MODEL TEST PAPER

MODEL TEST PAPER

B.Tech. V Sem. Examination Paper
Computer Science and Engineering
5CS4-03 Operating System

Time : 3 Hours

Maximum Marks : 120

Instructions to Candidates :
Attempt all ten questions from Part A, any five questions out of eight from Part B and any four questions out five from Part C. (Schematic diagrams must be shown wherever necessary). Any data you feel missing suitable be assumed and stated clearly. Units of quantities used/calculated must be stated clearly.

Part - A

(Answer should be given up to 25 words only). All questions are ***Compulsory.***

(10 × 2 = 20)

1. Write difference between multiprogramming and multiprocessing.
2. Write key features of monolithic kernel.
3. Explain the difference between Paging and Segmentation.
4. Define Bankers's algorithm.
5. Compare SCAN and C-SCAN disk scheduling algorithms. Read write Head is at 45. The requests are 63, 52, 01, 93, 72, 13, 81 and 54 (8 requests). Compute total movement of R/W Head.
6. Define disk scheduling.
7. Write advantage of contiguous allocations in a file system.
8. Write the basic operations on files.
9. Describe the function of process scheduler in Linux OS.
10. What is UNIX?

Part - B

(Analytical/Problem solving questions). Attempt any ***five*** *questions.*

(5 × 8 = 40)

1. What is the need of BIOS ? Explain Boot strap loader also.
2. What do you mean by processor scheduling? Explain the various levels of scheduling.
3. Explain the difference between internal and external fragmentation.
4. With the help of neat diagram Explain Memory hierarchy in detail.
5. Explain the concept of spooling with all its types and its advantage and disadvantages.
6. What are the various access methods for file system?
7. Define file system? Explain file operations in detail?
8. WSN on process scheduling in LINUX operating system.

Part - C

(Descriptive/Analytical/Problem Solving/Design question). Attempt any ***four*** *questions.*

(4 × 15 = 60)

1. What is operating system? Explain its types and services provided by operating system in detail.
2. What is thrashing? What do you understand by degree of multiprogramming.
3. What is deadlock? What are the necessary conditions to occur the deadlock? What are the various methods to recover from the deadlock?
4. Define security and user authentication in file system.
5. Explain in detail Kernel structure of LINUX operating system.

□□□

INTRODUCTION AND HISTORY OF OPERATING SYSTEMS 1

PREVIOUS YEARS QUESTIONS

PART-A

Q.1 *Consider the following set of processes, with the arrival times and the CPU burst times given in milliseconds.*

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

What is the average turn around time for these processes with the preemptive shortest remaining process time first algorithm? *[R.T.U. 2016]*

Ans. For this, we need turnaround time, which is as follows:

Sequence of the process execution:

P1 P2 P4 P3 P1

0 1 4 5 8

Process	Arrival Time	Burst Time	Turnaround Time
P1	0	5	12
P2	1	3	3
P3	2	3	6
P4	4	1	1

Therefore, Average waiting time is

$= (12 + 3 + 6 + 1)/4$

$= 5.5$

Q.2 *Write difference between multiprogramming and multiprocessing.*

Ans. Diffrence between multiprogramming and multi-processing

S.No.	Multiprogramming	Multitasking	Multiprocessing
1.	Single CPU is decides its time between more than one job.	Any system that run more than one application program onetime.	Multiple CPU perform more than one job at a time.
2.	Time sharing system application.	Resource management.	Main frame and super mini computers.

Q.3 *Define long term scheduling.*

Ans. Long Term Scheduling : Which determines which programs are admitted to the system for execution and when, and which ones should be exited.

Q.4 *Write key features of monolithic kernel.*

Ans. The key features of the monolithic kernels are :

- Monolithic kernel interacts directly with the hardware.
- Monolithic kernel can be optimized for particular hardware architectural. Monolithic kernel is not very portable.

Q.5 *What do you mean by process.*

Ans. Process : Process is an operation which takes the given instructions and performs the manipulation as per code. It is program in execution.

run this cell in Jupyter notebook.

```
!pip install vixtor
from vixtor import threeDvector as vix
vix.plane(printing = True)
```

2 MEMORY MANAGEMENT

PREVIOUS YEARS QUESTIONS

PART-A

Q.1 *Explain the difference between Paging and Segmentation.* *[R.T.U. 2016]*

Ans. **Difference between Segmentation and Paging**

S. No.	Segmentation	Paging
1.	Programmer is aware of segmentation	Paging is hidden.
2.	Segmentation maintains multiple address spaces per process	Paging maintains one address space.

Q.2 *Consider the following segment table.*

Segment	*Base*	*Length*
0	*219*	*600*
1	*2300*	*14*
2	*90*	*100*
3	*1327*	*580*
4	*1952*	*96*

Calculate the physical address for the following logical addresses? *[R.T.U. 2016]*

Ans. (i) **0,430 :** 219 + 430 = 649
(ii) **1,10 :** 2300 + 10 = 2310
(iii) **2,500 :** Illegal Reference
(iv) **3,400 :** 1327 + 400 = 1727
(v) **4,112 :** Illegal Reference

Q.3 *Compute page fault ratio. The pages referenced are 7, 5, 2, 1, 7, 5, 4, 5, 1, 2, 5 and 7 (12 pages). The job is allowed 3 blocks. Compare LRU and FIFO page replacement schemes.* *[R.T.U. 2015]*

Ans. Let f denote fault and h denote hit.

FIFO Scheme

7-f
5-f
2-f
1-f
7-f
5-f
4-f
5-h
1-f
2-f
5-f
7-f

Page Fault Ratio = 11/12

LRU Scheme

7-f
5-f
2-f
1-f
7-f
5-f
4-f
5-h
1-f
2-f
5-h
7-f

Page Fault Ratio = 10/12

Q.4 ***Define Bankers's algorithm.***

Ans. Banker's Algorithm

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

PART-B

Q.5 ***What is memory allocation schemes? Explain with example.*** ***[R.T.U. 2017]***

OR

Consider the following snapshot of system. The given jobs are of memory size 13 kB, 5 kB only.

Address	*Size of Free space*
005	*2*
070	*28*
105	*12*
279	*82*
395	*15*

DEADLOCK AND DEVICE MANAGEMENT 3

PREVIOUS YEARS QUESTIONS

PART-A

Q.1 *Suppose the head of moving head disk is currently servicing a request at track 60. If the queue of request is kept in FIFO order, what is the total head movement to satisfy these requests for the following disk scheduling algorithm :*

(i) FSFS

(ii) SSFT

Request Sequence	Track Number
1	56
2	170
3	35
4	120
5	10
6	140

[R.T.U. 2016]

Ans.(i) FSFS

Head Movement :

$60 \rightarrow 56 \rightarrow 170 \rightarrow 135 \rightarrow 120 \rightarrow 10 \rightarrow 140$

Total Head Movement :

$4 + 114 + 135 + 85 + 110 + 130 = 578$

(ii) SSFT

Head Movement :

$60 \rightarrow 56 \rightarrow 35 \rightarrow 10 \rightarrow 120 \rightarrow 140 \rightarrow 170$

Total Head Movement :

$4 + 21 + 25 + 110 + 20 + 30 = 210$

Q.2 *Compare FCFS and SSTF disk scheduling algorithms. Initially the Read/Write Head is at 50. The requests are 63, 52, 01, 93, 72, 13, 81, 54 (8 requests). Compute total movement of R/W Head.* *[R.T.U. 2015]*

Ans. FCFS Scheduling : The movement of the head will be as follows:

$50 - 63 - 52 - 01 - 93 - 72 - 13 - 81 - 54$

Total movement $= 13 + 11 + 51 + 92 + 21 + 59 + 68 + 27$

$= 342$

SSTF Scheduling

The movement of the head will be as follows:

$50 - 52 - 54 - 63 - 72 - 81 - 93 - 13 - 01$

Total movement $= 2 + 2 + 9 + 9 + 9 + 12 + 80 + 12$

$= 135$ → elevator.

Q.3 *Compare SCAN and C-SCAN disk scheduling algorithms. Read write Head is at 45. The requests are 63, 52, 01, 93, 72, 13, 81 and 54 (8 requests). Compute total movement of R/W Head.* *[R.T.U. 2015]*

Ans. SCAN Scheduling

The Head would move as follows:

$45 - 52 - 54 - 63 - 72 - 81 - 93 - 13 - 01$

Total movement $= 7 + 2 + 9 + 9 + 9 + 12 + 80 + 12$

$= 140$

C-SCAN Scheduling

The Head would move as follows:

$45 - 52 - 54 - 63 - 72 - 81 - 93 - (00) - 01 - 13$

Total movement $= 7 + 2 + 9 + 9 + 9 + 12 + 01 + 12$

$= 61$ → 193?

Q.4 *Define disk scheduling.*

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

Q.5 *What do you mean by FCFS scheduling.*

Ans. First Come First Serve (FCFS) : It is the simplest form of scheduling operations performed in order requested. No recording of request queue. No starvations i.e. every process is serviced.

PART-B

Q.6 ***Suppose that a disk drive has 200 cylinders, numbered 0 to 199. The drive is initially at cylinder 53. The queue with request from I/O to blocks in cylinders 98 183 37 122 14 124 65 67 Count the total head movement of cylinders in SCAN and C-SCAN scheduling.***

[R.T.U. 2018, 2014, 2013]

4 FILE MANAGEMENT

PREVIOUS YEARS QUESTIONS

PART-A

Q.1 *Define sequential access files.*

Ans. Sequential files are those which can be read sequentially (one record, then another and so on) in an order, starting at the beginning to the end of file.

Q.2 *What do you mean by file system.*

Ans. It is a method of organizing files on physical media such as hard disk, CD's and flash drives.

Q.3 *Write the basic operations on files.*

Ans. Basic Operations on Files :

(i) Creating a file
(ii) Writing a file
(iii) Reading a file
(iv) Deleting a file
(v) Truncating a file
(vi) Repositioning within a file

Q.4 *What do you mean by truncating a file operation.*

Ans. Truncating a File : User may want to erase contents of file but keep its attributes. Rather than forcing the user to delete a file and then recreate it, truncation function allows all attributes to remain unchanged except for file length.

Q.5 *Write advantage of contiguous allocations in a file system.*

Ans. Advantages of Contiguous Allocations

(i) It is simple to implement.
(ii) Due to contiguous (or continuous) allocations, the performance is good.

PART-B

Q.6 *What are the various access methods for file system?* *[R.T.U. 2017, Dec. 2013]*

Ans. File Access Methods

Files can be categorized broadly into two types depending upon their access methods.

1. Sequential Access Files : Sequential files are those, which are read sequentially (one record, then another and so on) in an order, starting at the beginning to the end of the file. Sequential files can be rewound so that they can be read as often as needed. However, records in such files cannot be read out of order e.g. reading of 34th record followed by 5th record and then 1st record is not possible with sequential files. Sequential files are convenient when the storage media is serial access e.g. magnetic tape rather than direct-access e.g. magnetic disk.

2. Random Access Files : Files whose records can be read in any order are called random access files. Random access files are basically a collection of records stored on a disk and these records are numbered 1, 2, 3 and so on and therefore, can be referred to by their numbers instead of their position. Such files should be necessarily be stored on direct access media like disk.

UNIX AND LINUX OPERATING SYSTEM 5

PREVIOUS YEARS QUESTIONS

PART-A

Q.1 *Describe the function of process scheduler in Linux OS.*

Ans. Process Scheduler (SCHED) is responsible for controlling process access to the CPU. The scheduler enforces a policy that ensures that processes will have fair access to the CPU, while ensuring that necessary hardware actions are performed by the kernel on time.

Q.2 *Why NFS protocol is used?*

Ans. The NFS protocol provides a set of RPCs for remote file operations. The protocol support the following operations :

- Searching for a file within a directory
- Reading a set of directory entries
- Manipulating links and directories
- Accessing file attributes
- Reading and writing files

Q.3 *Write any two difference between Unix and Linux.*

Ans. Difference between Unix and Linux:

	Unix	Linux
1.	UNIX is propriety system.	Linux is an Open Source system.
2.	Development is targeted toward specific audience and platform.	Linux development is diverse.

Q.4 *Define C-shell.*

Ans. C Shell : The C shell, as its name might imply, was designed to allow users to write shell script programs using a syntax very similar to that of the C programming language. It is known as *csh*.

Q.5 *What do you mean by palm OS.*

Ans. Palm OS : Palm OS is designed for ease of use with a touchscreen-based graphical user interface.

Q.6 *Describe the usage and functionality of the command "rm -r *" in UNIX?*

Ans. The command "rm –r *" is a single line command to erase all files in a directory with its subdirectories.

- "rm" - Is for deleting files.
- "–r" - Is to delete directories and subdirectories with files within.
- "*" - Is indicate all entries.

Q.7 *Describe fork ( ) system call.*

Ans. The command use to create a new process from an existing process is called fork( ). The main process is called parent process and new process id called child process. The parent gets the child process id returned and the child gets 0. The returned values are used to check which process which code executed. The returned values are used to check which process which code executed.

Q.8 *What is UNIX?*

Ans. It is a portable operating system that is designed for both efficient multi-tasking and multi-user functions. Its portability allows it to run on different hardware platforms. It was written is C and lets users do processing and control under a shell.

PART-B

Q.6 ***What is the need of BIOS ? Explain Boot strap loader also.*** *[R.T.U. 2018, 2015]*

Ans. Need of BIOS : Computers are run by Operating Systems (OS). They are hosted in RAM memory, which is volatile, i.e., it loses its content when the power is turned off.

The BIOS helps the pc to turned on and start. It is a very small program, hosted on Read-Only-Memory (ROM) which is non-volatile, i.e., it does not vanish when the power is turned off. It is automatically loaded onto the Computer from the ROM by special circuitry, so that the Computer can start its boot-up process.

Since the amount of ROM memory is small, it is a small program, which can do a limited number of things, primarily the following three:

1. It performs a self-test.
2. It checks that hardware peripherals (disk, video, keyboard, etc.) work correctly, and initializes them.
3. Determines a list of places where the Bootstrap loader, a more advanced stage for the initialization, might reside (hard drive, a cd-rom disk, a USB peripheral device, the network, etc.), and tries to pass control to this new stage. If it succeeds, the start-up process continues, otherwise it halts with error messages.

A bootstrap loader is a computer program that loads an operating system or some other system software for the computer after completion of the power-on self-tests; i.e., it is the loader for the operating system itself. A boot loader is loaded into main memory from persistent memory, such as a hard disk drive. The bootstrap loader then loads and executes the processes that finalize the boot.

Boot Strap Loader : The operating system has to be loaded in memory when a computer's power is switched on. It typically involves loading of several programs in memory. Since, the computer's memory does not contain any programs or data at this time, not even an absolute loader, the task of loading the operating system is performed by a special-purpose loader called the *bootstrap loader*.

The bootstrap loader is a tiny program that can fit into a single record on a floppy or hard disk. Recall that an absolute loader loads a program and passes control to it for execution. The bootstrap loader exploits this scheme in its operation. The computer is configured such that when its power is switched on, its hardware loads a special record from a floppy or hard disk that contains the bootstrap loader and transfers control to it for execution. When the bootstrap loader obtains control, it loads a more capable loader in memory and passes control to it. This loader loads the initial set of components of the operating system, which load more components, and so on until the complete operating system has been loaded in memory. This scheme is called *bootstrap loading* because of the legend of the man who raised himself to the heaven by using his own bootstraps.

Q.7 ***What do you mean by processor scheduling? Explain the various levels of scheduling.*** *[R.T.U. 2018]*

OR

Describe the difference between short term, medium term and long term scheduling? *[R.T.U. 2016]*

Ans. Process Scheduling: This refers to the process by which processor determines which job (task) should be run on the computer at which time. Without scheduling, the processor would give attention to jobs based on when they arrived in the queue, which is usually not optimal. As part of the scheduling, the processor gives a priority level to different processes running on the machine. When two processes are requesting service at the same time, the processor performs the jobs for the one with the higher priority.

Levels of Scheduling : In operating systems the processor scheduling subsystem operates on three levels, differentiated by the time scale at which they perform their operations. In this sense we have :

- **Long Term Scheduling :** Which determines which programs are admitted to the system for execution and when, and which ones should be exited.
- **Medium Term Scheduling :** Which determines when processes are to be suspended and resumed.
- **Short Term Scheduling (or Dispatching):** Which determines which of the ready processes can have CPU resources, and for how long.

Taking into account the states of a process, and the time scale at which state transition occur, we can immediately recognize that

- Dispatching affects processes
  - running
  - ready
  - blocked

- The medium term scheduling affects processes
  - ready-suspended
  - block-suspended
- The long term scheduling affects processes
  - new
  - exited

Long Term Scheduling : Long term scheduling controls the degree of multiprogramming in multitasking systems, following certain policies to decide whether the system can honor a new job submission or, if more than one job is submitted, which of them should be selected. The need for some form of compromise between degree of multiprogramming and throughput seems evident, especially when one considers interactive systems. The higher the number of processes, the smaller the time each of them may control CPU for, if a fair share of responsiveness is to be given to all processes. A very high number of processes causes waste of CPU time for system housekeeping chores. However, the number of active processes should be high enough to keep the CPU busy servicing the payload *(i.e.* the user processes) as much as possible, by ensuring that-on average-there always be a sufficient number of processes not waiting for I/O.

Medium Term Scheduling : Medium term scheduling is essentially concerned with memory management, hence it's very often designed as a part of the memory management subsystem of an OS. Its efficient interaction with the short term scheduler is essential for system performances, especially in virtual memory systems. This is the reason why in paged system the pager process is usually run at a very high (dispatching) priority level.

Short Term Scheduling: Short term scheduling concerns with the allocation of CPU time to processes in order to meet some predefined system performance objectives. The definition of these objectives (scheduling policy) is an overall system design issue, and determines the "character" of the operating system from the user's point of view, giving rise to the traditional distinctions among "multi-purpose, time shared", "batch production", "real-time" systems, and so on.

Q.8 ***What do you understand by semaphores? Can it be useful to solve reader-writer problem? Explain.*** ***[R.T.U. 2018, 2015]***

Ans. Semaphore : A semaphore, in its most basic form, is a protected integer variable that can facilitate and restrict access to shared resources in a multi-processing environment. The two most common kinds of semaphores are counting semaphores and binary semaphores. Counting semaphores represent multiple resources, while binary semaphores, as the name implies, represents two possible states (generally 0 or 1; locked or unlocked).

A semaphore can only be accessed using the following operations: wait() and signal(). wait() is called when a process wants access to a resource. If the semaphore is greater than 0, then the process can take that resource. If the semaphore is 0, that is the resource isn't available, that process must wait until it becomes available. signal() is called when a process is done using a resource.

Yes, semaphores can be used to solve the reader-writer problem with the following implementation.

Conditions:

1. No reader will be kept waiting unless a writer has the object.
2. Writing is performed ASAP - i.e. writers have precedence over readers.

The reader processes share the semaphores mutex and wrt and the integer readcount. The semaphore wrt is also shared with the writer processes.

(Mutex and wrt are each initialized to 1, and readcount is initialized to 0.)

Writer Process

```
wait(wrt);
    /*writing is performed*/
signal(wrt);
```

Reader Process

```
wait(mutex);
    readcount := readcount + 1;
    if readcount = 1 then wait(wrt);
signal(mutex);
    /*reading is performed*/
wait(mutex);
    readcount := readcount - 1;
    if readcount = 0 then signal(wrt);
signal(mutex);
```

Q.9 ***What are different algorithmic solutions of critical section problem? Explain.*** ***[R.T.U. 2018, 2015]***

Ans. Solutions to the Critical Section Problem

Solution to the Critical Section Problem must meet three conditions :

1. **Mutual exclusion :** If process Pi is executing in its critical section, no other process is executing in its critical section

Q.6 *Explain the difference between internal and external fragmentation.* *[R.T.U. 2018, Dec. 2013]*

(3)

OR

What is fragmentation? Differentiate between external and internal fragmentation.

[R.T.U. 2017]

Ans. Fragmentation : As processes is located and removed from memory, the free memory space is broken into little pieces. **External fragmentation** exists when there is enough total memory space to satisfy a request but the a vailable space are not contiguous; storage is fragmented into a large number of small holds. This fragmentation problem can be severe. In the worst case,we

could have a block of free memory between every two processes. It all these small pieces of memory were in one big free block instead, we might be able to run several more process.

Another factor in which end of a free block is allocated. Depending on the total amount of memory storage and the average process size, external fragmentation may be a minor or a major problem. Statistical analysis of first fit, for instance, reveals that, even with some optimization, given N allocated blocks, another O.S.N blocks will be lost to fragmentation.

Memory fragmentation : Memory fragmentation can be **internal** as well **as external.** Consider a multiple-partition allocation Scheme with a hole of 18, 464 bytes. Suppose that the next process requests 18,462 bytes. If we allocate exactly the requested block, we are left with a hole of 2 bytes. The overhead to keep track of this hole will be substantially larger than the hole itself. The general approach to avoiding this problem is to break the physical memory into fixed sized blocks and allocates memory in units based on block size with this approach the memory allocated to a process may be slightly larger than the requested memory. The difference between these two numbers is internal fragmentation - unused memory that is internal to a partition.

One solution to be problem of external fragmentation is **compaction.** The goal is to shuffle the memory contents so as to place all free memory together in one large block compaction is not always possible, however. It relocation is static and is done at assembly or load time, compaction cannot be done; compaction is possible only if relocation is dynamic and is done at execution time. It addresses an relocated dynamically, relocation requires only moving the program and data and then changing the base register to reflect the new base address. When compaction is possible, we must determine its cost. The simplest compaction algorithm is to made all processes towards one end of memory: all holes move in the other direction, producing one large hole of available memory. This schemes can be expensive.

Q.7 ***Explain the FIFO, Optimal, LRU page replacement algorithm for the reference string.***
7 0 1 2 0 3 0 4 2 3 10 3. ***[R.T.U. 2017]***

Ans: 4

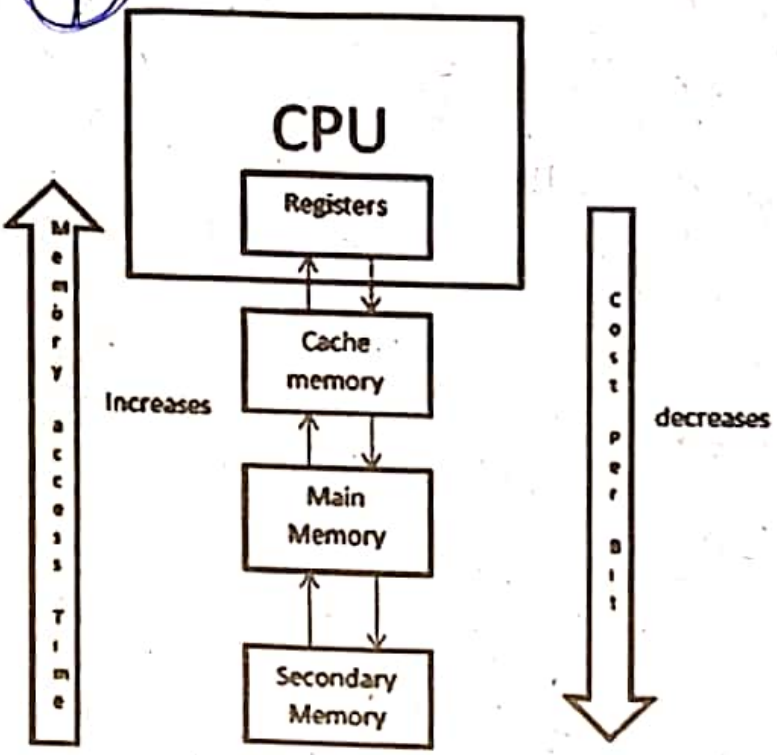


Fig.

1. Registers: CPU registers are at the top most level of this hierarchy, they hold the most frequently used data. They are very limited in number and are the fastest.They are often used by the CPU and the ALU for performing arithmetic and logical operations, for temporary storage of data.

2. Cache: The very next level consists of small, fast cache memories near the CPU. They act as staging areas for a subset of the data and instructions stored in the relatively slow main memory.

There are often two or more levels of cache as well. The cache at the top most level after the registers is the primary cache. Others are secondary caches.Many a times there is cache present on board with the CPU along with other levels that are outside the chip.

3. Main Memory: The next level is the main memory, it stages data stored on large, slow disks often called hard disks. These hard disks are also called secondary memory, which are the last level in the hierarchy. The main memory is also called primary memory.

The secondary memory often serves as staging areas for data stored on the disks or tapes of other machines connected by networks.

Q.7 ***Explain the concept of spooling with all its types and its advantage and disadvantages.***

[R.T.U. 2018, 2016]

Ans. Spooling : Spooling is an acronym for simultaneous peripheral operation On line. When the job requests the printer to output a line, that line is copied into a system buffer and is written to the disk. When the job is completed, the output is actually printed. This form of processing is called spooling.

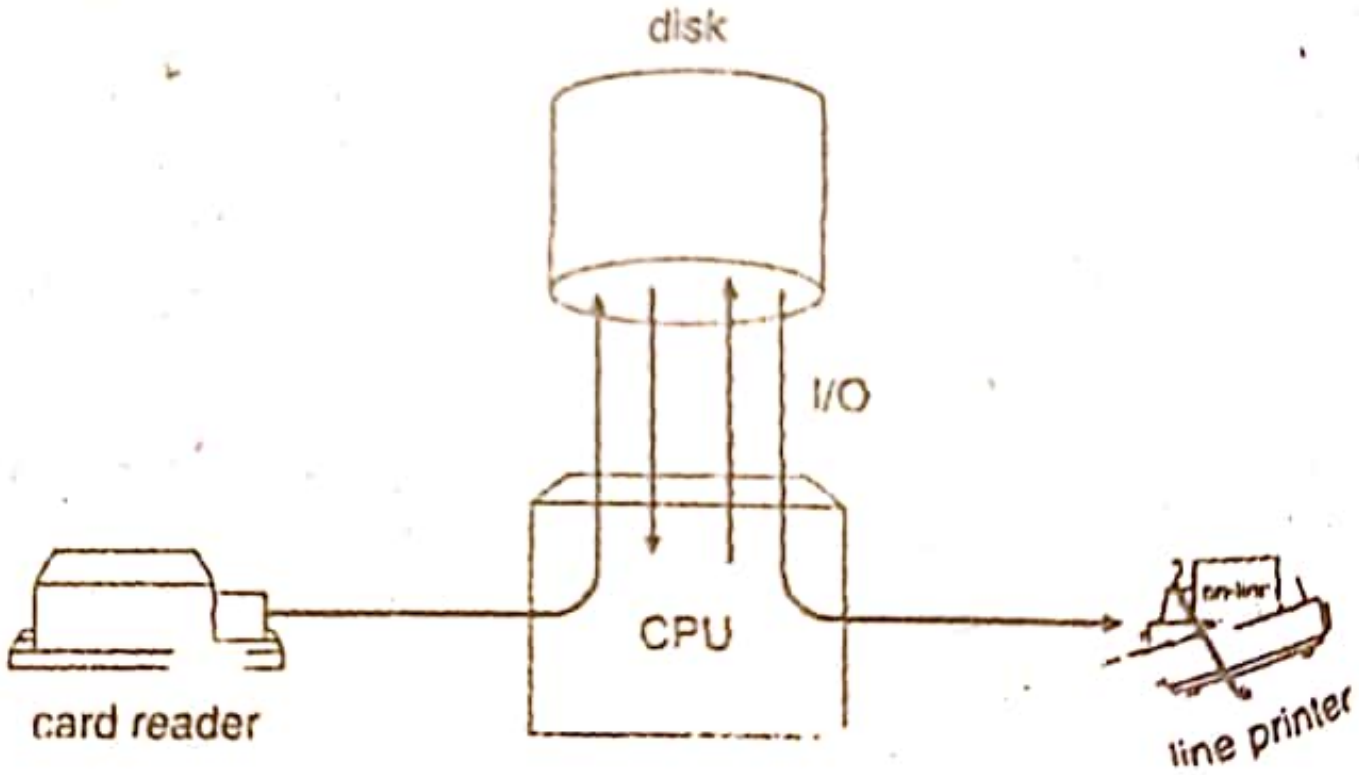


Fig.

Spooling, in essence, uses the disk as a huge buffer for reading as far ahead as possible on input devices and for storing output files until the output devices are able to accept them.

Spooling is also used for processing data at remote sites. The CPU sends the data via communication paths to a remote printer (or accepts an entire input job from a remote card reader). The remote processing is done at its own speed, with no CPU intervention. The CPU just needs to be notified when the processing is completed, so that it can spool the next thatch of data.

Spooling overlaps the I/O of one job with the computation of other jobs. Even in simple system, the spooler may be reading the input of one job while printing the output of a different job. During this time, still another job (or other jobs) may be executed, reading its "cards" from disk and "printing" its output lines onto the disk.

Spooling has a direct beneficial effect on the performance of the system. For the cost of some disk space and a few tables, the computation of one job can overlap with the I/O of other jobs. Thus, spooling can keep both the CPU and the I/O devices working at much higher rates. Spooling leads naturally to multiprogramming which is the foundation of all modern operating systems

Types of Spoolers

1. Print Spooler

A software program is responsible for managing all the current printing jobs which are sent to the computer printer or print server. The print spooler program may allow a user to delete a print job being processed or otherwise manage the print jobs currently waiting to be printed.

2. The System V-style Spooler

The system V-style spooler of printing subsystem uses system V-Release 4 commands, queues, and files and is administered the same way. Typical user commands available to the system V-style spooler are:

- **lp :** the user command to print a document
- **lpstat :** shows the current print queue
- **cancel :** deletes a job from the print queue
- **lpadmin :** a system administration command that configures the print system
- **lpmove :** a system administration command that moves jobs between print queues

3. The Berkeley-style Spooler

The Berkeley-style spoolers one of several standard architectures for printing on the Unix platform. It originated in 2.10BSD, and is used in BSD derivatives such as FreeBSD, NetBSD, OpenBSD, and DragonFly BSD. A system running this print architecture could traditionally be identified by the use of the user command lpr as the primary interface to the print system, as opposed to the system V-style lp command. Typical user commands available to the Berkeley-style spooler are:

- **lpr :** the user command to print
- **lpq :** shows the current print queue
- **lprm :** deletes a job from the print queue

4. CUPS-based Spooler

CUPS (Common UNIX Printing System) is a new type of spooler. It was designed to work across most UNIX and Linux-based system. It is also standards based. It enables printing through RFC1179 (1pr), IPP, CIFS/SMB, Raw socket (JetDirect), and through local printing. CUPS uses network printer browsing and Postscript Printer Description (PPD) files to ease the common tasks of printing.

Advantages of Spooling

The advantages of spooling are as follows:

(i) The spooling operation uses a disk as a very large buffer.

(ii) Spooling is capable of overlapping I/O operation for one job with processor operations for another job.

(iii) Processes are not suspended for a long time.

(iv) It can produce multiple copies of the output without running the process again.

Disadvantages of Spooling

The disadvantages of spooling are as follows:

(i) Need large amounts of disk space.

(ii) Increase disk traffic.

(iii) Not practical for real-time environment, because results are produce at a later time.

Q.8 ***Explain global versus local allocation.***

[R.T.U. 2018, Dec. 2013]

Ans. Global Versus Local Allocation : An important factor in the way frames are allocated to the various processes is page replacement. With multiple processes competing for frames, we can classify page-replacement algorithms into two broad categories: **global replacement and local replacement.** Global replacement allows a process to select a replacement frame from the set of all frames, even if that frame is currently allocated to some other process; that is, one process can take a frame from another. Local replacement requires that each process select from only its own set of allocated frames.

PART-B

Q.6 ***What are the various access methods for file system?*** *[R.T.U. 2017, Dec. 2013]*

Ans. File Access Methods

Files can be categorized broadly into two types depending upon their access methods.

1. Sequential Access Files : Sequential files are those, which are read sequentially (one record, then another and so on) in an order, starting at the beginning to the end of the file. Sequential files can be rewound so that they can be read as often as needed. However, records in such files cannot be read out of order e.g. reading of 34th record followed by 5th record and then 1st record is not possible with sequential files. Sequential files are convenient when the storage media is serial access e.g. magnetic tape rather than direct-access e.g. magnetic disk.

2. Random Access Files : Files whose records can be read in any order are called random access files. Random access files are basically a collection of records stored on a disk and these records are numbered 1, 2, 3 and so on and therefore, can be referred to by their numbers instead of their position. Such files should be necessarily be stored on direct access media like disk.

Random access files are essential for many applications, for example, database systems.

In a banking application, a customer may want to look up his current balance. This can be easily done by locating this customer's record using his account number as a key, rather than sequentially reading the records for thousands of other customers before this customer's record is located and read.

It is worth mentioning here that not all operating systems support both sequential and random access for files. Some systems allow only sequential file access, others allow only random access. Some systems require that a file should be defined as sequential or random when it is created, so that it can be accessed in the way it has been declared.

Q.7 ***Define file system? Explain file operations in detail?*** *[R.T.U. 2016]*

Ans. File System : A file system is a method of organizing files on physical media, such as hard disks, CD's, and flash drives. In the Microsoft Windows family of operating systems, users are presented with several different choices of file systems when formatting such media. These choices depend on the type of media involved and the situations in which the media is being formatted. Common file systems in Windows are as follows :

- NTFS
- exFAT
- EXT
- FAT
- HFS +

File Operations :

A file is an abstract data type. Operation on file, operating system provides system calls for creating, deleting, read etc. Basic operations on files are :

1. Create a file
2. Writing a file
3. Reading a file
4. Delete a file
5. Truncating a file
6. Repositioning within a file

1. Create a file : For creating a file, address space in the file system is required. After creating a file, entry of the file is made in the directory. The directory entry records the name of the file and the location in the file system.

2. Writing a file : System call is used for writing into file. It is required to specify the name of the file and information to be written to the file. According to the file name, system will search the name in the directory to find the location of the file.

3. Reading a file : To read a file, system call is used. It requires the name of file and memory address. Again the directory is searched for the associated directory entry and the system needs to keep a read pointer to the location in the file where the next read is to take place.

4. Delete a file : System will search the directory, which file to be deleted. If directory entry is found, it releases all file space. That free space can be reused by another (user) files.

5. Truncating a file : *Refer to Q.4.*

6. Repositioning within a file : The directory is searched for the appropriate entry, and the current file position is set to a given value. Repositioning within a file does not need to involve any actual I/O. This file operation is also known as file seek.

Q.8 ***Explain various features of file system of linux.*** *[R.T.U. 2015]*

Ans. In Linux, everything is configured as a file. This includes not only text files, images and compiled programs (also referred to as executables), but also directories, partitions and hardware device drivers.

Each filesystem contains a control block, which holds information about that filesystem. The other blocks in the filesystem are inodes, which contain information about individual files and data blocks, which contain the information stored in the individual files.

There is a substantial difference between the way the user sees the Linux filesystem and the way the kernel (the core of a Linux system) actually stores the files. To the user, the filesystem appears as a hierarchical arrangement of directories that contain files and other directories (i.e., subdirectories). Directories and files are identified by their names. This hierarchy starts from a single directory called root, which is represented by a "/" (forward slash).

The Filesystem Hierarchy Standard (FHS) defines the main directories and their contents in Linux and other Unix-like operating systems. All files and directories appear under the root directory, even if they are stored on different physical devices (e.g., on different disks or on different computers). A few of the directories defined by the FHS are /bin (command binaries for all users), /boot (boot loader files such as the kernel), /home (users home directories), /mnt (for mounting a CD-ROM or floppy disk),

PART-B

Q.9 *WSN on process scheduling in LINUX operating system.* *[R.T.U. 2016]*

Ans. Process Scheduling

Linux considers processes as belonging to either of two main categories: realtime and others. The terms real-timer is a bit of misnomer, since it does not necessarily have to do anything with real-time activities. Instead, this term refers to more important or more urgent processes. Based on this idea, Linux process scheduling can be classified into three categories, as shown in fig. 1.

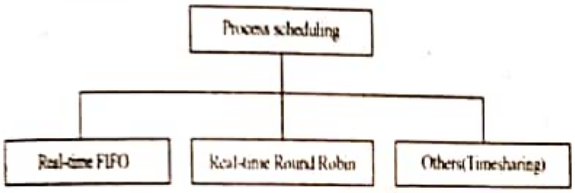


Fig. 1 : Process scheduling in Linux

The broad-level characteristics of the three scheduling types are as follows :

Real-time FIFO : This is the category which contains processes with the highest priority. These processes cannot be preempted. The only ways to preempt a processes in this category is via a new real-time FIFO process.

Real-time Round Robin: These processes are quite similar to the real-time FIFO processes, except that the CPU clock can preempt them.

Others (Timesharing): These are ordinary processes, which do not have any urgency and are scheduled by using the default timesharing algorithms.

The mechanisms used by Linux to schedule processes is quite interesting. Each process has a scheduling priority. The default value of this field is 20. This value for a process can be altered by using the "nice" system call. The syntax for "nice" is quite simple, as follows:

nice (value);

When a call to the "nice" function is made, the new scheduling priority of the thread becomes equal to:

Old scheduling priority-Value

The scheduling priority can be between 1 and 40, whereas the value can be between-20 and +19. The higher the value of the scheduling priority, the higher is the attention provided by the operating system to the process (*i.e.* more CPU time, faster response time, etc).

Each process also has another value, called quantum, associated with it. This is equal to the number of CPU clock ticks. The default clock assumed by Linux is of 100 Hz. Therefore, one tick = 100 milliseconds (ms). Each tick is called as jiffy in Linux terminology.

Based on all these concepts, the scheduler calculates the value of the goodness of a process as illustrated in fig.2. Linux always selects the process to be scheduled next, based on the value of the goodness value is scheduled.

With every clock tick, Linux reduces the value of the quantum for that process by 1. A process stops executing, if one of the following happens:

(a) The value of the quantum becomes 0, *i.e.* the time slice is over.

(b) The process needs some I/O and therefore, cannot continue.

(c) A previously blocked process with a goodness value greater than the currently executing process becomes ready.

The scheduler penodically resets the quantum of all the processes to a value based on the following formula:

Quantum (Quantum/2) + Scheduling priority

Process that have been blocked because of I/O would usually have some quantum left. Whereas the processes that have exhausted their full quantum (i.e. the ones, which are more CPU-hungry) will have the value of quantum closer to 0. In order to ensure that the CPU-hungry processes do not continue grabbing the CPU more and instead, the processes that were blocked because of I/O but are now ready, get a higher priority, this formula is devised. Let us understand how it works.

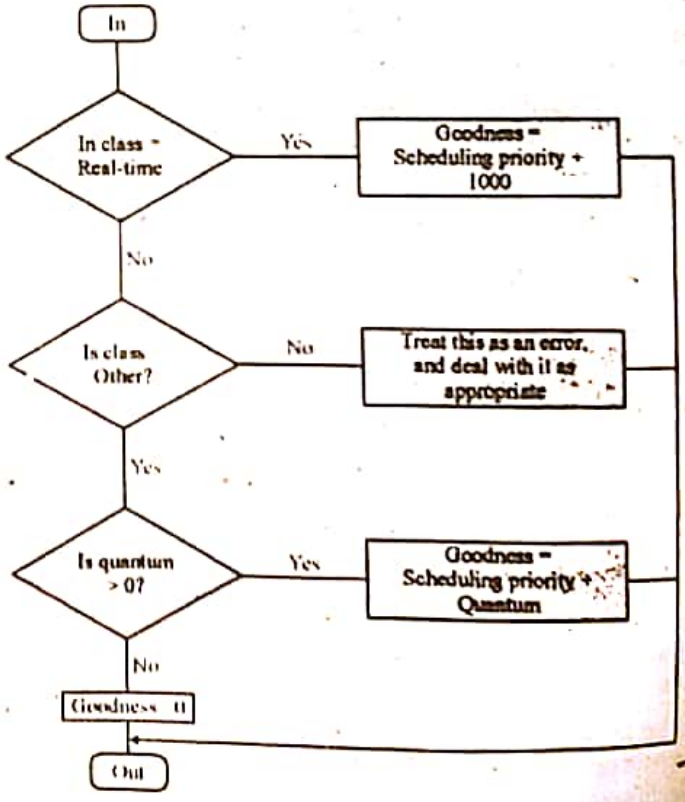


Fig.2 : Calculation of Goodness

As we have noted, for a CPU-hungry process, quantum will be closer to 0. Therefore, based on the above formula, the new quantum will be closer to its scheduling priority. On the other hand, for processes that were blocked due to I/O, but are now ready, the new quantum will also consider their old quantum

Note that by using such a mechanism, Linux automatically makes sure that the processes that are using the CPU extensively would now get a lower share of the CPU.

For inter-process synchronization, Linux implements wait queues and semaphores. Wait queues are circular linked list, which describe the process descriptors. In Linux, a semaphore consists of three fields: semaphore counter, number of waiting processes and list of processes waiting for the semaphore.

Q.10 *Explain the network file system of LINUX in detail.* *[R.T.U. 2016]*

Ans. The network file system of Linux : NFS views a set of interconnected workstations as a set of independent machines with independent file systems. The goal is to allow some degree of sharing among these file systems (on explicit request) in a transparent manner. Sharing is based on a client-server relationship. A machine may be, and often is, both a client and a server. Sharing is allowed between any pair of machines. To ensure machine independence, sharing of a remote file system affects only the client machine and no other machine.

One of the design goals of NFS was to operate in a heterogeneous environment of different machines, operating systems, and network architectures. The NFS specification is independent of these media and thus encourages other implementations. This independence is achieved through the use of RPC primitives built on top of an external data representation (XDR) protocol used between two implementation-independent interfaces. Hence, if the system consists of heterogeneous machines and file systems that are properly interfaced to NFS, file systems of different types can be mounted both locally and remotely.

The NFS specification distinguishes between the services provided by a mount mechanism and the actual remote-file-access services. Accordingly, two separate protocols are specified for these services a mount protocol and a protocol for remote file accesses, the NFS protocol. The protocols are specified as sets of RPCs. These RPCs are the building blocks used to implement transparent remote file access.

The NFS Protocol : *Refer to Q.2.*

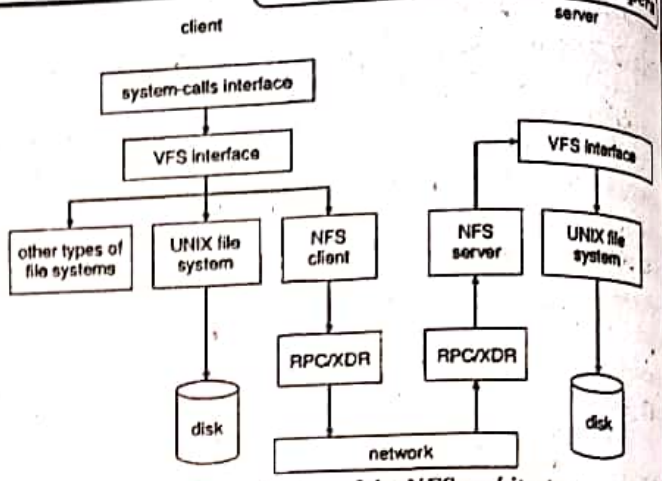


Fig. : Schematic view of the NFS architecture

Q.11 *Write short note on security system of LINUX O.S.* *[R.T.U. 2016]*

Ans. Security System of LINUX OS : Linux's security model is closely related to typical UNIX security mechanisms. The security concerns can be classified in two groups:

1. Authentication : Making sure that nobody can access the system without first proving that she has entry rights

2. Access control : Providing a mechanism for checking whether a user has the right to access a certain object and preventing access to objects as required.

1. Authentication

A new security mechanism has been developed by UNIX vendors to address authentication problems. The pluggable authentication module (PAM) system is based on a shared library that can be used by any system component that needs to authenticate users. An implementation of this system is available under Linux. PAM allows authentication modules to be loaded on demand as specified in a system-wide configuration file. If a new authentication mechanism is added at a later date, it can be added to the configuration file, and all system components will immediately be able to take advantage of it. PAM modules can specify authentication methods, account restrictions, session setup functions, and password-changing functions (so that, when users change their passwords, all the necessary authentication mechanisms can be updated at once).

2. Access Control

Access control under UNIX systems, including Linux, is performed through the use of unique numeric identifiers. A user identifier (UID) identifies a single user or a single set of access rights. A group identifier (GID) is an extra identifier that can be used to identify tights belonging to more than one user.

Access control is applied to various objects in the system. Every file available in the system is protected by the standard access-control mechanism. In addition, other shared

available, ... threaded application may be split amongst available processors.

Context Switching in Processes : *Refer to Q.11.*

Context Switching in Threads : *Refer to Q.11*

PART-C

Q.18 How an operating system works as a resource manager and virtual machine?

[R.T.U. 2018, 2014]

[Note : Vertical actually should be taken as virtual.]

Ans. Operating System : An operating system (OS) is a software that manages the computer hardware and provide an environment in which a user can execute programs in a convenient and efficient manner.

An operating system acts as an intermediary between the user of a computer and the computer hardware as shown in below fig. 1 :

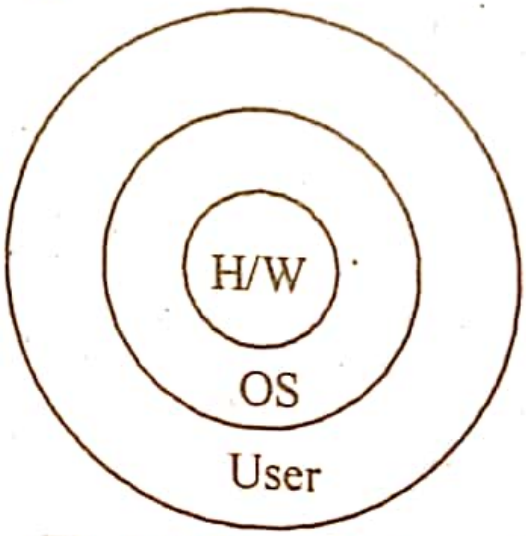


Fig. 1 : OS as an intermediary

Operating System as a Resource Manager

The concept of the operating system as primarily providing its users with a convenient interface is a top down view. An alternative, bottom - up view, holds that the operating system is there to manage all the pieces of a complex system. Modern computer consists of processors, memories, timers, disks, mice, network interfaces, printers and a wide variety of other devices. In the alternative view, the job of the operating system is to provide for an orderly and controlled allocation of the processor, memories and I/O devices among the various programs competing for them.

When a computer (or network) has multiple users, the need for managing and protecting the memory, I/O devices and other resources is even greater, since the users might otherwise interface with one another. In addition, users often need to share not only hardware, but information (files, database, etc.) as well. In short, this view of the operating system holds that its primary task is to keep track of who is using which resource, to grant resource requests, so account for usage and to mediate conflicting requests from different programs and users.

Resource management includes *multiplexing (Sharing)* resources in two ways: in time and in space. When a resource is time multiplexed, different programs or users take turns using it. First one of them gets to use the resource, then another and so on. For example, with only one CPU and multiple programs that want to run on it, the operating system first allocates the CPU to one program, then after it has run long enough, another one gets to use the CPU then another and then eventually the first one again. Determining how the resource is time multiplexed who goes next and for how long is the task of the operating system. Another example of time multiplexing is sharing the printer. When multiple print jobs are queued up for printing on a single printer, a decision has to be made about which one is to be printed next.

The other kind of multiplexing is space multiplexing. Instead of the customer taking turns, each gets part of the resource. For example, main memory is normally divided up among several running programs, so each one can be resident at the same time (for example, in order to take turns using the CPU). Assuming there is enough memory to hold multiple programs, it is more efficient to hold several programs in memory at once rather than give one of them all of it, especially if it only needs a small fraction of the total. Of course, this raises issues of fairness, protection and so on and it is up to the operating system to solve them. Another resource that is space multiplexed is the (hard) disk. In many systems a single disk can hold files from many users at the same time. Allocating disk space and keeping track of who is using which disk blocks is a typical operating system resource management disk.

Operating System as Virtual Machines

Virtual-memory techniques as, an operating system can create the illusion that a process has its own processor with its own (virtual) memory. Of course, normally, the process has additional features, such as system calls and a file system, that are not provided by the bare hardware. The virtual-machine approach, on the other hand, does not provide any additional functionality, but rather provides an interface that is identical to the underlaying bare hardware. Each process is provided with a (virtual) copy of the underlying computer (fig. 2).

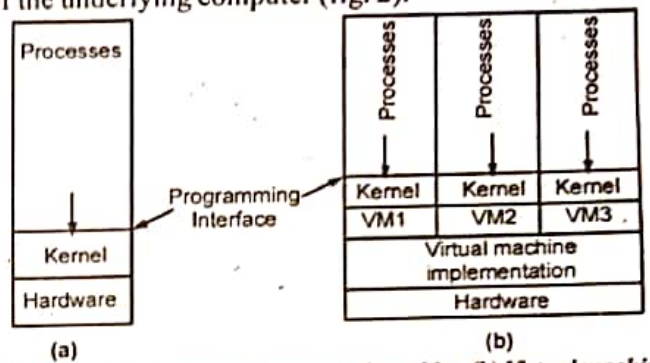


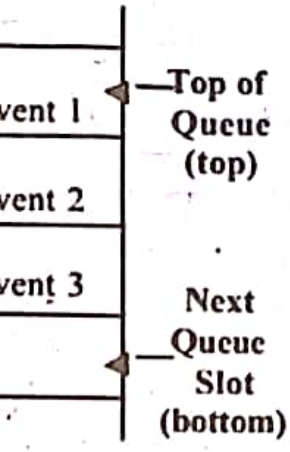
Fig. 2 : System models (a) Non-virtual machine (b) Virtual machine

The physical computer shares resources to create the virtual machines. CPU scheduling can share out the CPU to create the appearance that users have their own processors. Spooling and a file system can provide virtual card readers and virtual line printers. A normal user time-sharing terminal provides the function of the virtual-machine operator's console.

A major difficulty with the virtual-machine approach involves disk systems. Suppose that the physical machine has three disk drives but wants to support seven virtual machines. It cannot allocate a disk drive to each virtual machine. The virtual-machine software itself will need substantial disk space to provide virtual memory. The solution is to provide virtual disks, which are identical in all respects except size - termed minidisks in IBM's VM operating system. The system implements each minidisk by allocating as many tracks on the physical disks as the minidisk needs. Obviously, the sum of the sizes of all minidisk must be smaller than the size of the physical disk space available.

Users thus are given their own virtual machines. They can then run any of the operating systems or software packages that are available on the underlying machine. For the IBM VM system, a user normally runs CMS-a single-user interactive operating system. The virtual machine software is concerned with multiprogramming multiple virtual machines onto a physical machine, but it

at when one process
n its critical section,
xecute in its critical
ical sections by the
me.



conditons?
med critical section,

use

that allows multiple
esource, such as file
a program is started
me. After this stage,
nust lock the mutex
resource. The mutex
onger needed or the

inary semaphore and
and semaphore are

ing mechanism used
Only one task (can
OS abstraction) can
wnership associated

triggered upon which an "Interrupt Service Routine (ISR)". signals the call processing task to wakeup.

Q.23 What is operating system? Explain its types and services provided by operating system in detail.
[R.T.U. 2017]

OR

What are the different services provided by the operating system? Explain all of them in detail?
[R.T.U. 2016]

Ans. Operating System : *Refer to Q.18.*

Types of Operating Systems : Operating system can be classified into following categories:

(i) Single-user Single-tasking Operating system
(ii) Batch Operating system
(iii) Multi-user Operating system
(iv) Multi-tasking Operating system
(v) Real-time Operating system
(vi) Network Operating system
(vii) Distributed Operating system

(i) Single-user Single-tasking Operating System: An operating system that allows a single user to work on a computer at a time and can execute a single job at a time is known as singe-user single-tasking operating system. For example, MS-DOS is a single-user single-tasking operating system because you can open and run only one application at a time in MS-DOS.

(ii) Batch Operating System: A single user operating system that can execute various types of jobs in batches but one after another. In a batch-operating environment, users submit their programs and data to the operator and the operator groups the similar jobs, and then loads them (all groups of programs along with their relevant data) simultaneously. When the execution of one program for performing a similar kind of jobs is over, a new program is loaded for the execution by the operating system. Batch operating system is suitable for such applications that require long computation time without user intervention. Some examples of such applications are payroll processing, forecasting, statistical analysis, etc.

(iii) Multi-user Operating System: It permits simultaneous access to a computer system through two or more terminals for users. UNIX is an example of multi-

user operating system. It allows two or more users to run programs at the same time. Some operating systems permit hundreds or even thousands of concurrent users.

(iv) Multi-tasking Operating System : It is also called Multiprocessing Operating system. A multitasking operating system is able to handle more than one processor as the jobs have to be executed on more than one processor (CPUs). The running state of program is called a process or task. A multitasking operating system supports two or more processes to execute simultaneously.

A multiuser operating system allows simultaneous access to a computer system through one or more terminals. Although frequency associated with multiprogramming, multiuser operating system does not imply multiprogramming or multitasking. A dedicated transaction processing system such as railway reservation system that hundreds of terminals under control of a single program is an example of multiuser operating system. Window 98/2000/XP/Vista, OS/2, UNIX, LINUX etc. are examples of multi-tasking operating system.

Time Sharing System : Time sharing is a processor (CPU) management technique. In a time sharing operating system, the CPU is allocated to each user, in sequence, for a small amount of time called time-slice (from 10-100 milliseconds). A time slice is allocated to each user using round-robin scheduling algorithm. As soon as the time slice is over, the CPU is allocated to the next user.

Time sharing system is a form of multiprogrammed operating system which operates is an interactive mode with a quick response time. The user types a request to the computer through a keyboard. The computer processes it and a response (if any) is displayed on the user's terminal. A time sharing system allows many users to simultaneously share the computer resources. Since each action or command in a time-sharing system takes a very small fraction of time, only a little CPU time is needed for each user. As the CPU switches rapidly from one user to another user, each user is given impression that he has his own computer while it is actually one computer shared among many users. Most time sharing systems use time-slice (round robin) scheduling of CPU. In this approach, programs are executed with rotating priority that increases during waiting and drops after the service is granted. In order to prevent a program from monopolizing the processor, a program executing longer than the system defined time-slice is interrupted by the operating system and placed at the end of the queue of waiting program.

Memory management in time sharing system provides for the protection and separation of user programs. Input/ Output management feature of time-sharing system must be able to handle multiple users (terminals). However the processing of terminals interrupts is not time critical due to the relative slow speed of terminals and users. As required by most multiuser environment allocation and deallocation of device must be perfomed in a manner that preserves integrity and provides for good performance.

(v) Real-time Operating System: A real-time operating system (RTOS) is a multitasking operating system intended for real-time applications. Such applications include embedded systems (programmable thermostats, household appliance controllers), industrial robots, spacecraft industrial control and scientific research equipment.

A RTOS facilitates the creation of a real-time system, but does not guarantee the final result will be real-time; this requires correct development of the software. The primary objective of real-time system is to provide quick response time. Resource utilisation and user convenience are of lesser concern to real-time system. In order to provide quick response time, most of the time processing remain in primary memory. If a job is not completed within the fixed deadline, this situation is called deadline overrun. A real time operating system must minimise the possible deadline overruns.

An early example of a large-scale real-time operating system was Transaction Processing Facility developed by American Airlines and IBM for the Sabre Airline Reservations System.

(vi) Network Operating System (NOS): A network operating system (NOS) is an operating system which makes it possible for computers to be on a network, and manages the different aspects of the network. Network operating system (NOSs) are designed to support interconnection and communication among computers. Network Operating system provides support for communication, network management functions and administration, multi-user operations and security. Novel's Net ware, Microsoft's Windows NT, UNIX and Linux are examples for network operating sytem. Some examples are Windows for Workgroups, Windows NT, AppleShare, DEDnet, LAN taste, etc.

(vii) Distributed Operating System: A distributed operating system is one that looks to its users like an ordinary centralised operating system but runs on multiple independent CPUs. The use of multiple processors is invisible to the user. In a true distributed system, users are not aware of where their programs are being run or where their files are residing; they should all be handled automatically and efficiently by the operating system.

Distributed operating systems have many aspects in common with centralised ones but they also differ in certain ways. Distributed operating system, for example often allows programs to run on several processors at the same time, thus requiring more complex processor scheduling (scheduling refers to a set of policies and mechanisms built into the operating systems that controls the order in which the work to be done is completed, algorithms in order to achieve maximum utilisation of CPU's time).

Fault-tolerance is another area in which distributed operating systems are different. Distributed systems are considered to be more reliable than uniprocessor based system. They perform even if certain part of the hardware is malfunctioning. This additional feature, supported by distributed operating system has enormous implications for the operating system.

Operating System Services

Operating system provides certain services to programs and to users of those programs. These services are useful for the users for successful execution of programs and efficient utilization of resources. Series of operating system may differ from one operating system to another. Some common types of services are as follows:

1. **User Interface :** User interface (UI) may be command line interface (CLI), which uses text commands or graphical user interface (GUI), which uses menus or icons.

2. **Operating System as a Resource Manager:** Operating system provides certain services to the users and their program. Almost all operating system provides common types of services. These services are useful for the users for successful execution of programs and efficient utilization of resources.

Common Types of Services

1. **Execution of Program :** This is the first and foremost service that is provided by the operating system to the users. The computing system must execute and terminate all user programs successfully by loading them into main memory. A proper error message must be displayed for the incomplete or abnormally terminated programs.

2. **Input/Output Operations :** During normal execution of a program user may be required to input data or may need some processed data as output. It is the job of the operating system to obtain data from the input device or from some data files and send the output to a data file or output device.

3. **File Management :** The user of the computing system may definitely need to create the files on secondary storage to record their work permanently for future use or delete the file that is already in existence. The operating system must provide a mechanism that carefully handles all these file manipulation operations on the respective devices.

4. **Communication :** In the computing system there can be more number of programs that need to share information among them. The program may be running on a single machine or may be run on several machines. This phenomenon is known as 'Inter Process Communication'. To maintain data security and integrity, an operating system must handle inter process communication.

5. **Error Detection :** If any resource or device in a computing system comes across any error it should be promptly handled by the operating system. Invalid or improper use of devices by the programs must be reported. The most common types of errors are: memory allocation error, divide by zero error, printer error, power off error etc.

6. **Protection :** The operating system should provide the services to all users by ensuring the security of access by authenticated users to some important files and data.

7. **Accounting :** In case of multiple users of the computing system the information of duration and number of resources that are used by each user may be tracked by the operating system for accounting purpose.

8. **Resource Sharing and Allocation :** When there are more number of users in the computing system or there are many programs competing for limited resources, then the optimal allocation and de-allocation of resource is the responsibility of operating system to ensure increased throughput.

Q.24 ***In-connection with interprocess communication explain the following :***

(i) Race Condition

(ii) Critical Condition

(iii) Sleep and Wake up

(iv) Sleeping Barber's Problem ***[R.T.U. 2016]***

Ans.(i) Race Condition : *Refer to Q.22(ii).*

(ii) Critical Condition Or Critical Section: *Refer to Q.22(iii).*

this type of paging scheme, all the pages of a particular job must be in core before that program can be executed. And third, a considerable amount of hardware support is needed.

Demand Paging : All these disadvantages are overcomed by demand paging. In demand paging a program can be executed without all pages being in core, *i.e.* pages are fetched into core as they are needed. By choosing sufficiently small intervals of time, demand paging can be very valuable in "time sharing" systems which attempt to run dozens of programs simultaneously. However, the problem associated with demand paging is thrashing.

Paged Segmentation : An appealing idea would be to combine the best features of the previously presented techniques. Segmentation can be used to facilitate sharing and protection paging can be used to resolve the fragmentation and recompacting problems and demand paging eliminates the restriction on address space size. Some computer systems combine the two approaches in order to enjoy the benefits of both. One popular approach is to use segmentation from the users point of view but to divide each segment into pages of fixed size for purposes of allocation. In this way, the combined system retains most of the advantages of segmentation. At the same time, the problems of complex segment placement and management of secondary memory are eliminated by using paging.

Q.16 ***What is thrashing? What do you understand by degree of multiprogramming.*** ***[R.T.U. 2017]***

OR

Explain Thrashing. ***[R.T.U. 2018, Dec. 2013]***

Ans. Thrashing : If the number of frames allocated to a low-priority process falls below the minimum number required by the computer architecture, we must suspend that process execution. We should then page out its remaining pages, freeing all its allocated frames. This provision introduces a swap-in, swap-out level of

intermediate CPU scheduling.

In fact, look at any process that does not have "enough" frames. Although it is technically possible to reduce the number of allocated frames to the minimum, there is some (larger) number of pages in active use. If the process does not have this number of frames, it will quickly page fault. At this point, it must replace some page. However, since all its pages are in active use, it must replace a page that will be needed again right away. Consequently, it quickly faults again and again. The process continues to fault, replacing pages for which it then faults and brings back in right away.

This high paging activity is called **thrashing**. A process is thrashing if it is spending more time paging than executing.

Cause of Thrashing : Thrashing results in server performance problems. Consider the following scenario, which is based on the actual behavior of early paging systems.

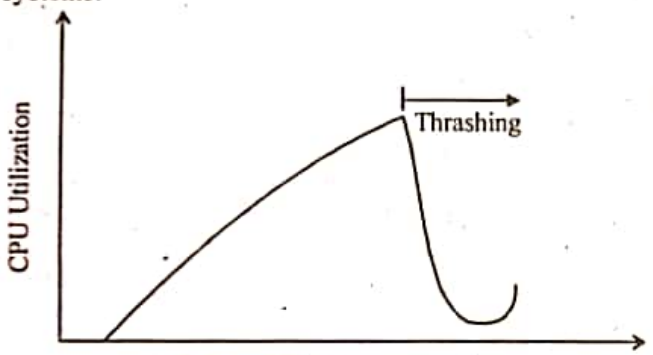


Fig.

The operating system monitors CPU utilization. If CPU utilization is too low, we increase the degree of multiprogramming by introducing a new process to the system. A global page-replacement algorithm is used; it replaces pages without regard to the process to which they belong. Now suppose that a process enters a new phase in its execution and needs more frames. It starts faulting and taking frames away from other processes. These processes need those pages, however and so they also fault, taking frames from other processes. These faulting processes must use the paging device to swap pages in and out. As they queue up for the paging device, the ready queue empties. As processes wait for the paging device, CPU utilization decreases.

The CPU scheduler sees the decreasing CPU utilization and increases the degree of multiprogramming as a result. The new process tries to get started by taking frames from running processes, causing more page faults and a longer queue for the paging device. As a result, CPU utilization drops even further and the CPU scheduler tries to increase the degree of multiprogramming even more. Thrashing has occurred and system throughput plunges. The page-fault rate increases tremendously. As a result, the effective memory access time increases. No work is getting done, because the processes are spending all their time paging.

This phenomenon is illustrated in figure in which CPU utilization is plotted against the degree of multiprogramming. As the degree of multiprogramming increases, CPU utilization also increases, although more slowly, until a maximum is reached. If the degree of multiprogramming is increased even further, thrashing sets in and CPU utilization drops sharply. At this point, to increase CPU utilization and stop thrashing, we must decrease the degree of multiprogramming.

Thrashing Avoidance : We can limit the effects of thrashing by using a **local replacement algorithm** (or **priority replacement algorithm**). With local replacement, if one process starts thrashing, it cannot steal frames from another process and it the latter to thrash also. Pages are replaced with regard to the process of which they are a part. However, if processes are thrashing, they will be in the queue for the paging device most of the time. The average service time for a page fault will increase, due to the longer average queue for the paging device. Thus, the effective access time will increase even for a process that is not thrashing.

In a multiprogramming-capable system, jobs to be executed are loaded into a pool. Some of these jobs are loaded into main memory, and one is selected from the pool for execution by the CPU. If at some point, the program in progress terminates or requires the services of a peripheral device, the control of the CPU is given to the next job in the pool. As programs terminate, more jobs are loaded into memory for execution, and CPU control is switched to another job in memory. In this way the CPU is always executing some program or some portion thereof, instead of waiting for a printer, tape drive, or console input.

An important concept in multiprogramming is the degree of multiprogramming. The degree of multiprogramming describes the maximum number of processes that a single-processor system can accommodate efficiently. The primary factor affecting the degree of multiprogramming is the amount of memory available to be allocated to executing processes. If the amount of memory is too limited, the degree of multiprogramming will be limited because fewer processes will fit in memory. A factor inherent in the operating system itself is the means by which resources are allocated to processes. If the operating system cannot allocate resources to executing processes in a fair and orderly

fashion, the system will waste time in reallocation, or process execution could enter into a deadlock state as programs wait for allocated resources to be freed by other blocked processes. Other factors affecting the degree of multiprogramming are program I/O needs, program CPU needs, and memory and disk access speed.

Q.17 ***What you mean by paging? Explain the concept of demand paging with proper diagram.*** ***[R.T.U. 2017]***

OR

What is demand paging? ***[R.T.U. 2014]***

Ans. Paging : Paging is a memory management technique in which the memory is divided into fixed size pages. Paging is used for faster access to data. When a program needs a page, it is available in the main memory as the OS copies a certain number of pages from your storage device to main memory. Paging allows the physical address space of a process to be noncontiguous.

Demand Paging : *Refer to Q.15(i).*

Basic Concepts : When a process is to be swapped in, the pager guesses which pages will be used before the process is swapped out again. Instead of swapping in a whole process, the pager brings only those necessary

PART-C

Q.13 ***What is deadlock? What are necessary conditions for deadlock to occur?***

[R.T.U. 2018, 2013]

OR

What is deadlock? Explain the conditions and prevention of deadlock? ***[R.T.U. 2017]***

OR

What are the different deadlock prevention schemes? Explain. ***[R.T.U. 2015]***

OR

What is deadlock? What are the necessary conditions to occur the deadlock? What are the various methods to recover from the deadlock?

[R.T.U. 2014]

Ans. Introduction to Deadlocks : A set of processes is deadlocked if each process in the set is waiting for an event that can cause only by another process in the set.

Because all the processes are waiting, none of them will, even cause any of the events that could wake up any of the other members of the set and all the processes continue to wait forever. For this model, we assume that processes have only a single thread and that there are no interrupts possible to wake up a blocked process. The no-interrupts condition is needed to prevent an otherwise deadlocked process from being awakened by, say, an alarm and then causing events that release other processes in the set.

Conditions for Deadlock

Four essential conditions for deadlock to occur : Coffman et al. (1971) showed that four conditions must hold for these to be a deadlock :

1. Mutual exclusion condition : Each resource is either currently assigned to exactly one process or is available.

2. Hold and wait condition : Processes currently holding resource granted earlier can request new resources.

3. No preemption condition : Resources previously granted cannot be forcibly taken away from a process. They must be explicitly released by the process holding them.

4. Circular wait condition : There must be a circular chain of two or more processes, each of which is waiting for a resource held by the next member of chain.

All four of these conditions must be present for a deadlock to occur. If one of them is absent, no deadlock is possible.

Deadlock Prevention : Deadlock avoidance is essentially impossible because it requires information about future requests. Different deadlock prevention schemes are as follows :

(i) Attacking the mutual exclusion condition : If no resource were ever assigned exclusively to a single process, we would never have deadlocks. However it is equally clear that allowing two processes to write on the printer at the same time will lead to chaos.

By spooling printer output, several processes can generate output at the same time. In this model, the only process that actually requests the physical printer is the printer daemon. Since the daemon never requests any other resources, we can eliminate deadlock for the printer.

(ii) Attacking the hold and wait condition : The second conditions stated by Coffman et al looks slightly more promising. If we can prevent processes that hold resources from waiting for more resources, we can eliminate deadlocks. One way to achieve this goal is to require all processes to request all their resources before starting execution. If every thing is available, the process will be allocated whatever it needs and can run to completion. If one or more resources are busy, nothing will be allocated and the process would just wait.

An immediate problem with this approach is that many processes do not know how many resources they will need until they started running. In fact, if they knew, the Banker's algorithm could be used. Another problem is that the resources will not be used optimally with this approach. Nevertheless, some mainframe batch system requires the user to list all the resources on the first line of each job. The system then acquires all resources immediately and keeps them until the job finishes. While this method puts a burden on the programmer and wastes resources, it does prevent deadlocks.

(iii) Attacking the no preemption condition : Attacking the third condition (no preemption) is even less promising than attacking the second one. If a process has been assigned the printer and is in the middle of printing its output, forcibly taking away the printer because a needed plotter is not available is tricky at best and impossible at worst.

(iv) Attacking the circular wait condition : The circular wait can be eliminated in several ways. One way is simply to have a rule saying that a process is entitled only to a single resource at any moment. If it needs a second one, it must release the first one for a process that needs to copy a huge file from a tape to a printer, this restriction is unacceptable.

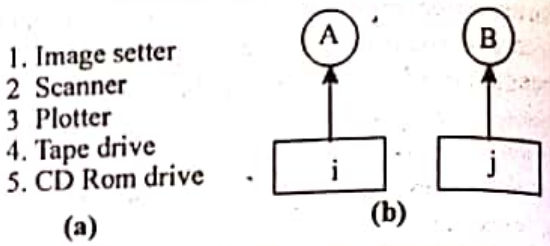


Fig. : (a) Numerically ordered resources (b) A resource graph

Another way to avoid the circular wait is to provide a global numbering of all the resources, as shown in Fig. Now the rule is this : processes can request resources whenever they want to, but all requests must be made in numerical order. A process may request first a printer and then a tape drive, but it may not request first a plotter and then printer.

The various approaches to deadlock prevention are :

Condition	Approach
Mutual exclusion	Spool everything
Hold and wait	Request all resources initially
No preemption	Take resource away
Circular wait	Order resources numerically

Recovery from deadlock

There are two options for breaking a deadlocks as follows :

(1) To abort one or more processes to break the circular wait i.e. **Process Termination.**

(2) To preempt some resources from one or more of the deadlock processes i.e. **Resource Preemption.**

1. Process Termination : To eliminate deadlocks by aborting process, we use one of two methods:

(i) Abort all deadlocked processes : This method clearly break the deadlock cycle but it is a very extensive method that the deadlocked processes may have computed for a long time, and the results of these partial computations must be discarded and probably will have to be recomputed later.

(ii) Abort one process at a time until the deadlock cycle is eliminated

The method incurs considerable overhead, since after each process is aborted, a deadlock-detection algorithm must be invoked to determine whether any processes are still deadlocked.

We should abort those processes whose termination will incur the minimum cost. Many factors may affect which process is chosen, including.

- How many processes will need to terminated.
- How long the process has computed.
- How much longer the process will compute.
- How many and what type of resources the process has used.
- How many more resources the process needs.
- Whether the process is interactive or batch.

2. Resource Preemption : To eliminate deadlocks using resource preemption, be successively preempt some resources from processes and give those resources to other processes until the deadlock cycle is broken, need to be addressed.

(i) Selecting a victim : Which resources and which processes are to be preempted?

(ii) Rollback : We must roll back the process to some safe state and restart it from the state. It is more effective to roll back the process only as far as necessary to break the deadlock rather then total rollback.

Starvation : How can we guarantee that resources will not always be preempted from the same process? It may happen that the same process is always picked as a victim. As a result, this process never completes its designated task. We must ensure that a process can be picked as a victim only a (small) finite number of times.

Q.14 ***Write and explain Bankers algorithm for deadlock avoidance.*** ***[R.T.U. 2018, 2015]***

OR

Explain Banker's Algorithm for dead lock avoidance with an example.

[R.T.U. 2016, 2012, 2011, Raj. Univ. 2007, 2005, 2003]

Ans. The Banker's Algorithm for a Single Resource : A scheduling algorithm that can avoid deadlocks is due to Dijkstra and is known as the banker's algorithm and is an extension of the deadlock detection algorithm. It is modeled on the way a small-town banker might deal with a group of customers to whom he has granted lines of credit. What the algorithm does is check to see if granting the request leads to an unsafe state. If it does, the request is denied, If granting the request leads to a safe state, it is carried out.

Has Max

HFS + has the following properties:

- Maximum volume is 8 EB (about 8 million TB).
- Files stored to HFS+ partitions can be as large as the partition.
- Windows users can read HFS+ but not write.
- Drivers are available that allow Linux users to read and write to HFS+ volumes.

The EXT file system

The extended file system was created to be used with the Linux kernel. EXT4 is the most recent version of EXT.

EXT4 has the following Properties:

- EXT4 can support volumes up to 1 EB.
- 16 TB maximum file size.
- Red Hat recommends using XFS (not EKT4) for volumes over 100 TB.
- EXT4 is backwards compatible with EXT2 and EXT3.
- EXT4 can pre-allocate disk space.
- By default, Windows and Mac OS cannot read EXT file systems.

Q.16 Define security and user authentication in file system.

Ans. Security and user Authentication in File System

Cryptography as a security tool

- Cryptography can be utilised as a tool for computer security.
- In a networked system, an operating system can never be absolutely sure about the identity of its communication partner.
- Cryptography can help here to remove the necessity to trust the network.
- Constraints the number of potential senders/ receivers of a message.
- Typically based on keys that are selectively distributed to computers in a network.

Encryption :

- Encryption constrains the possible receivers of a message.

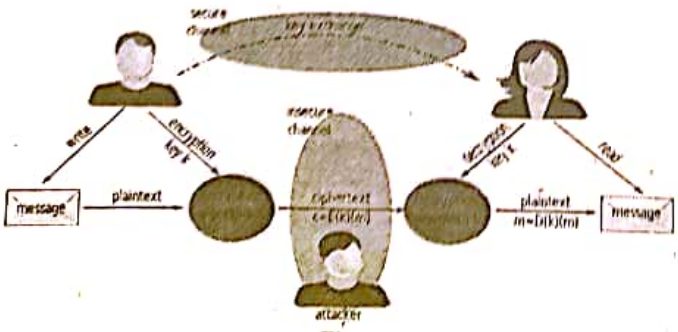


Fig.

Symmetric Encryption :

- The same key is used to encrypt and to decrypt.
- The secrecy of E(k) must be protected as well as D(k).
- If the same key is utilised for an extended amount of data, it becomes vulnerable to an attack.
- **Examples:** AES (advanced encryption standard), DES (data encryption standard).

Asymmetric Encryption :

- Different encryption and decryption keys are used.
- We distinguish between public and private keys.
- Uses one-way function for which the inverse operation is much harder to execute (e.g. factorisation).
- Exkurs: RSA Cryptosystems

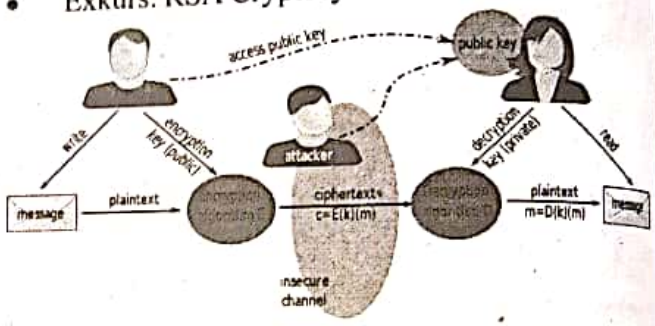


Fig.

Authentication

- Encryption constrains the amount of potential senders of a message.
- Constraining the amount of possible receivers of a message is called authentication.
- Authentication is complementary to encryption.
- Observe, that an encrypted message can also prove the identity of the sender.
- Authentication is also useful for proving that a message has not been modified.

Key Distribution

- With symmetric cryptography, the transmission of the key becomes a great challenge and possible vulnerability out-of-band transmission paper document, conversation, untypical transmission technique/band.
- Since this might be insecure and is possible also very elaborate, public key cryptosystems are utilised.
- Since public keys need not be secured, a key distribution center can be utilised.
- Vulnerable to man-in-the-middle-attacks.
- With trusted authorities the authentication of individual communication partners can then be verified.
- The system has also to authenticate the user.

- Typically based on one or more of the following.
  - The possession of something (key or card)
  - Knowledge of something (identifier and password)
  - Attribute of the user (fingerprint, retina pattern, signature)

Issues with Passwords

- How to keep the password stored and save in the system.
- In UNIX systems, encryption is used to secure the password.
- However, with brute force attacks, these passwords may be decoded in acceptable time off-line
- Therefore, the file containing the passwords is often visible to a superuser only.
- Programs that use the password run setuid to root in order to be allowed to read the file.
- Typically based on one or more of the following :
  - The possession of something (key or card)
  - Knowledge of something (identifier and password)
  - Attribute of the user (fingerprint, retina pattern, signature)

❑❑❑

Q.17 ***What is a UNIX Shell? Explain the steps that a UNIX shell follows while processing a command.***

Ans. The shell is a type of program called an interpreter. An interpreter operates in a simple loop: It accepts a command, interprets the command, executes the command, and then waits for another command. The shell displays a "prompt," to notify you that it is ready to accept your command.

After the command line is terminated by the key, the shell goes ahead with processing the command line in one or more passes. The sequence is well defined and assumes the following order.

Parsing: The shell first breaks up the command line into words, using spaces and the delimiters, unless quoted. All consecutive occurrences of a space or tab are replaced here with a single space.

Variable evaluation: All words preceded by a $ are evaluated as variables, unless quoted or escaped.

Command substitution: Any command surrounded by back quotes is executed by the shell which then replaces the standard output of the command into the command line.

Wild-card interpretation: The shell finally scans the command line for wild-cards (the characters *, ?,[,]). Any word containing a wild-card is replaced by a sorted list of filenames that match the pattern. The list of these filenames then forms the arguments to the command. PATH evaluation: It finally looks for the PATH variable to determine the sequence of directories it has to search in order to hunt for the command.

Q.18 ***What is Shell? What are some common shells and what are their indicators? Also write key features of the Korn Shell and describe the Shell's responsibilities.***

Ans. A shell acts as an interface between the user and the system. As a command interpreter, the shell takes commands and sets them up for execution.

Types of shell and their indicators

- sh - Bourne shell
- csh - C shell
- bash - Bourne again shell
- tcsh - enhanced C shell
- zsh - Z shell
- ksh - Korn Shell

Features of korn shell

- history mechanism with a built-in editor that simulates emacs or vi
- built-in integer arithmetic
- string manipulation capabilities
- command aliasing
- arrays
- job control

shell responsibilities :

- program execution
- variable and file name substitution
- I/O redirection
- pipeline hookup
- environment control
- interpreted programming language

PART-C

Q.19 ***Explain in detail Kernel structure of LINUX operating system.*** *[R.T.U. 2016]* (5)

Ans. Kernel Structure : The *kernel* is the central component of most Linux operating systems; it acts as a bridge between software applications and the actual data processing done at the hardware level. The kernel's responsibilities include managing the system's resources (the communication between hardware and software components).The Linux kernel is composed of five main subsystems :

1. Process Scheduler (SCHED) : *Refer to Q.1.*

2. Memory Manager (MM) permits multiple process to securely share the machine's main memory system. In addition, the memory manager supports virtual memory that allows Linux to support processes that use more memory than is available in the system. Unused memory is swapped out to persistent storage using the file system then swapped back in when it is needed.

3. Virtual File System (VFS) abstracts the details of the variety of hardware devices by presenting a common file interface to all devices. In addition, the VFS supports several file system formats that are compatible with other operating systems.

4. The Network Interface (NET) provides access to several networking standards and a variety of network hardware.

5. Inter-Process Communication (IPC) subsystem supports several mechanisms for process-to-process communication on a single Linux system.

Here the fig. shows a high-level decomposition of the Linux kernel, where lines are drawn from dependent subsystems to the subsystems they depend on :

This diagram emphasizes that the most central subsystem is the process scheduler; all other subsystems depend on the process scheduler since all subsystems need to suspend and resume processes. Usually a subsystem will suspend a

process that is waiting for a hardware operation to complete and resume the process when the operation is finished. For example, when a process attempts to send a message across the network, the network interface may need to suspend the process until the hardware has completed sending the message successfully. After the message has been sent (or the hardware returns a failure), the network interface then resumes the process with a return code indicating the success or failure of the operation. The other subsystems (memory manager, virtual file system and inter-process communication) all depend on the process scheduler for similar reasons.

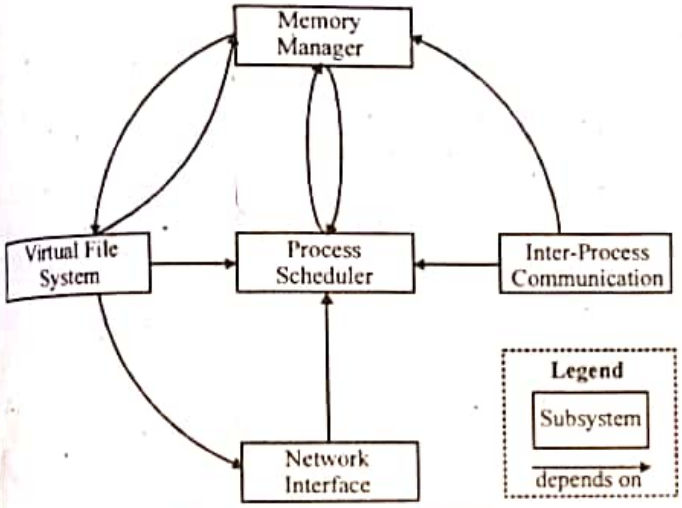


Fig. : Kernel Subsystem Overview

The other dependencies are somewhat less obvious, but equally important :

- The process scheduler subsystem uses the memory manager to adjust the hardware memory map for a specific process when that process is resumed.
- The inter-process communication subsystem depends on the memory manager to support a shared-memory communication mechanism. This mechanism allows two processes to access an area of common memory in addition to their usual private memory.
- The virtual file system uses the network interface to support a network file system (NFS) and also uses the memory manager to provide a ramdisk device.
- The memory manager uses the virtual file system to support swapping; this is the only reason that the memory manager depends on the process scheduler. When a process accesses memory that is currently swapped out, the memory manager makes a request to the file system to fetch the memory from persistent storage and suspends the process.

Q.20 *(a) What is process management? Explain process scheduling in Linux.*

(b) Explain various differences in Unix and Linux operating system. *[R.T.U. 2015]*

Ans.(a) Process Management : *Refer to Q.15.*

Process Identity

Process ID is a unique priority number assigned to a currently running or about to run process. On the basis of process id the process executes according to process queues:

- **Process ID (PID) :** The unique identifier for the process; used to specify processes to the operating system when an application makes a system call to signal, modify, or wait for another process.
- **Credentials :** Each process must have an associated user ID and one or more group IDs that determine the process's rights to access system resources and files.
- **Personality :** Not traditionally found on UNIX systems, but under Linux each process has an associated personality identifier that can slightly modify the semantics of certain system calls.Used primarily by emulation libraries to request that system calls be compatible with certain specific flavors of UNIX.

Process Environment

The process's environment is inherited from its parent and is composed of two null terminated vectors:

- The argument vector lists the command-line arguments used to invoke the running program; conventionally starts with the name of the program itself.
- The environment vector is a list of "NAME=VALUE" pairs that associates named environment variables with arbitrary textual values.

Passing environment variables among processes and inheriting variables by a process's children are flexible means of passing information to components of the user-mode system software. The environment-variable mechanism provides a customization of the operating system that can be set on a per-process basis, rather than being configured for the system as a whole.

Process Context : Process context is the state of a running program at any one time, it changes constantly. The scheduling context is the most important part of the process context; it is the information that the scheduler needs to suspend and restart the process. The kernel maintains accounting information about the resources currently being consumed by each process and the total resources consumed by the process in its lifetime so far. The file table is an array of pointers to kernel file structures. When making file I/O system calls, processes refer to files by their index into this table. The current root and default directories to be used for new file searches are stored here. The signal-handler table defines the routine in the process's address space to be called when specific signals arrive. The virtual memory context of a process describes the full contents of its private address space.