Operating system .

Destro : kali Linux, Ubuntu, Windows

Mother(root) of all operating system: Unix.

Multics : Mit , e t and bell , ge electronic

Multiple computing system.

Multics - Unix - (1972) E t and bell free and open source

John lion : they write the Book on the design philosophy of Unix .

Linus Torvalds : creator of the “Linux Operating System”

Ubuntu - - community driven OS. It is the Destro of Linux.

Task Of Operating Systems:

1) File Management

2) Memory Management

3) process Management

4) CPU scheduling

5) Hardware Abstraction

When you buy new Hard disk ⦾ you have to format it so you can store data in it .

Formatting secondary storage means creating new file system. Due to which previous data get lost this is side effects of formatting.

New Hard disk you Buy they have no partitions.

they have space but not partitioned it. So partitioning means creating different sections, like bedroom, kitchen. study room (wall created) it is called creating file system

Define FILE: File is an Un formatted uniform stream of bytes.

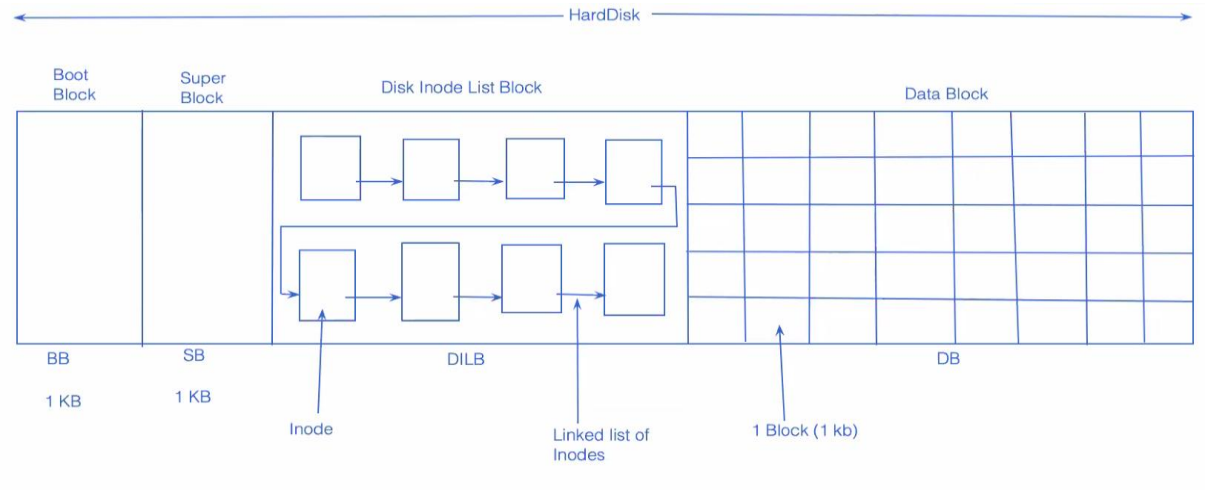
File is everything that is storable

What is File system? File system is way of storing and representing the files in our hard disk.

FAT 32: File allocation Table 32 bit

FAT 64:

Partition of Hard Disk



1) **Boot Block** (Boot loader) Hard disk madhil Operating System ram vr yete (Booting process)
This block contains the data which is required to start(Boot) the operating system.

2) **Super Block** (all information about Hard Disk. Holds metadata about data)

i) No of Inodes
ii) No of Free Inodes
iii) Total no of Blocks
iv) Total no of free blocks
v) Total no of used blocks
vi) Size of file system

3**) DILB** (data Inode link block) use of Link list 🖇🖇

4) **DB** (Data block) each block of 1 KB

Actual data of file get stored in the Data Block

program20.c is file that is stores in any one of the DB Ex. 111 Block number.

Managing and maintaining the large numbers of Data block very difficult.

Inode : Indexed node (systematic way to store data)

It contains all information about **file program20.c** , i.e

1) Inoded Number
2) Size of file
3) Actual size of file
4) File creation date and time
5) Permissions to that file
6) Block number in which file is stored
7) etc

(No. of Files can be created is equal to No. of Free Inodes) rather than the size of Hard Disk

The number of inodes is determined at the time of file system creation and is **fixed** for the lifetime of the file system.

If you have 10 inodes in your file system, you can create up to 10 files or directories. Once you've used all 10 inodes, you won't be able to create additional files or directories within that file system unless you increase the number of inodes, which typically requires creating a new file system with a larger inode count.

When we Delete the file then Inode is not deleted because they have are fixed number of value.

And it can used to another file.

As we are not maintaining the Directory structure so we stored the file name in Inode. But in every operating system Inode don't containing the File name . File name stored in directory

Every thing that is storable has the file sub system :

Ex. Pendrive , HDD, CD etc.

Project :

CVFS : Interview Questions

1) Strat with Explain what is CVFS, File, OS, Task of Operating System ,Hard disk (partition)⊙

2) Which OS task you performed in project i.e File Management

3) Dig of file system. By using this this OS file system dig I have try to implement in my project On the Ram i.e illusions of Hard Disk file system on the Ram

Uses :

1) It used to illustrate or emulate the feel of Linux commands on Non Linux Operating System (Ex Windows)
2) This project Abstract the internal operating system and user's feel like a Linux

Limitations:

1) all things run on Ram so, no file is preserved on hard Disk.

2) virtual project aahe mhanun Sagle RAM vr store hotay.

3) This project support regular files only

4) We are not maintaining the **Directory Structure** for the files

Future Enhacement

1) As this project is virtual and my planning is to convert this project from virtual to real platform Where actual files gets created on hard disk
2) Adding more Linux command related to the file so user feel more interactive with project and he can access and mange it easily.

Books For Reference:

1)Linux System programming by Robert love

2)Advanced unixs programming by w Richerd Stiven

3)Linux programming interface by Michal Kerriks

How project Made ?

1) Design of all data structure and it's members

Who is Mentor or Guide For this ?

Marvellous

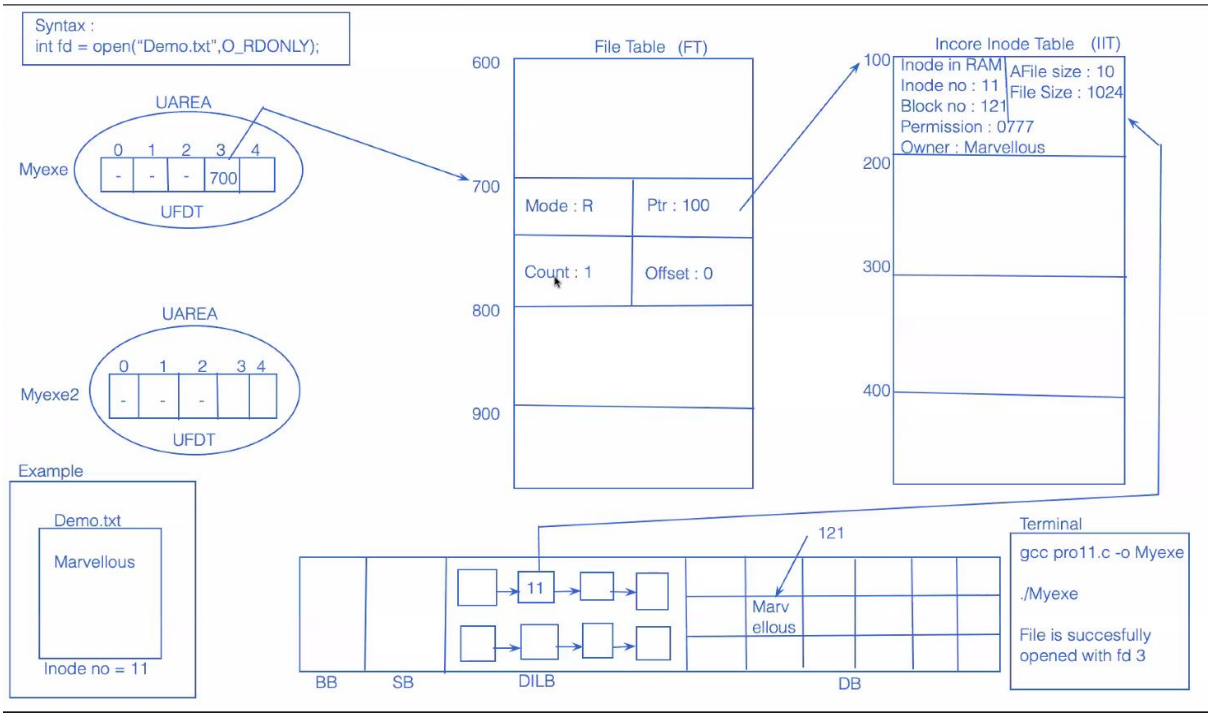
What are the difficulties that you faced in this project?

Developing a file system project can be a complex and challenging task. There are several difficulties and challenges faced during the development of project

1) Designing and understanding the Required Data structure for project
2) Complexity: File systems are inherently complex because they need to manage data storage efficiently while providing fast and reliable access
3) Manging the file permission

Is there any improvement needed in this project?

File Subsystem Diagram



UAREA : User Area It contains the UFDT Array

UFDT : (User File Descriptor Table)

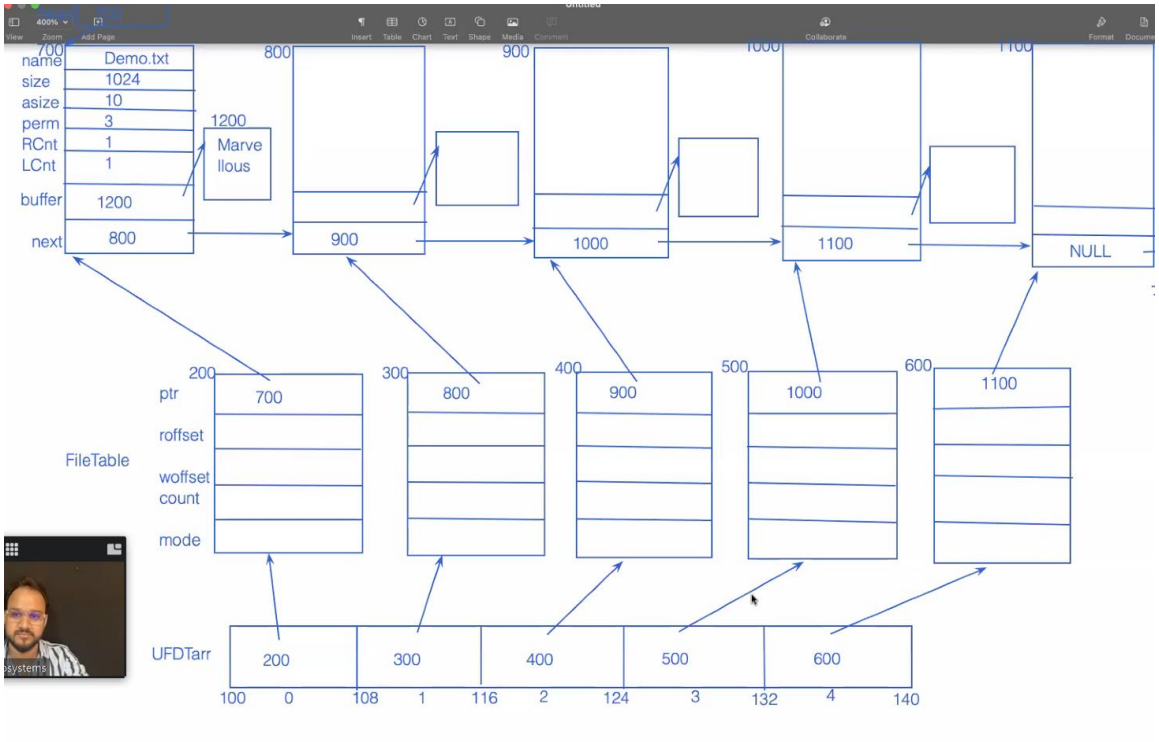
First 3 positions are reserved for 0: std in , 1: std out , 2:std err

Due to this when new file is created the **FD is assign from the 3**

IIT (Incore Inode Table):

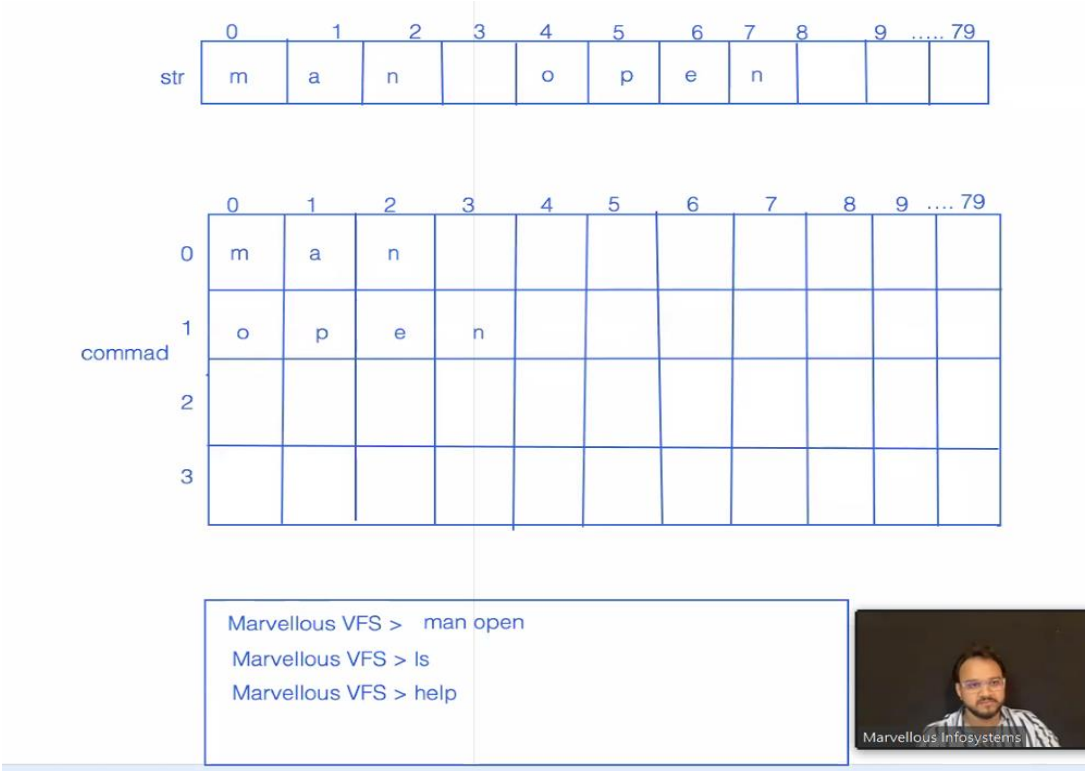
It contains the all Inodes of the files that are open (On RAM) . In above dig. We open only Demo.txt so only one Inode is present in the IIT

When any file is open it's Inode go into the IIT



All Data structures Combined of the project (IMP Digram)

How Command is working and stored ?




```
 char command[4][80]; // command is 2D Array. which containing 4  One Dimensional array and
each  1D Array containg 80 elements and each of type Character
char str[80];   //for taking input in string;
fgets(str, 80, stdin); //or scanf("%[^'\n']s",str) ;

// sscanf() fucntion is takes input from string str and split according the format specifiers
count = sscanf(str, "%s %s %s %s", command[0], command[1], command[2], command[3]);
```

What is the difference between a special file and a regular file?

Regular Files:

- Regular files are the most common type of files in a Unix-like file system.
- They store user data, text, binary information, or program code.
- Regular files are created by users and applications to store information.
- They can be manipulated and modified using standard file operations like reading, writing, and appending data.

Special Files (Device Files):

- Special files are created by the operating system or device drivers to provide a standardized interface for interacting with hardware.

- Special files, also known as device files, are used to interact with hardware devices and certain kernel features.
- Users typically do not create or modify special files directly

Why file Descriptor is greater than 2 ?

File descriptors greater than 2 (0, 1, and 2 are reserved) are used to represent open files or other I/O resources. Here's what each of the first three file descriptors typically represents:

1. File Descriptor **0 (stdin):** This is associated with the standard input stream, usually your keyboard. When you read from the standard input, you are reading from file descriptor 0.
2. File Descriptor **1 (stdout):** This is associated with the standard output stream, usually your terminal or console. When you write to the standard output, you are writing to file descriptor 1.
3. File Descriptor **2 (stderr):** This is associated with the standard error stream, which is also usually your terminal or console. It's used for error messages or diagnostic output. When you write to the standard error, you are writing to file descriptor 2.
4. For example, if you open a file in C using **open("myfile.txt", O_RDONLY)**, you might get a file descriptor like 3, which would represent the opened file "myfile.txt".

What is the difference between library function and system call?

Aspect	Library Function	System Call
Purpose	Common programming tasks	Interacting with the OS kernel
Level	User-level within the program	Interface between user and kernel
Privilege	No inherent privileges	Privileged operations
Overhead	Lower overhead within the same user-level context	Higher overhead due to context switch
Examples	String manipulation functions (e.g., **strlen**, **strcpy**), mathematical functions (e.g., **sqrt**, **sin**)	File I/O (e.g., **open**, **read**, **write**), process management (e.g., **fork**, **exec**), memory management (e.g., **malloc**, **free**), hardware control (e.g., device driver functions)

All Linux File System Command Must Do ?

ls

ls - l

ls - a

rm

cat

cd

chmod

cp

df

find

grep

ln

mkdir

pwd

touch

uname

stat

man

mkfs

Output of code:

$ help

clear : To clear console

create : To create the file

open : To open the file

read : To read the contents from file

write : To write contents into file

ls : To List out all files

lseek : To change file offset

stat : To Display information of file using name

fstat : To Display information of file using file descriptor

truncate : To remove all data from file

close : To close the file

rm : To Delete the file

closeall : To close all opened files

exit : To Terminate file system

$ man create

Description : Used to create new regular file

Usage: create File_name Permission

$ create a.txt 3

File is successfully created with file descriptor : 0

$ read a.txt

ERROR : File empty

$ man read

Description : Used to read data from regular file

Usage: read File_name NO_Of_Bytes_To_Read

Usage: read File_name

Note : If you not specifies the no of bytes to read it will read all the data

$ man write

Description : Used to write data into regular file

Usage: write File_name

After this enter the data that we want to write

$ write a.txt

Enter the data :

Hello World

11 bytes of data written successfully.

$ read a.txt

Hello World

$ stat a.txt

--------Statistical Informantion about file----------

File Name : a.txt

Inode Number : 1

File Size : 1024

Actual File Size : 11

Reference Count : 1

File Permission : Read & Write

$ man fstat

Description : Used Display information of file using file descriptor

Usage: fstat File_Descriptor

$ fstat 0

--------Statistical Informantion about file----------

File Name : a.txt

Inode Number : 1

File Size : 1024

Actual File Size : 11

Reference Count : 1

File Permission : Read & Write

$ ls

File Name Inode number File Size

---

a.txt 1 11

$ read a.txt 5

Hello

$ read a.txt 5

Worl

$ read a.txt 2

d

$ read a.txt 2

ERROR : Reached at end of file

$ lseek a.txt -5 2

Successfully Changed

$ read a.txt 5

World

$ create b.txt 2

File is successfully created with file descriptor : 1

$ write b.txt

Enter the data :

This is text file

17 bytes of data written successfully.

$ read b.txt

ERROR : Permission denied

$ stat b.txt

--------Statistical Informantion about file----------

File Name : b.txt

Inode Number : 2

File Size : 1024

Actual File Size : 17

Reference Count : 1

File Permission : Write

$ ls

File Name Inode number File Size

---

a.txt 1 11

b.txt 2 17

$ truncate a.txt

Data is successfully Removed

$ read a.txt

ERROR : File empty

$ rm a.txt

File is successfully deleted

$ ls

File Name Inode number File Size

---

b.txt 2 17

$ closeall

All files closed successfully

$ exit

Terminating the Virtual File System