Customised Virtual File System

-This project is used to emulate all functionalities provided by File systems.

Platform required

Windows NT platform OR Linux Distributions

Architectural requirement

Intel 32 bit processor

SDK used

None

Technology Used

System Programming usin C

About Virtual File System

In this project we emulated all data structures which are used by operating system to manage File system-oriented tasks.

As the name suggest its virtual because we maintain all records in Primary storage.

In this project we create all data structures which required for file Subsystems as Incore Inode table, File table, UAREA, User File Descriptor Table, Super block, Disk Inode List Block, Data Block, Boot Block etc.

We provide all implementations of necessary system calls and commands of File subsystem as Open, Close, Read, Write, Lseek, Create, RM, LS, Stat, Fstat etc.

While providing the implementations of all above functionality we use out own data structures by referring Algorithms of UNIX operating system.

By using this project we can get overview of UFS (UNIX File System) on any platform.

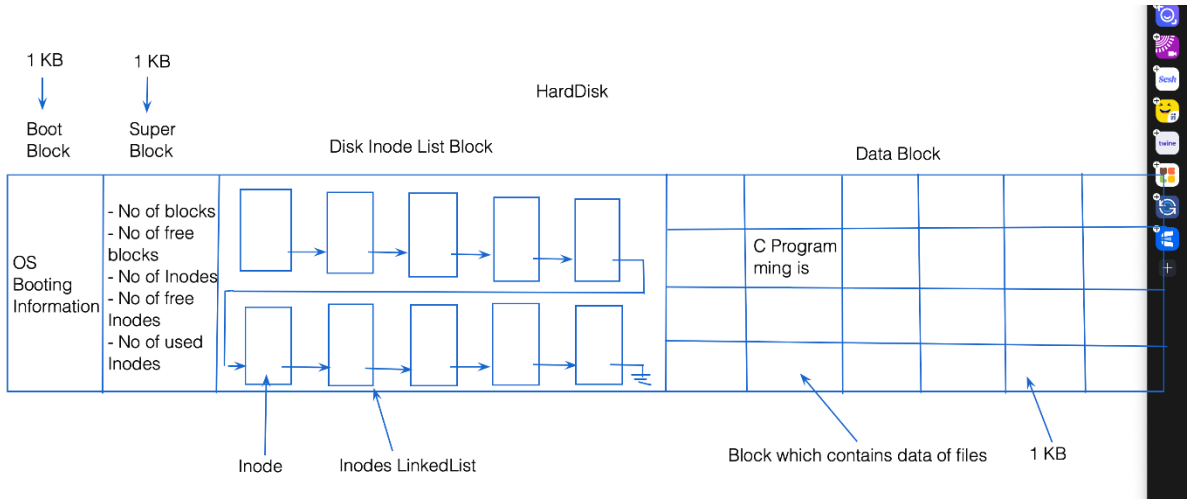
File System

A file system is a method an operating system uses to store, organize, and manage files and directories on a storage device. Some common types of file systems include:

1. FAT (File Allocation Table): An older file system used by older versions of Windows and other operating systems.
2. NTFS (New Technology File System): A modern file system used by Windows. It supports features such as file and folder permissions, compression, and encryption.

3. ext (Extended File System): A file system commonly used on Linux and Unix-based operating systems.
4. HFS (Hierarchical File System): A file system used by macOS.
5. APFS (Apple File System): A new file system introduced by Apple for their Macs and iOS devices.

File System Diagram



Parts of File System:

Boot Block

Super Block

Disk Inode List Block

Data Block

Boot Block

A boot block contains the bootstrap code that is read into the machine upon booting.

Super Block

A superblock describes the state of the file system:

- how large it is;
- how many files it can store;
- where to find free space on the file system;
- who has ownership of it;
- and more.

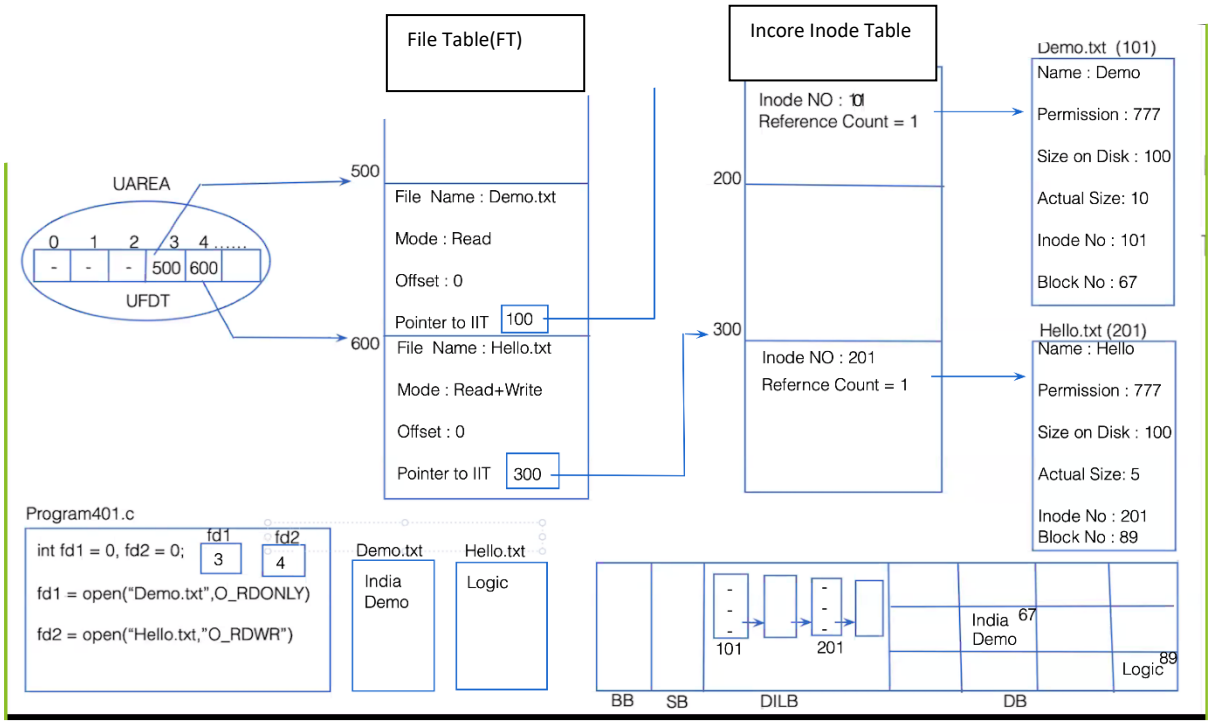
Disk Inode List Block

The inode list is an array of "information nodes" analogous to the FAT (File Allocation Table) system

Data Block

data blocks start at the end of the inode list and contain file data and directory blocks.

CVFS Layout Diagram



UAREA

The u area contains information specific to the process (e.g., open files, current directory, signal actions, accounting information) and a system stack segment for process use. If the process makes a system call the stack frame information for the system call is stored in the system stack segment. Again, this information is kept by the operating system in an area that the process does not normally have access to. Thus, if this information is needed, the process must use special system calls to access it. Like the process itself, the contents of the u area for the process are paged in and out by the operating system.

File Table

The system needs to keep track of each "connection" to a file. Since the same file may have many open connections, this "connection" is truly distinct from the concept of a file. Since distinct processes may even share a connections (say two processes wanted to write to the same log

file, each adding new data to the end of the file), the "connection" information cannot be kept inside the process table. So the kernel keeps a data structure called the *system open-file table* which has an entry for each connection. Each entry contains the connection *status*, e.g. read or write, the *current offset* in the file, and a pointer to a *vnode*, which is the OS's structure representing the file, irrespective of where in the file you may currently be looking. (Note: Linux names their vnode-ish data structure "generic inodes".)

Process Table:

Managing processes is one of the kernel's biggest responsibilities. It decides which process actually gets to run on the CPU (or CPUs) at any point in time. The kernel keeps a data structure (in kernel space) to track information about processes: the *process table*. Each process has an entry in this table (entries are called *process control blocks* that include all sorts of information. The processes ID (pid) is one piece of info, we'll look at some others shortly. You can actually look at all this process table information by examining files (at least they look like files) in the /proc directory. Try man -s4 proc for more information. And, of course, the ps command gives some info as well. Think of the kernel as a business and the processes as customers. The process table is like the records business keep on clients.

System Calls Used in Project

Open()

The open() system call allows you to access a file on a file system. It allocates resources to the file and provides a handle that the process may refer to. Many processes can open a file at once or by a single process only. It's all based on the file system and structure.

read()

It is used to obtain data from a file on the file system. It accepts three arguments in general:

- A file descriptor.
- A buffer to store read data.
- The number of bytes to read from the file.

The file descriptor of the file to be read could be used to identify it and open it using open() before reading.

close()

It is used to end file system access. When this system call is invoked, it signifies that the program no longer requires the file, and the buffers are flushed, the file information is altered, and the file resources are de-allocated as a result.

exec()

When an executable file replaces an earlier executable file in an already executing process, this system function is invoked. As a new process is not built, the old process identification stays, but the new process replaces data, stack, data, head, etc.

exit()

The exit() is a system call that is used to end program execution. This call indicates that the thread execution is complete, which is especially useful in multi-threaded environments. The operating system reclaims resources spent by the process following the use of the exit() system function.

Lseek()

lseek is a system call that is used to change the location of the read/write pointer of a file descriptor.

chmod()

chmod is the command and system call used to change the access permissions and the special mode flags (the setuid, setgid, and sticky flags) of file system objects (files and directories).

Command used in Project

1. ls - The most frequently used command in Linux to list directories
2. pwd - Print working directory command in Linux
3. cd - Linux command to navigate through directories
4. mkdir - Command used to create directories in Linux
5. mv - Move or rename files in Linux
6. cp - Similar usage as mv but for copying files in Linux
7. rm - Delete files or directories
8. touch - Create blank/empty files
9. ln - Create symbolic links (shortcuts) to other files
10. cat - Display file contents on the terminal
11. clear - Clear the terminal display
12. echo - Print any text that follows the command
13. less - Linux command to display paged outputs in the terminal
14. man - Access manual pages for all Linux commands

15. uname - Linux command to get basic information about the OS
16. whoami - Get the active username
17. tar - Command to extract and compress files in Linux
18. grep - Search for a string within an output
19. head - Return the specified number of lines from the top
20. tail - Return the specified number of lines from the bottom
21. diff - Find the difference between two files
22. cmp - Allows you to check if two files are identical
23. comm - Combines the functionality of diff and cmp
24. sort - Linux command to sort the content of a file while outputting
25. export - Export environment variables in Linux
26. zip - Zip files in Linux
27. unzip - Unzip files in Linux
28. ssh - Secure Shell command in Linux
29. service - Linux command to start and stop services
30. ps - Display active processes
31. kill and killall - Kill active processes by process ID or name
32. df - Display disk filesystem information
33. mount - Mount file systems in Linux
34. chmod - Command to change file permissions
35. chown - Command for granting ownership of files or folders
36. ifconfig - Display network interfaces and IP addresses
37. traceroute - Trace all the network hops to reach the destination
38. wget - Direct download files from the internet
39. ufw - Firewall command
40. iptables - Base firewall for all other firewall utilities to interface with
41. apt, pacman, yum, rpm - Package managers depending on the distro
42. sudo - Command to escalate privileges in Linux
43. cal - View a command-line calendar
44. alias - Create custom shortcuts for your regularly used commands
45. dd - Majorly used for creating bootable USB sticks
46. whereis - Locate the binary, source, and manual pages for a command
47. whatis - Find what a command is used for
48. top - View active processes live with their system usage
49. useradd and usermod - Add new user or change existing users data
50. passwd - Create or update passwords for existing users