

ANSWER KEY SUBMISSION

Date of Exam & Session	23.6.2022 FN	Category of Exam	CLA1/CLA2/CLA3/SURPRISE TEST
Course Name	Operating Systems	Course Code	18CSC205J
Name of the Faculty submitting	Dr.S.Veena & Dr.Shanmuganathan	Date of submission of Answer Key	23.6.2022
Department to which the Faculty belongs to	CSE	Total Marks	50

PART A (20 X 1= 20)

Q. No.	Questions	Marks
1.	When a program tries to access a page that is mapped in address space but not loaded in physical memory, then what occurs a) page fault occurs b) fatal error occurs c) segmentation fault occurs d) no error occurs	1
2.	Which algorithm chooses the page that has not been used for the longest period of time whenever the page required to be replaced? a) first in first out algorithm b) additional reference bit algorithm c) least recently used algorithm d) counting based page replacement algorithm	1
3.	Working set model for page replacement is based on the assumption of _____ a) modularity b) locality c) globalization d) random access	1
4.	Applying the LRU page replacement to the following reference string. 1 2 4 5 2 1 2 4 The main memory can accommodate 3 pages and it already has pages 1 and 2. Page 1 came in before page 2. How many page faults will occur? a) 2 b) 3 c) 4 d) 5	1
5.	Which of the following page replacement algorithms suffers from Belady's Anomaly? a) Optimal replacement b) LRU c) FIFO d) Both optimal replacement and FIFO	1
6.	A process refers to 5 pages, A, B, C, D, E in the order : A, B, C, D, A, B, E, A, B, C, D, E. If the page replacement algorithm is FIFO, the number of page transfers with an empty internal store of 3 frames is? a) 8 b) 10	1

e. Shanmuganathan *Veena*

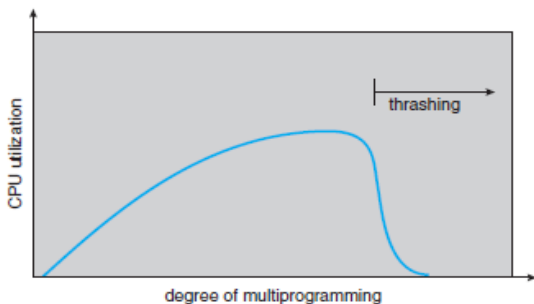
SRM INSTITUTE OF SCIENCE AND TECHNOLOGY, RAMAPURAM
DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

	c) 9 d) 7	
7.	The aim of creating page replacement algorithms is to _____ a) replace pages faster b) increase the page fault rate c) decrease the page fault rate d) to allocate multiple pages to processes	1
8.	A FIFO replacement algorithm associates with each page the _____ a) time it was brought into memory b) size of the page in memory c) page after and before it d) page recently used	1
9.	What is the Optimal page – replacement algorithm? a) Replace the page that has not been used for a long time b) Replace the page that has been used for a long time c) Replace the page that will not be used for a long time d) Replace the page that will big in size.	1
10.	Optimal page – replacement algorithm is difficult to implement, because _____ a) it requires a lot of information b) it requires future knowledge of the reference string c) it is too complex d) it is extremely expensive	1
11.	Defective sectors on a disk are often called as _____. a) good block b) destroyed blocks c) bad blocks d) error blocks	1
12.	File type can be represented by _____ a) file name b) file extension c) file identifier d) file permission	1
13.	_____ is a sequence of procedures and functions. a) object file b) source file c) executable file d) text file	1
14.	Which of the following is crucial time while accessing the data on the disks? a) seek time b) Rotational delay c) Transmission time d) Waiting time	1
15.	The _____ policy restricts scanning to one direction only. a) SCAN b) C-SCAN c) FCFS d) SSTF	1
16.	_____ are often used where very rapid access is required, where fixed length records are used, and where records are always accessed one at a time. a) Direct files b) Sequential files c) Indexed files d) Indexed sequential file	1
17.	_____ refers to the logical structuring of records.	1

e. shayon *Ueenu*

	a) file organisation b) structural organisation c) physical organisation d) logical organisation	
18.	The universe consists of _____ a) all users in the system b) all users that are not owners c) all users that aren't included in the group or owners d) all users that are included in group	1
19.	Virtual memory uses disk space as an extension of _____. a) secondary storage b) main memory c) tertiary storage d) cache memory	1
20.	_____ is series or collection of bits where each bit corresponds to a disk block. a) Bit Vector b) Array c) List d) Index	1

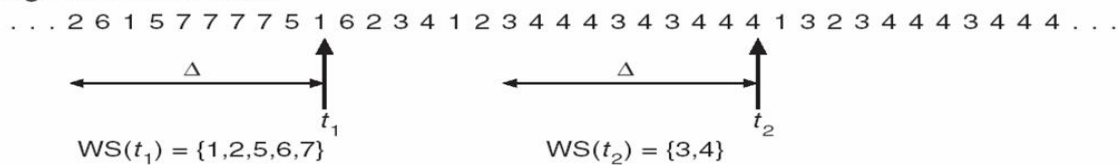
PART B (3 X 10 = 30)
ANSWER ALL THE QUESTIONS

Q. No.	Questions	Marks
21	<p>a) Explain in detail about thrashing and working set model</p> <p>Thrashing : (5 Marks)</p> <ul style="list-style-type: none"> ➤ A process is busy with swapping pages in and out. This high paging activity is called Thrashing. ➤ A process is in thrashing if it is spending more time on paging rather than executing. <p>Cause of Thrashing</p> <p>If a process does not have “enough” pages, the page-fault rate is very high. This leads to:</p> <ul style="list-style-type: none"> ➤ low CPU utilization. ➤ operating system thinks that it needs to increase the degree of multiprogramming. another process added to the system <div style="text-align: center;">  </div> <p>Working Set Model: (5 Marks)</p> <ul style="list-style-type: none"> ▪ It is based on the assumption of locality. <ul style="list-style-type: none"> $\Delta \equiv$ working-set window \equiv a fixed number of page references Example: 10,000 instruction ▪ The idea is to examine the most recent Δ page references. ▪ WSS_i (working set of Process P_i) = total number of pages referenced in the most recent Δ (varies in time) 	10

e. shayon *Ueans*

- Δ if Δ too small will not encompass entire locality.
- Δ if Δ too large will encompass several localities.
- Δ if $\Delta = \infty \Rightarrow$ will encompass entire program.
- $D = \sum WSS_i \equiv$ total demand frames
- if $D > m \Rightarrow$ Thrashing
 - Δ Policy if $D > m$, then suspend one of the processes.
- if $D \leq m \Rightarrow$ No Thrashing
- The sequence of memory references shown in Figure,
- if $\Delta = 10$ memory references,
- Then the working set at time
- t_1 is $\{1, 2, 5, 6, 7\}$
- t_2 , the working set has changed to $\{3, 4\}$.

page reference table



The **working set model** states that a process can be in RAM if and only if all of the pages that it is currently using (often approximated by the most recently used pages) can be in RAM.

b) Give a brief discussion about virtual memory and how OS handles the page fault effectively?

Virtual memory (5 Marks)

Virtual memory is a technique that allows the execution of processes that are not completely in Main memory.

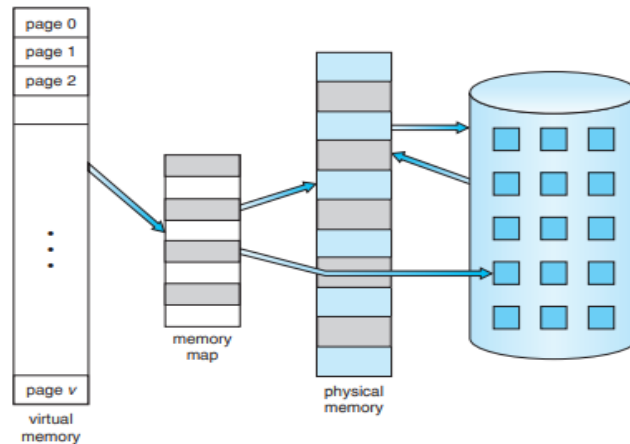
Virtual memory – involves the separation of logical memory as perceived by users from physical memory.

Need for virtual memory

- Programs often have code to handle unusual error conditions. Since these errors seldom, if ever, occur in practice, this code is almost never executed.
- Arrays, lists, and tables are often allocated more memory than they actually need. An array may be declared 100 by 100 elements, even though it is seldom larger than 10 by 10 elements.
- An assembler symbol table may have room for 3,000 symbols, although the average program has less than 200 symbols. Certain options and features of a program may be used rarely. Even in those cases where the entire program is needed, it may not all be needed at the same time.
- One major advantage of this scheme is that programs can be larger than physical memory.
- Further, virtual memory abstracts main memory into an extremely large, uniform array of storage, separating logical memory as viewed by the user from physical memory.
- This separation allows an extremely large virtual memory to be provided for programmers when only a smaller physical memory is available.

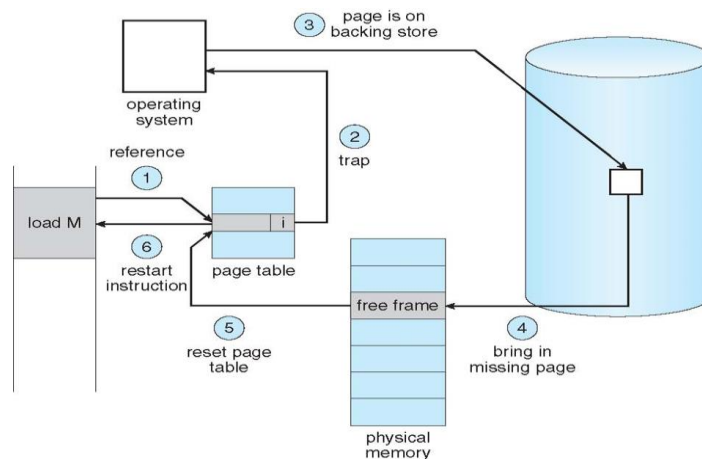
10

(Handwritten signatures)



Steps in Handling a Page Fault (5 Marks):

1. Check an internal table (usually kept with the process control block) whether the reference was a valid or an invalid memory access.
2. If the reference
 - Invalid, we terminate the process.
 - Valid but we have not yet brought in that page, we now page it in.
3. Find a free frame
4. Schedule a disk operation to read the desired page into the newly allocated frame.
5. When the disk read is complete, modify the internal table kept with the process and the page table to indicate that the page is now in memory.
6. We restart the instruction that was interrupted by the trap. The process can now access the page as though it had always been in memory.



- 22 a) Consider a disk queue with requests for I/O to blocks on cylinders 98, 183, 41, 122, 14, 124, 65, 67. The head is initially at cylinder number 53. The cylinders are numbered from 0 to 199. What is the total head movement (in number of cylinders) incurred while servicing these requests for each of the following disk scheduling algorithms:
 i.FCFS ii. SSTF iii. SCAN iv. LOOK v. C-LOOK

i.FCFS(First Come First Serve)

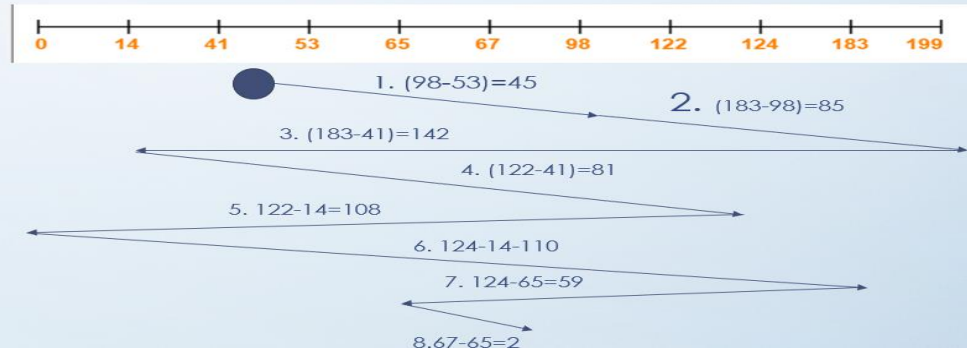
- FCFS is the simplest of all the Disk Scheduling Algorithms. In FCFS, the requests

2

e. shayon *Ueenu*

are addressed in the order they arrive in the disk queue.

- head is initially at cylinder number 53. The cylinders are numbered from 0 to 199.
- 98, 183, 41, 122, 14, 124, 65, 67



- Total head movements using FCFS = $[(98 - 53) + (183 - 98) + (183 - 41) + (122 - 41) + (122 - 14) + (124 - 14) + (124 - 65) + (67 - 65)]$
- $= 45 + 85 + 142 + 81 + 108 + 110 + 59 + 2$
- $= 632$

ii. SSTF(Shortest Seek Time First)

- **Shortest Seek Time First (SSTF)** selects the request with the minimum seek time from the current head position

head is initially at cylinder number 53. The cylinders are numbered from 0 to 199.

98, 183, 41, 122, 14, 124, 65, 67



$$\begin{aligned} \text{Total head movement using SSTF} &= (65-53) + (67-65) + (67-41) + (41-14) + (98-14) + \\ &+ (122-98) + (124-122) + (183-124) \\ &= 12 + 2 + 26 + 27 + 84 + 24 + 2 + 59 = 236 \end{aligned}$$

iii. SCAN

- The disk arm starts at one end of the disk, and moves toward the other end, servicing requests until it gets to the other end of the disk, where the head movement is reversed and servicing continues.
- SCAN algorithm Sometimes called the elevator algorithm

(Handwritten signatures)

- head is initially at cylinder number 53. The cylinders are numbered from 0 to 199.
- 98, 183, 41, 122, 14, 124, 65, 67



Total Head Movement using SCAN = $[(65-53)+(67-65)+(98-67)+(122-98)+(124-122)+(183-124)+(199-183)+(199-41)+(41-14)]$
 $=12+2+31+24+2+59+16+158+27=331$

iv. LOOK

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

head is initially at cylinder number 53. The cylinders are numbered from 0 to 199.

98, 183, 41, 122, 14, 124, 65, 67



Total Head Movement using look = $[(65-53)+(67-65)+(98-67)+(122-98)+(124-122)+(183-124)+(183-41)+(41-14)]$
 $=12+2+31+24+2+59+142+27=299$

v. C-LOOK

- In CLOOK, the disk arm in spite of going to the end goes only to the last request to be serviced in front of the head and then from there goes to the other end's last request. Thus, it also prevents the extra delay which occurred due to unnecessary traversal to the end of the disk.

head is initially at cylinder number 53. The cylinders are numbered from 0 to 199

Total Head Movement using C-LOOK = $[(65-53)+(67-65)+(98-67)+(122-98)+(124-122)+(183-124)+(183-14)+$

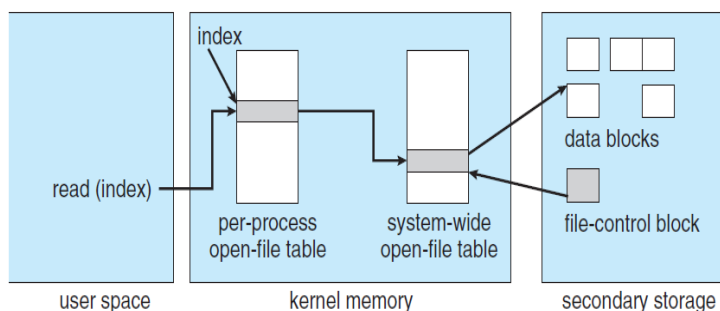
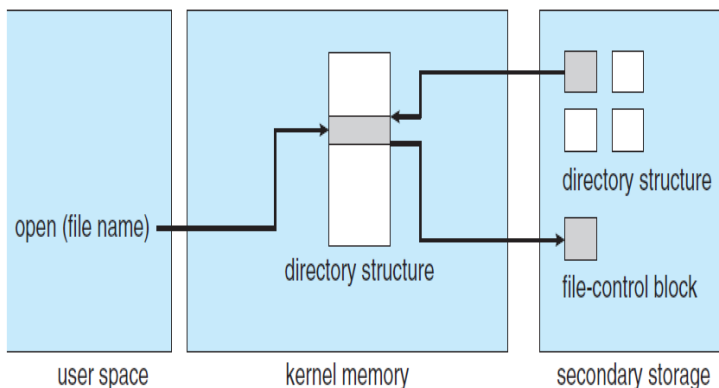
e. shayon *Ueans*

(41-14)]
 = 12+2+31+24+2+59+169+27)=326

b) Explain the File system implementation and Directory implementation.

File-System Implementation

- **Types - On-disk and in-memory structures**
- **On Disk Data Structures**
 - ✓ **Boot Control Block** - Boot Control Block contains all the information which is needed **to boot an operating system from that volume**. It is called boot block in UNIX file system. In NTFS, it is called the partition boot sector.
 - ✓ **Volume Control Block** - Volume control block all the information regarding that volume such as number of blocks, size of each block, partition table, pointers to free blocks and free FCB blocks. In UNIX file system, it is known as super block. In NTFS, this information is stored inside master file table.
 - ✓ **Directory Structure (per file system)** -A directory structure (per file system) contains file names and pointers to corresponding FCBs.
 - ✓ **File Control Block** - File Control block contains all the details about the file such as ownership details, permission details, file size,etc. In UFS, this detail is stored in inode.
- **In-Memory File System Structures**
 - ✓ The in-memory data structures are used for file system management as well as performance improvement via caching. This information is loaded on the mount time and discarded on ejection.



✓ Virtual File Systems

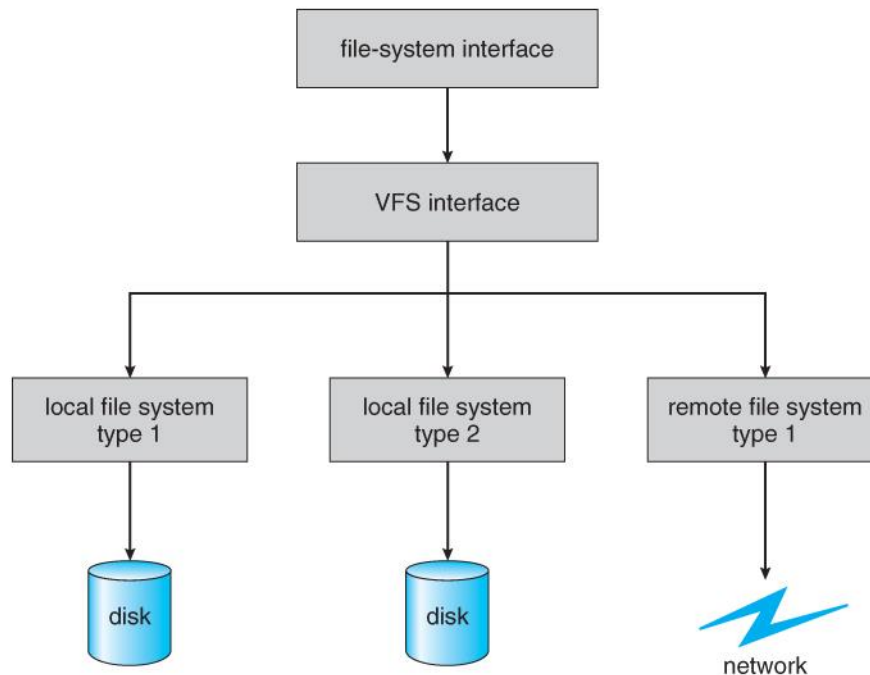
Virtual File Systems, VFS, provide a common interface to multiple different filesystem types. In addition, it provides for a unique identifier (vnode) for files across the entire space, including across all filesystems of different types. (UNIX

(Handwritten signatures)

inodes are unique only across a single filesystem, and certainly do not carry across networked file systems.)

The VFS in Linux is based upon four key object types:

- The inode object, representing an individual file
- The file object, representing an open file.
- The superblock object, representing a filesystem.
- The dentry object, representing a directory entry.



Directory Implementation

Directories need to be fast to search, insert, and delete, with a minimum of wasted disk space.

Linear List – 2 Marks

- A linear list is the simplest and easiest directory structure to set up, but it does have some drawbacks.
- Finding a file (or verifying one does not already exist upon creation) requires a linear search.
- Deletions can be done by moving all entries, flagging an entry as deleted, or by moving the last entry into the newly vacant position.
- Sorting the list makes searches faster, at the expense of more complex insertions and deletions.
- A linked list makes insertions and deletions into a sorted list easier, with overhead for the links.
- More complex data structures, such as B-trees, could also be considered.

Hash Table – 2 Marks

- A hash table can also be used to speed up searches.
- Hash tables are generally implemented *in addition to* a linear or other structure

[Handwritten Signature]

- a) Consider a reference string: 4, 7, 6, 1, 7, 6, 1, 2, 7, 2. the number of frames in the memory is 3. Find out the number of page faults respective to:

1. Optimal Page Replacement Algorithm (5 Marks):

Request	4	7	6	1	7	6	1	2	7	2
Frame 3			6	6	6	6	6	2	2	2
Frame 2		7	7	7	7	7	7	7	7	7
Frame 1	4	4	4	1	1	1	1	1	1	1
Miss/Hit	Miss	Miss	Miss	Miss	Hit	Hit	Hit	Miss	Hit	Hit

Number of Page Faults = 5

2. FIFO Page Replacement Algorithm (5 Marks):

Request	4	7	6	1	7	6	1	2	7	2
Frame 3			6	6	6	6	6	6	7	7
Frame 2		7	7	7	7	7	7	2	2	2
Frame 1	4	4	4	1	1	1	1	1	1	1
Miss/Hit	Miss	Miss	Miss	Miss	Hit	Hit	Hit	Miss	Miss	Hit

Number of Page Faults in FIFO = 6

- b) Explain the various Directory structure with advantages and disadvantages.
A **directory** is a container that is used to contain folders and file. It organizes files and folders into a hierarchical manner.

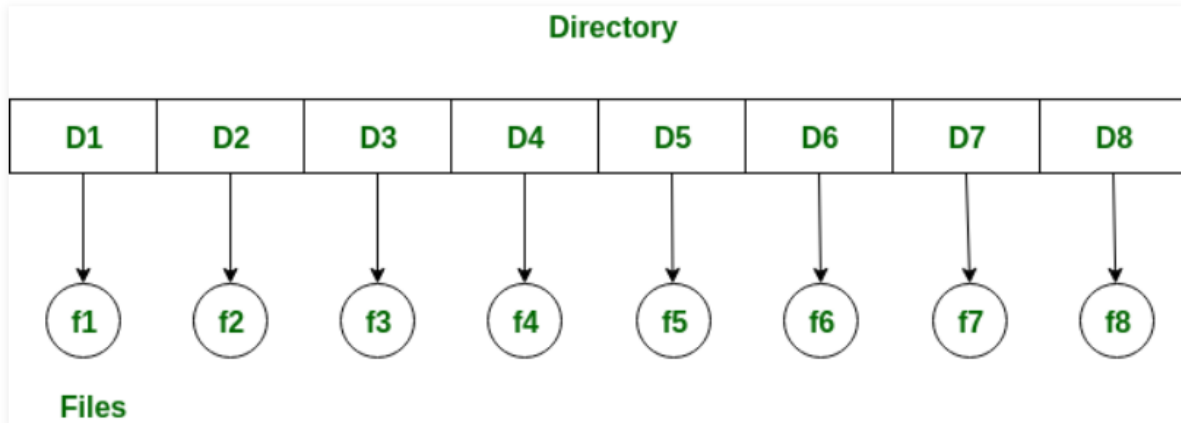
- There are several logical structures of a directory, these are given below.

- Single-level directory**
- Two-level directory**
- Tree-structured directory**
- Acyclic graph directory**
- General graph directory structure**

Single level Directory Structure

- Single level directory is simplest directory structure. In it all files are contained in same directory which make it easy to support and understand.
- A single level directory has a significant limitation, however, when the number of files increases or when the system has more than one user. Since all the files are in the same directory, they must have the unique name. If two users call their dataset test, then the unique name rule is violated.

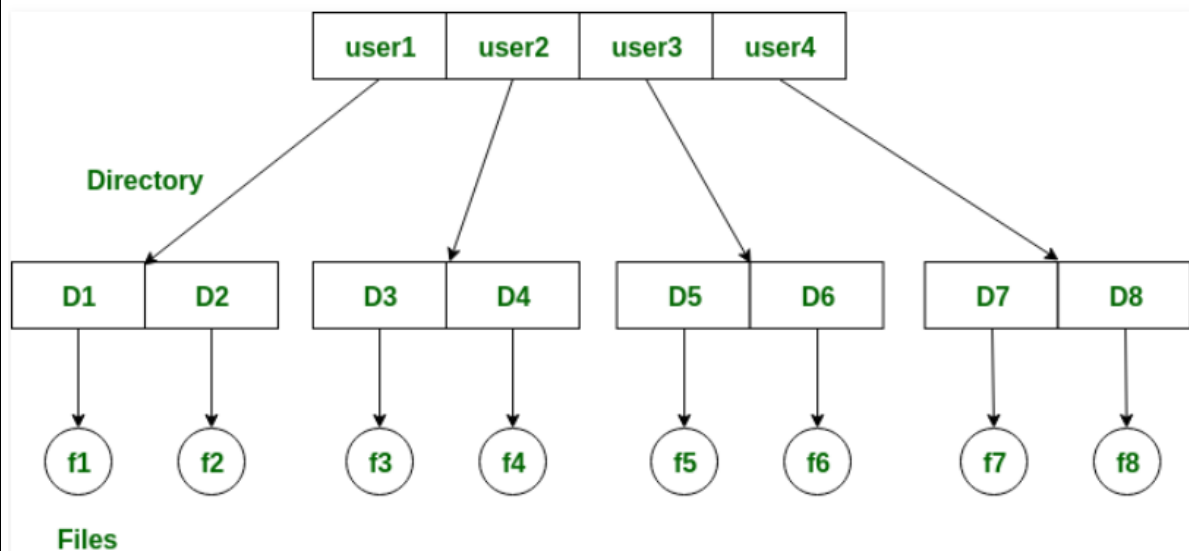
e. shanmugam *Udenu*



2

Two-level directory

- In the two-level directory structure, each user has their own *user files directory (UFD)*. The UFDs have similar structures, but each lists only the files of a single user. The system's *master file directory (MFD)* is searched whenever a new user id is logged in. The MFD is indexed by username or account number, and each entry points to the UFD for that user.

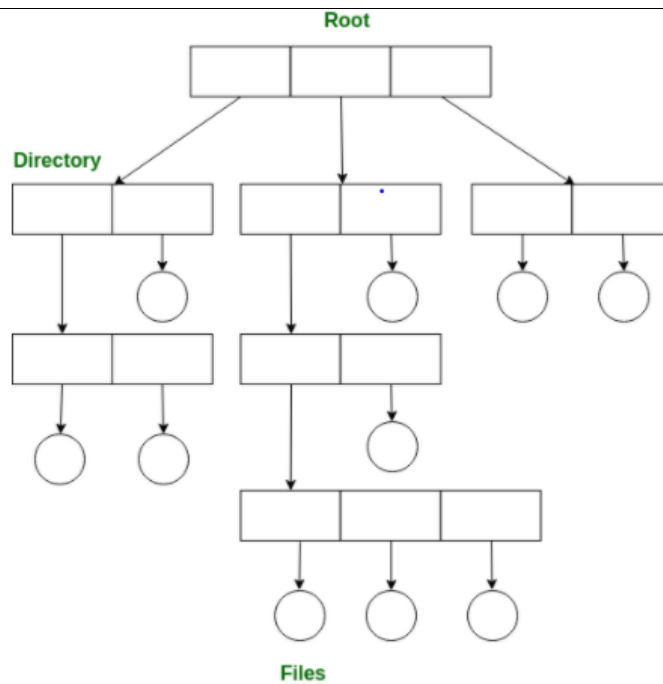


2

Tree-structured directory

- Once we have seen a two-level directory as a tree of height 2, the natural generalization is to extend the directory structure to a tree of arbitrary height.
- This generalization allows the user to create their own subdirectories and to organize their files accordingly.

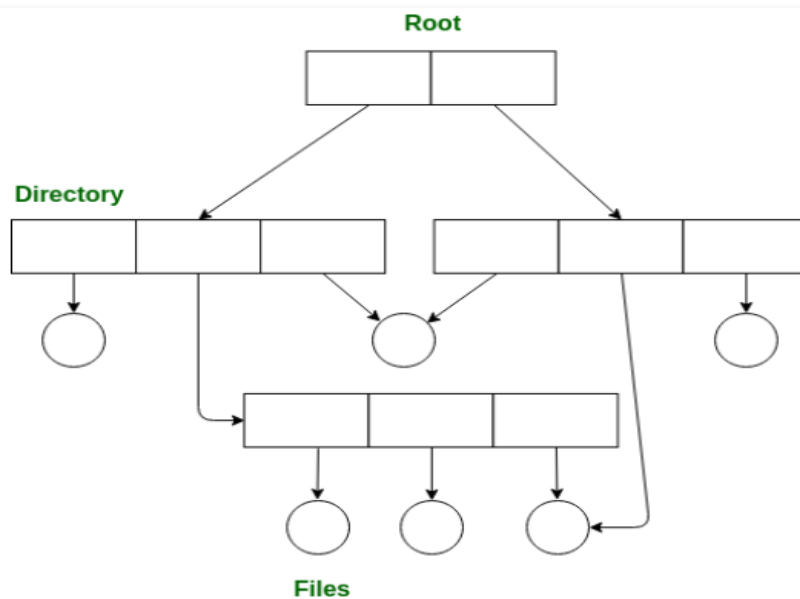
[Handwritten Signature]



2

Acyclic graph directory

- An acyclic graph is a graph with no cycle and allows to share subdirectories and files. The same file or subdirectories may be in two different directories. It is a natural generalization of the tree-structured directory.



2

General graph directory structure

- In general graph directory structure, cycles are allowed within a directory structure where multiple directories can be derived from more than one parent directory.

e. shanmugam *Udenu*

