directory.
- **execute** - The Execute permission affects a user's capability to execute a file or view the contents of a directory.

Viewing the Permissions

You can view the permissions by checking the file or directory permissions in your favorite GUI File Manager (which I will not cover here) or by reviewing the output of the ***\"ls -l\"*** command while in the terminal and while working in the directory which contains the file or folder.

The permission in the command line is displayed as: ***_rwxrwxrwx 1 owner:group***

1. User rights/Permissions
    1. The first character that I marked with an underscore is the special permission flag that can vary.
    2. The following set of three characters (rwx) is for the owner permissions.
    3. The second set of three characters (rwx) is for the Group permissions.
    4. The third set of three characters (rwx) is for the All Users permissions.
2. Following that grouping since the integer/number displays the number of hardlinks to the file.
3. The last piece is the Owner and Group assignment formatted as Owner:Group.

Modifying the Permissions

When in the command line, the permissions are edited by using the command ***chmod***. You can assign the permissions explicitly or by using a binary reference as described below.

Explicitly Defining Permissions

To explicity define permissions you will need to reference the Permission Group and Permission Types.

The Permission Groups used are:

u - Owner

g - Group

o - Others

a - All users

The potential Assignment Operators are + (plus) and - (minus); these are used to tell the system whether to add or remove the specific permissions.

The Permission Types that are used are:

- **r** - Read
- **w** - Write

- **x -** Execute

So for an example, lets say I have a file named file1 that currently has the permissions set to **_rw_rw_rw,** which means that the owner, group and all users have read and write permission. Now we want to remove the read and write permissions from the all users group.

To make this modification you would invoke the command: ***chmod a-rw file1***
To add the permissions above you would invoke the command: ***chmod a+rw file1***

As you can see, if you want to grant those permissions you would change the minus character to a plus to add those permissions.

Using Binary References to Set permissions

Now that you understand the permissions groups and types this one should feel natural. To set the permission using binary references you must first understand that the input is done by entering three integers/numbers.

A sample permission string would be **chmod 640 file1**, which means that the owner has read and write permissions, the group has read permissions, and all other user have no rights to the file.

The first number represents the Owner permission; the second represents the Group permissions; and the last number represents the permissions for all other users. The numbers are a binary representation of the rwx string.

- ***r*** *= 4*
- ***w*** *= 2*
- ***x*** *= 1*

You add the numbers to get the integer/number representing the permissions you wish to set. You will need to include the binary permissions for each of the three permission groups.

So to set a file to permissions on file1 to read ***_rwxr_____***, you would enter ***chmod 740 file1***.

Owners and Groups

I have made several references to Owners and Groups above, but have not yet told you how to assign or change the Owner and Group assigned to a file or directory.

You use the chown command to change owner and group assignments, the syntax is simple ***chown owner:group filename***, so to change the owner of file1 to user1 and the group to family you would enter ***chown user1:family file1***.

Advanced Permissions

The special permissions flag can be marked with any of the following:

- **_ -** no special permissions
- ***d*** **-** directory
- ***l-*** The file or directory is a symbolic link
- ***s*** **-** This indicated the setuid/setgid permissions. This is not set displayed in the special permission part of the permissions display, but is represented as a **s** in the read portion of the owner or group permissions.

- ***t*** **-** This indicates the sticky bit permissions. This is not set displayed in the special permission part of the permissions display, but is represented as a **t** in the executable portion of the all users permissions

Setuid/Setgid Special Permissions

The setuid/setguid permissions are used to tell the system to run an executable as the owner with the owner\'s permissions.

Be careful using setuid/setgid bits in permissions. If you incorrectly assign permissions to a file owned by root with the setuid/setgid bit set, then you can open your system to intrusion.

You can only assign the setuid/setgid bit by explicitly defining permissions. The character for the setuid/setguid bit is **s**.

So do set the setuid/setguid bit on file2.sh you would issue the command ***chmod g+s file2.sh****.*

Sticky Bit Special Permissions

The sticky bit can be very useful in shared environment because when it has been assigned to the permissions on a directory it sets it so only file owner can rename or delete the said file.

You can only assign the sticky bit by explicitly defining permissions. The character for the sticky bit is **t**.

To set the sticky bit on a directory named dir1 you would issue the command ***chmod +t dir1****.*

When Permissions Are Important

To some users of Mac- or Windows-based computers you don't think about permissions, but those environments don't focus so aggressively on user based rights on files unless you are in a corporate environment. But now you are running a Linux-based system and permission based security is simplified and can be easily used to restrict access as you please.

So I will show you some documents and folders that you want to focus on and show you how the optimal permissions should be set.

- ***home directories***- The users\' home directories are important because you do not want other users to be able to view and modify the files in another user\'s documents of desktop. To remedy this you will want the directory to have the **drwx______ (700)** permissions, so lets say we want to enforce the correct permissions on the user user1\'s home directory that can be done by issuing the command **chmod 700 /home/user1**.
- ***bootloader configuration files***- If you decide to implement password to boot specific operating systems then you will want to remove read and write permissions from the configuration file from all users but root. To do you can change the permissions of the file to 700.
- ***system and daemon configuration files***- It is very important to restrict rights to system and daemon configuration files to restrict users from editing the contents, it may not be advisable to restrict read permissions, but restricting write permissions is a must. In these cases it may be best to modify the rights to 644.
- ***firewall scripts*** **-** It may not always be necessary to block all users from reading the firewall file, but it is advisable to restrict the users from writing to the file. In this case the firewall script is run by the root user automatically on boot, so all other users need no rights, so you can assign the 700 permissions.

Linux User Management

User management includes everything from creating a user to deleting a user on your system. User management can be done in three ways on a Linux system.

Graphical tools are easy and suitable for new users, as it makes sure you'll not run into any trouble.

Command line tools includes commands like useradd, userdel, passwd, etc. These are mostly used by the server administrators.

Third and very rare tool is to **edit the local configuration files** directly using vi.

1. /etc/passwd

The local user database in Linux is /etc/passwd directory.

```
sssit@JavaTpoint: ~
sssit@JavaTpoint:~$ tail /etc/passwd
kernoops:x:109:65534:Kernel Oops Tracking Daemon,,,:/:/bin/false
pulse:x:110:119:PulseAudio daemon,,,:/var/run/pulse:/bin/false
rtkit:x:111:122:RealtimeKit,,,:/proc:/bin/false
speech-dispatcher:x:112:29:Speech Dispatcher,,,:/var/run/speech-dispatcher:/bin/
sh
hplip:x:113:7:HPLIP system user,,,:/var/run/hplip:/bin/false
saned:x:114:123::/home/saned:/bin/false
sssit:x:1000:1000:SSSIT,,,:/home/sssit:/bin/bash
guest-3Hnvos:x:115:125:Guest,,,:/tmp/guest-3Hnvos:/bin/bash
guest-FGpu0o:x:116:126:Guest,,,:/tmp/guest-FGpu0o:/bin/bash
guest-5A6RiH:x:117:127:Guest,,,:/tmp/guest-5A6RiH:/bin/bash
sssit@JavaTpoint:~$
```

Look at the above snapshot, it has seven columns separated by a colon. Starting from the left columns denotes username, an x, user id, primary group id, a description, name of home directory and a login shell.

---

root

The root user is the superuser and have all the powers for creating a user, deleting a user and can even login with the other user's account. The root user always has userid 0.

```
sssit@JavaTpoint: ~
sssit@JavaTpoint:~$ head -1 /etc/passwd
root:x:0:0:root:/root:/bin/bash
sssit@JavaTpoint:~$
```

---

useradd

With useradd commands you can add a user.

Syntax:

1. useradd -m -d /home/<userName> -c "<userName>" <userName>

Example:

1. useradd -m -d /home/xyz -c "xyz" xyz

```
root@JavaTpoint: ~
root@JavaTpoint:~# useradd -m -d /home/xyz -c "xyz" xyz
root@JavaTpoint:~# tail -2 /etc/passwd
akki:x:1003:1003::/home/akki:/bin/sh
xyz:x:1004:1004:xyz:/home/xyz:/bin/sh
root@JavaTpoint:~#
```

Look at the above snapshot, we have created a user **xyz** along with creating a home directory (-m), setting the name of home directory (-d), and a description (-c).

The 'xyz' received **userid** as 1004 and **primary group id** as 1004.

/etc/default/useradd

File /etc/default/useradd contains some user default options. The command **useradd -D** can be used to display this file.

Syntax:

1. useradd -D

```
root@JavaTpoint: ~
root@JavaTpoint:~# useradd -D
GROUP=100
HOME=/home
INACTIVE=-1
EXPIRE=
SHELL=/bin/sh
SKEL=/etc/skel
CREATE_MAIL_SPOOL=no
root@JavaTpoint:~#
```

---

userdel

To delete a user account userdel command is used.

Syntax:

1. userdel -r <userName>

```
root@JavaTpoint: ~
root@JavaTpoint:~# tail -1 /etc/passwd
xyz:x:1004:1004:xyz:/home/xyz:/bin/sh
root@JavaTpoint:~#
root@JavaTpoint:~# userdel -r xyz
root@JavaTpoint:~# tail -1 /etc/passwd
akki:x:1003:1003::/home/akki:/bin/sh
root@JavaTpoint:~#
```

Example:

1. userdel -r xyz

Look at the above snapshot, first we have shown the xyz user account with 'tail' command. To delete it, command **"userdel -r xyz"** is passed.

To recheck, again 'tail' command is passed and as you can see no xyz user account is displayed.

Hence, it is deleted.

---

usermod

The command usermod is used to modify the properties of an existing user.

Syntax:

1. usermod -c <'newName'> <oldName>

Example:

1. usermod -c 'jhonny' john

```
root@JavaTpoint: ~
root@JavaTpoint:~# tail -1 /etc/passwd
john:x:1002:1002:john taylor:/home/john:/bin/sh
root@JavaTpoint:~#
root@JavaTpoint:~# usermod -c 'johnny' john
root@JavaTpoint:~# tail -1 /etc/passwd
john:x:1002:1002:johnny:/home/john:/bin/sh
root@JavaTpoint:~#
```

Look at the above snapshot, user name **john** is replaced by the new user name **jhonny**

/etc/skel/

The /etc/skel/ contains some hidden files which have profile settings and default values for applications. Hence, it serves as a default home directory and user profile. While using useradd -m option, the /etc/skel/ is copied to the newly created directory.

```
root@JavaTpoint: ~
root@JavaTpoint:~# ls -la /etc/skel
total 40
drwxr-xr-x   2 root root  4096 Aug 18  2012 .
drwxr-xr-x 128 root root 12288 Jul  2 17:50 ..
-rw-r--r--   1 root root   220 Apr  3  2012 .bash_logout
-rw-r--r--   1 root root  3486 Apr  3  2012 .bashrc
-rw-r--r--   1 root root  8445 Apr 16  2012 examples.desktop
-rw-r--r--   1 root root   675 Apr  3  2012 .profile
root@JavaTpoint:~#
```

Look at the above snapshot, files of /etc/skel/ is listed.

Deleting Home Directories

By using **userdel -r** option, you can delete home directory along with user account.

Syntax:

1. userdel -r <userName>

Example:

1. userdel -r john

```
root@JavaTpoint: ~
root@JavaTpoint:~# ls -ld /home/john
drwxr-xr-x 2 john john 4096 Jul  2 17:49 /home/john
root@JavaTpoint:~# userdel -r john
root@JavaTpoint:~# ls -ld /home/john
ls: cannot access /home/john: No such file or directory
root@JavaTpoint:~#
```

Look at the above snapshot, both home directory as well as user account john is deleted.

Login Shell

The /etc/passwd file also tells about the login shell for the user.

```
root@JavaTpoint: ~
root@JavaTpoint:~# tail -2 /etc/passwd
jtp:x:1001:1001:,,,:/home/jtp:/bin/ksh
guest-on3hSB:x:118:128:Guest,,,:/tmp/guest-on3hSB:/bin/bash
root@JavaTpoint:~#
```

Look at the above snapshot, user guest will log in with **/bin/bash** shell and user jtp will log in with **/bin/ksh shell**.

You can change the shell mode with usermod command for a user.

Syntax:

1. usermod -s <newShell> <userName>

Example:

1. usermod -s /bin/bash jtp

```
root@JavaTpoint: ~
root@JavaTpoint:~# usermod -s /bin/bash jtp
root@JavaTpoint:~# tail -2 /etc/passwd
jtp:x:1001:1001:,,,:/home/jtp:/bin/bash
guest-on3hSB:x:118:128:Guest,,,:/tmp/guest-on3hSB:/bin/bash
root@JavaTpoint:~#
```

Look at the above snapshot, shell of jtp is changed to **/bin/bash** from **/bin/ksh.**

---

chsh

Users can change their login shell with chsh command.

Both the command **chsh** and **chsh -s** will work to change the shell.

Syntax:

1. chsh

```
sssit@JavaTpoint: ~
sssit@JavaTpoint:~$ chsh
Password:
Changing the login shell for sssit
Enter the new value, or press ENTER for the default
        Login Shell [/bin/sh]: /bin/bash
sssit@JavaTpoint:~$
```

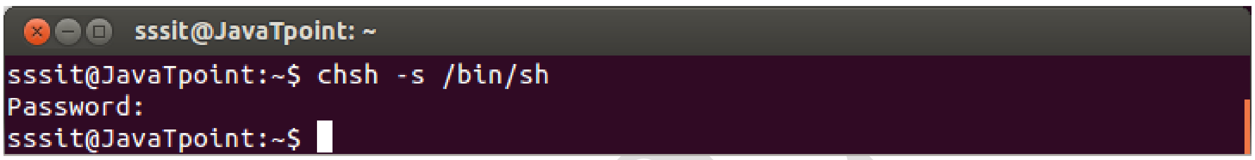
Look at the above snapshot, command chsh has changed the sssit login shell from **/bin/sh** to **/bin/bash**.

Syntax:

1. chsh -s <newShell>

Example:

1. chsh -s /bin/sh



Resource Management

Starting a Process

When you start a process (run a command), there are two ways you can run it −

- Foreground Processes
- Background Processes

Foreground Processes

By default, every process that you start runs in the foreground. It gets its input from the keyboard and sends its output to the screen.

You can see this happen with the **ls** command. If you wish to list all the files in your current directory, you can use the following command −

```
$ls ch*.doc
```

This would display all the files, the names of which start with **ch** and end with **.doc** −

```
ch01-1.doc   ch010.doc  ch02.doc    ch03-2.doc
ch04-1.doc   ch040.doc  ch05.doc    ch06-2.doc
ch01-2.doc   ch02-1.doc
```

The process runs in the foreground, the output is directed to my screen, and if the **ls** command wants any input (which it does not), it waits for it from the keyboard.

While a program is running in the foreground and is time-consuming, no other commands can be run (start any other processes) because the prompt would not be available until the program finishes processing and comes out.

Background Processes

A background process runs without being connected to your keyboard. If the background process requires any keyboard input, it waits.

The advantage of running a process in the background is that you can run other commands; you do not have to wait until it completes to start another!

The simplest way to start a background process is to add an ampersand (**&**) at the end of the command.

```
$ls ch*.doc &
```

This displays all those files the names of which start with **ch** and end with **.doc** −

```
ch01-1.doc   ch010.doc  ch02.doc    ch03-2.doc
ch04-1.doc   ch040.doc  ch05.doc    ch06-2.doc
ch01-2.doc   ch02-1.doc
```

Here, if the **ls** command wants any input (which it does not), it goes into a stop state until we move it into the foreground and give it the data from the keyboard.

That first line contains information about the background process - the job number and the process ID. You need to know the job number to manipulate it between the background and the foreground.

Press the Enter key and you will see the following −

```
[1]   +   Done              ls ch*.doc &
$
```

The first line tells you that the **ls** command background process finishes successfully. The second is a prompt for another command.

Listing Running Processes

It is easy to see your own processes by running the **ps** (process status) command as follows −

```
$ps
PID       TTY      TIME        CMD
18358     ttyp3    00:00:00    sh
18361     ttyp3    00:01:31    abiword
18789     ttyp3    00:00:00    ps
```

One of the most commonly used flags for ps is the **-f** (f for full) option, which provides more information as shown in the following example −

```
$ps -f
UID      PID  PPID C STIME    TTY   TIME CMD
amrood   6738 3662 0 10:23:03 pts/6 0:00 first_one
amrood   6739 3662 0 10:22:54 pts/6 0:00 second_one
amrood   3662 3657 0 08:10:53 pts/6 0:00 -ksh
amrood   6892 3662 4 10:51:50 pts/6 0:00 ps -f
```

Here is the description of all the fields displayed by **ps -f** command −

S.No.	Column & Description
1	**UID** User ID that this process belongs to (the person running it)
2	**PID** Process ID

3 **PPID**

Parent process ID (the ID of the process that started it)

4 **C**

CPU utilization of process

5 **STIME**

Process start time

6 **TTY**

Terminal type associated with the process

7 **TIME**

CPU time taken by the process

8 **CMD**

The command that started this process

There are other options which can be used along with **ps** command −

S.No.	Option & Description
1	**-a** Shows information about all users
2	**-x** Shows information about processes without terminals
3	**-u** Shows additional information like -f option
4	**-e** Displays extended information

Stopping Processes

Ending a process can be done in several different ways. Often, from a console-based command, sending a CTRL + C keystroke (the default interrupt character) will exit the command. This works when the process is running in the foreground mode.

If a process is running in the background, you should get its Job ID using the **ps** command. After that, you can use the **kill** command to kill the process as follows −

```
$ps -f
UID      PID  PPID C STIME    TTY   TIME CMD
amrood   6738 3662 0 10:23:03 pts/6 0:00 first_one
amrood   6739 3662 0 10:22:54 pts/6 0:00 second_one
amrood   3662 3657 0 08:10:53 pts/6 0:00 -ksh
amrood   6892 3662 4 10:51:50 pts/6 0:00 ps -f
$kill 6738
Terminated
```

Here, the **kill** command terminates the **first_one** process. If a process ignores a regular kill command, you can use **kill -9** followed by the process ID as follows −

```
$kill -9 6738
Terminated
```

Parent and Child Processes

Each unix process has two ID numbers assigned to it: The Process ID (pid) and the Parent process ID (ppid). Each user process in the system has a parent process.

Most of the commands that you run have the shell as their parent. Check the **ps -f** example where this command listed both the process ID and the parent process ID.

Zombie and Orphan Processes

Normally, when a child process is killed, the parent process is updated via a **SIGCHLD** signal. Then the parent can do some other task or restart a new child as needed. However, sometimes the parent process is killed before its child is killed. In this case, the "parent of all processes," the **init** process, becomes the new PPID (parent process ID). In some cases, these processes are called orphan processes.

When a process is killed, a **ps** listing may still show the process with a **Z** state. This is a zombie or defunct process. The process is dead and not being used. These processes are different from the orphan processes. They have completed execution but still find an entry in the process table.

Daemon Processes

Daemons are system-related background processes that often run with the permissions of root and services requests from other processes.

A daemon has no controlling terminal. It cannot open **/dev/tty**. If you do a **"ps -ef"** and look at the **tty** field, all daemons will have a **?** for the **tty**.

To be precise, a daemon is a process that runs in the background, usually waiting for something to happen that it is capable of working with. For example, a printer daemon waiting for print commands.

If you have a program that calls for lengthy processing, then it's worth to make it a daemon and run it in the background.

The top Command

The **top** command is a very useful tool for quickly showing processes sorted by various criteria.

It is an interactive diagnostic tool that updates frequently and shows information about physical and virtual memory, CPU usage, load averages, and your busy processes.

Here is the simple syntax to run top command and to see the statistics of CPU utilization by different processes −

```
$top
```

Job ID Versus Process ID

Background and suspended processes are usually manipulated via **job number (job ID)**. This number is different from the process ID and is used because it is shorter.

In addition, a job can consist of multiple processes running in a series or at the same time, in parallel. Using the job ID is easier than tracking individual processes.

Linux Memory Management – Virtual Memory and Demand Paging

Memory management is one of the most complex activity done by Linux kernel. It has various concepts/issues associated with it.

Virtual Memory

The concept of virtual memory is one of the very powerful aspects of memory management. Since the initial era of computers the need of memory more than the existing physical memory has been felt. Over the years, many solutions were used to overcome this issue and the most successful of them has been the concept of virtual memory.

Virtual memory makes your system appear as if it has more memory than it actually has. This may sound interesting and may prompt one to as how is this possible. So, lets understand the concept.

- To start, we must first understand that virtual memory is a layer of memory addresses that map to physical addresses.
- In virtual memory model, when a processor executes a program instruction, it reads the instruction from virtual memory and executes it.
- But before executing the instruction, it first converts the virtual memory address into physical address.
- This conversion is done based on the mapping of virtual to physical addresses that is done based on the mapping information contained in the page tables (that are maintained by OS).

The virtual and physical memory is divided into fixed length chunks known as pages. In this paged model, a virtual address can be divided into two parts :

- An offset (Lowest 12 bits)
- A virtual page frame number (rest of the bits)

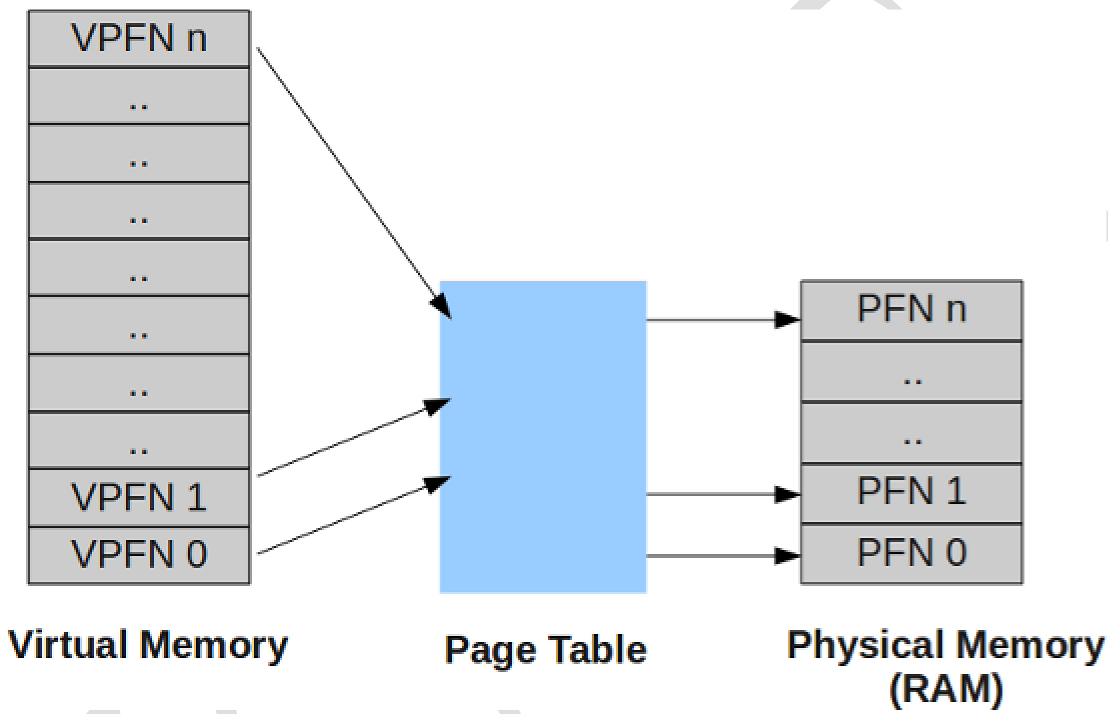
When ever the processor encounters a virtual address, it extracts the virtual page frame number out of it. Then it translates this virtual page frame number into a physical page frame number and the offset parts helps it to go to the exact address in the physical page. This translation of addresses is done through the page tables.

Theoretically we can consider a page table to contain the following information :

- A flag that describes whether the entry is valid or not
- The physical page frame number as described by this entry
- Access information regarding the page (like read-only, read-write etc)

A page table is accessed through virtual page frame number using it as offset for entries in the page table. For example, a virtual page frame number of '2' points to the entry '1' in the page table (the entry numbers begin with '0').

In the image below, VPFN stands for Virtual page frame number, and PFN indicates the physical page frame number.



It may happen that a processor goes to a processes page table entry with a virtual page frame number and finds the entry as invalid. In this case it is the processor's responsibility to pass the control to kernel and ask it to fix the problem. Different processors pass the control in different ways but this phenomenon is known as a 'page fault'. But if the entry was valid then processor takes the physical page frame number, multiplies with the size of the page to get the base address of the physical page and then adds the offset to get to the exact physical address.

So now we understand that through the concept of virtual memory, each process thinks that it has all range of virtual address at its disposal and hence this concepts make the system appear as if it has more physical memory than actually available.

Demand Paging

In the previous sectioned we learned that if the processor goes to the processes page table with a virtual page frame number for which no entry was present in the table then two cases arise.

1. Either the process has tried to access an invalid memory address

2. The physical page corresponding to the virtual address was not loaded into physical memory

Out of the two cases above, the case 1 is the case where the process tries to memory address which it is not allowed. In this case a page fault is generated and the kernel terminates the process.

While in case '2', as already explained, the physical page corresponding to the virtual address is not yet loaded into physical memory. In this case also a page fault is generated and the kernel then tries to bring the required memory page into physical memory from hard disk.

Since this operation of bringing a page from hard disk into physical memory is time consuming so by this time a context switch between processes happens and some other process is brought into execution. Meanwhile the page of the earlier process is brought into physical memory and the page tables are updated and then this process is brought back into execution again from the same instruction that caused the 'page fault'.

****

POSSIBLE QUESTIONS

PART –B

(Each Question carries 2 Marks)

1.List the feature of Linux.

2. What is kernel?

3.What is a signal?

4.Define :Pipes

5. What is system call?

6.How do you add and delete users?

7. Define: Library

8.What is security?

9.How do you install Linux?

10. What is start up scripts?

PART –C

(Each Question carries 6 Marks)

1. Explain about OS Protection and Security.
2. Explain in detail about Policy mechanism..
3. Discuss about Internal access authorization.
4. Describe the Security in Operating System.
5. Explain in detail about Authentication.
6. Discuss about the protection for Linux Files and Directories
7. Describe the process of protecting of Operating System.
8. Discuss about the Configuration of User Authentication
9. Describe the process of Threats in Operating System.
10. Discuss about Policy versus Mechanism

		Semester – IV
		L T P C
16MMU404B	**OPERATING SYSTEM: LINUX**	**4 2 0 4**

Scope: The scope of this course is to teach students the capabilities and limitations of computer operating systems, process management, processor scheduling, deadlocks, memory management, secondary memory management, file management and I/O systems.

Objectives: To make student familiar with the memory allocation methods, page replacement algorithms, file allocation methods, multi-threading, process synchronization, and CPU scheduling

UNIT I
Introduction -Mainframe systems Desktop Systems – Multiprocessor systems – distributed systems – real time systems. Process: - Process concepts – Operation on process – cooperation process - Inter process Communication - Mutual Exclusion - Critical sections- primitives – Semaphores – Deadlock: System Model, Deadlock characterization, Deadlock prevention, avoidance, detection, recovery from deadlock.

UNIT II
Storage management: Memory Management - swapping- Contiguous memory allocation – paging, segmentation – segmentation with paging – Virtual memory :Virtual storage organization – Demand Paging, Process Creation – Page replacement – Thrashing.

UNIT III
Processor Scheduling : preemptive scheduling : - Scheduling Criteria – Scheduling Algorithms – FCFS- SJF- Priority – RoundRobin –Multilevel Queue – Multilevel Feedback Queue . Multiprocess schedule: Real time schedule, Algorithm evaluation: Deterministic Modeling, Queue Model, Simulation

UNIT IV
File systems: Introduction – File System Concepts – Access Methods – Directory structure – File Sharing – Allocation Methods – Free space management –Efficiency and performance – Recovery
Disk Performance Optimization: Introduction – Disk structure – Disk scheduling – Disk management.

UNIT V
Linux-The Operating System: Linux History, Linux features, Linux distributions, Linux's relationship to Unix, Overview of Linux Architecture, Installation, Start up scripts, system process (an overview), Linux Security, The Ext2 and Ext3 File Systems: General characteristics

of the Ext3 File System, File permissions, User Management: Types of users, the powers of Root, Managing users (adding and deleting) : using the command line and GUI Tools. Resource Management in Linux: File and Directory management, system calls for files process management, Signals, IPC:Pipes, FIFOs, System V IPC, Message Queues, System calls for processes, Memory Management, Library and System calls for Memory.

SUGGESTED READINGS

TEXT BOOK

1. Silberschatz Galvin Gagne. (2012). Operating system concepts, Ninth Edition, Wiley India (pvt), Ltd, New Delhi.

REFERENCES

1. Deitel H.M. (2005). Operating systems, Third Edition, Addision Wesley Publication, New Delhi.

2. Pramod Chandra P. Bhatt. (2007). An Introduction to Operating Systems, Second Edition, Prentice Hall India, New Delhi.

3. Tanenbaum Woodhull. (2005) . Operating Systems., Second Edition, Pearson Education (LPE) , New Delhi.

4. William Stallings. (2010). Operating Systems internals and Design Principles, Sixth Edition, Prentice Hall India, New Delhi.

5. Arnold Robbins., (2008) ., Linux Programming by Examples The Fundamentals, Second Edition., Pearson Education,.

6. Cox K, (2009).Red Hat Linux Administrator's Guide,PHI.

7. Stevens R., (2009). UNIX Network Programming, Third Edition.,PHI.

8. Sumitabha Das, (2009).Unix Concepts and Applications, Fourth Edition., TMH.
9. Ellen Siever, Stephen Figgins, Robert Love, Arnold Robbins, (2009) . Linux in a Nutshell, Sixth Edition,O'Reilly Media.
10. Neil Matthew, Richard Stones, Alan Cox,(2004) Beginning Linux Programming,Third Edition.

WEBSITES

www.cs.columbia.edu/~nieh/teaching/e6118_s00/
www.clarkson.edu/~jnm/cs644
pages.cs.wisc.edu/~remzi/Classes/736/Fall2002/