

Customized Virtual File System (CVFS)

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

Name of the Project: ***Customized Virtual File System.***

Technology used: ***System programming using C.***

User Interface used: ***Command User Interface.***

Platform required: ***Windows NT platform or Linux Distributions.***

Hardware requirements: ***Intel 32 bit processor.***

Data structures used: ***Singly linear linked list.***

Description of the project:

It is a replica of the actual file system used in storage devices like HDD, SSD, Pen drive, etc.

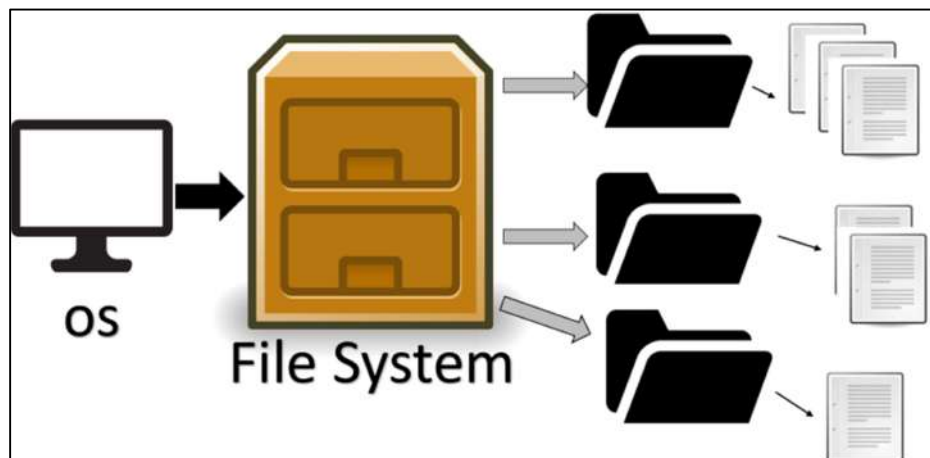
It is executed on RAM, therefore all the file systems operations performed are operational on RAM while the application is running, once the application is terminated all the data is cleared, hence called virtual file system.

The word 'customized' is used to denote the tailor made nature of the project as per our requirement like total space, total number of files that can be created, maximum individual file size, data structure selection, operational commands , etc.

Summing the above points the project is termed as Customized Virtual File System (CVFS).

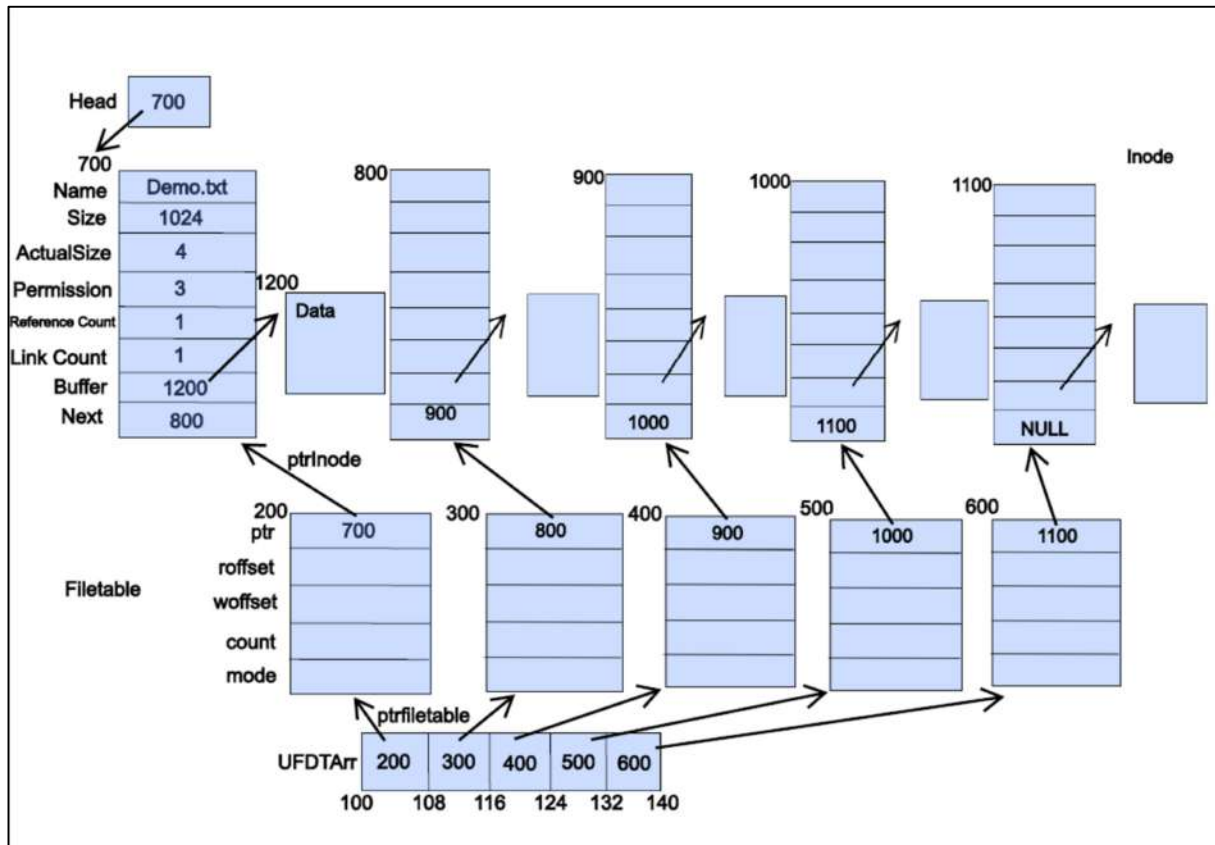
File system:

It is a data structure that the operating system uses to control how data is stored and retrived.



Flow of the project

Consider the below diagrammatic representation of file system,



Working:

To create a new file:

Enter command: create Demo.txt 3.

CreateFile() function which accepts two parameters name (string) and permission (integer). gets called in main.

In createFile() permission will be checked whether it is for read or write, if it does not have permissions it will return access denied.

Further availability for free inode is checked, if there are no any free inodes the function will return and display "There is no inodes".

Then the function will check whether the same named file exists in system through Get_Inode() function , if yes it will return "File already exists" message.

Then to get the inode structure address, file type is checked for '0' across all the inodes, when found the address is stored in a temp pointer.

After this the address of empty file table is searched by using UFDTArr pointer to file table i.e UFDTArr[i].ptrfiletable, by this file table address is found at which memory allocation is done stored in UFDTArr[i] and below variables are set,
Count =1, mode = permission, readoffset = 0, writeoffset = 0.
Now the address of free inode which is in temp pointer is assigned to file table pointer ptr by using

```
UFDTArr[i].ptrfiletable->ptrinode = temp.
```

After this all the variables from inode structure are initialized by using below instructions,

```
UFDTArr[i].ptrfiletable->ptrinode -> filetype = Regular.
```

```
UFDTArr[i].ptrfiletable -> ptrinode -> ReferenceCount = 1;
```

```
UFDTArr[i].ptrfiletable -> ptrinode -> LinkCount = 1;
```

```
UFDTArr[i].ptrfiletable -> ptrinode -> FileSize = MAXFILESIZE;
```

```
UFDTArr[i].ptrfiletable -> ptrinode -> FileActualSize = 0;
```

```
UFDTArr[i].ptrfiletable -> ptrinode -> permission = permission;
```

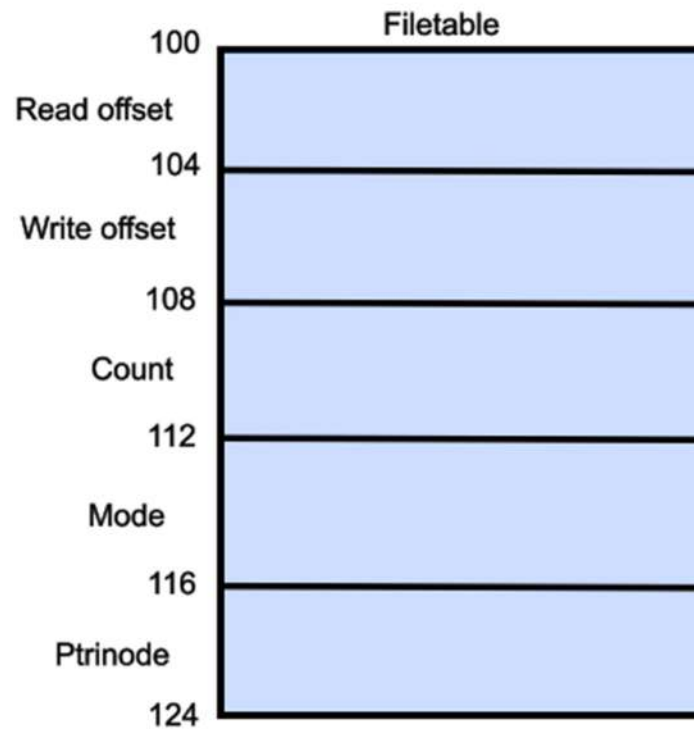
```
UFDTArr[i].ptrfiletable -> ptrinode -> Buffer = (char *)malloc(MAXFILESIZE);
```

Buffer pointer is used to store the address of the data block which contains the actual data content of the file, hence memory is allocated to store the character data of maximum size equivalent to max file size allowed/specified (1024 bytes)

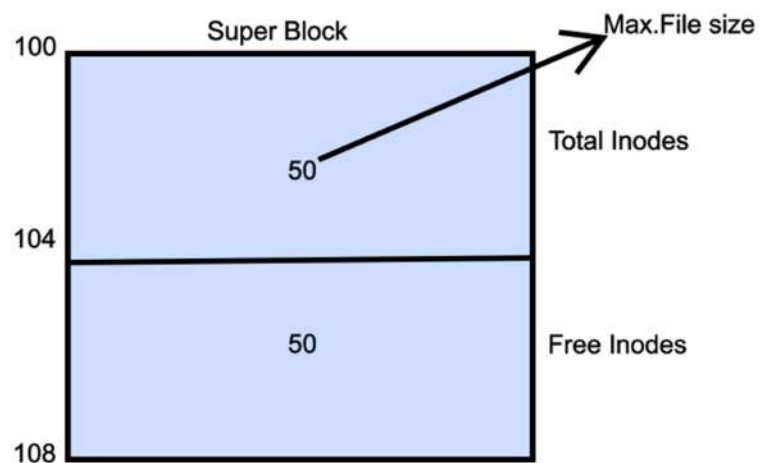
Lastly the createfile function returns integer value i.e. index of the UFDTArr[i] as file descriptor which is displayed as "File successfully created with fd 0" .

Data structures:

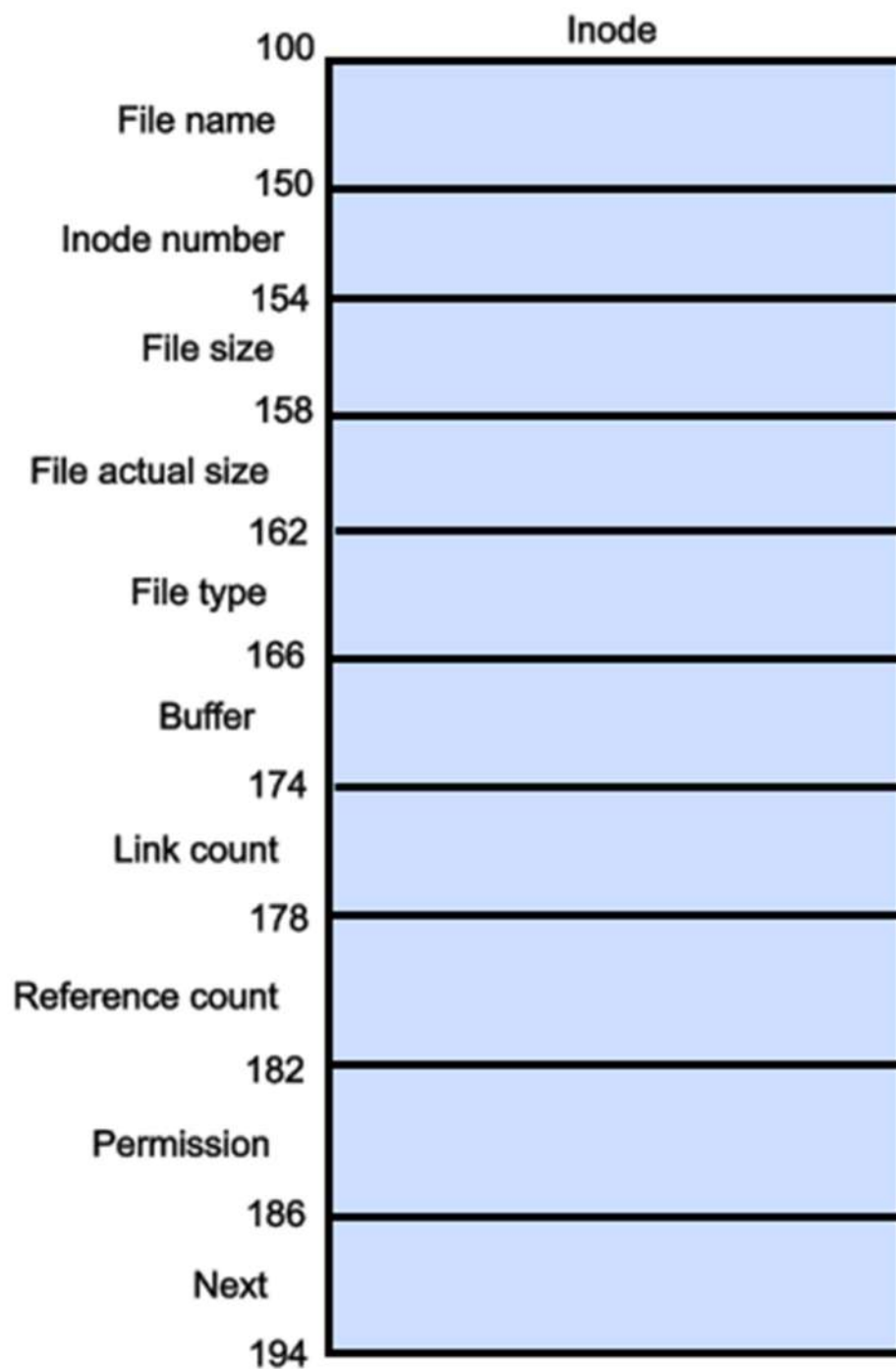
Filetable structure



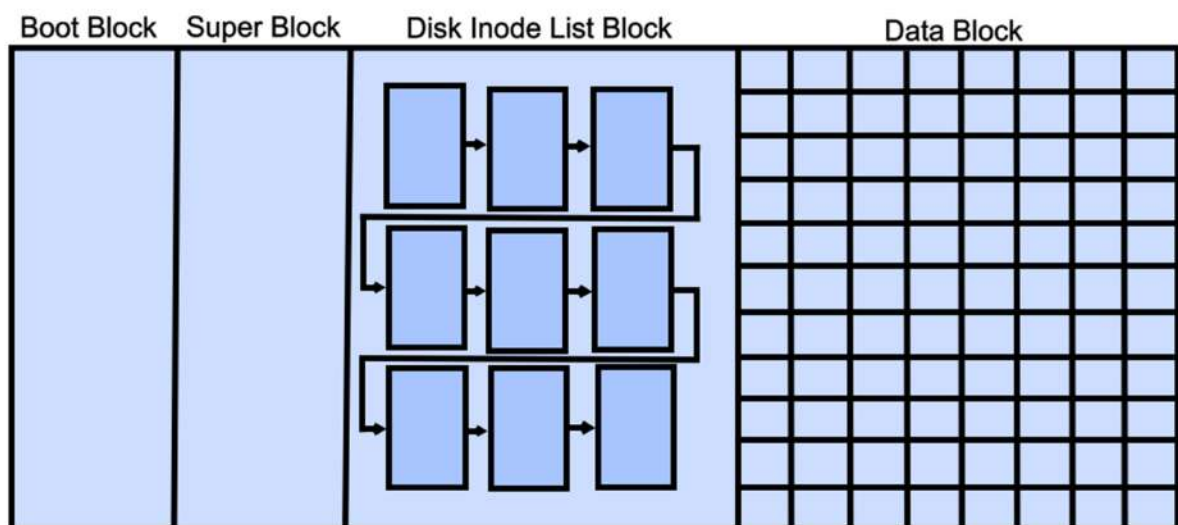
Superblock structure



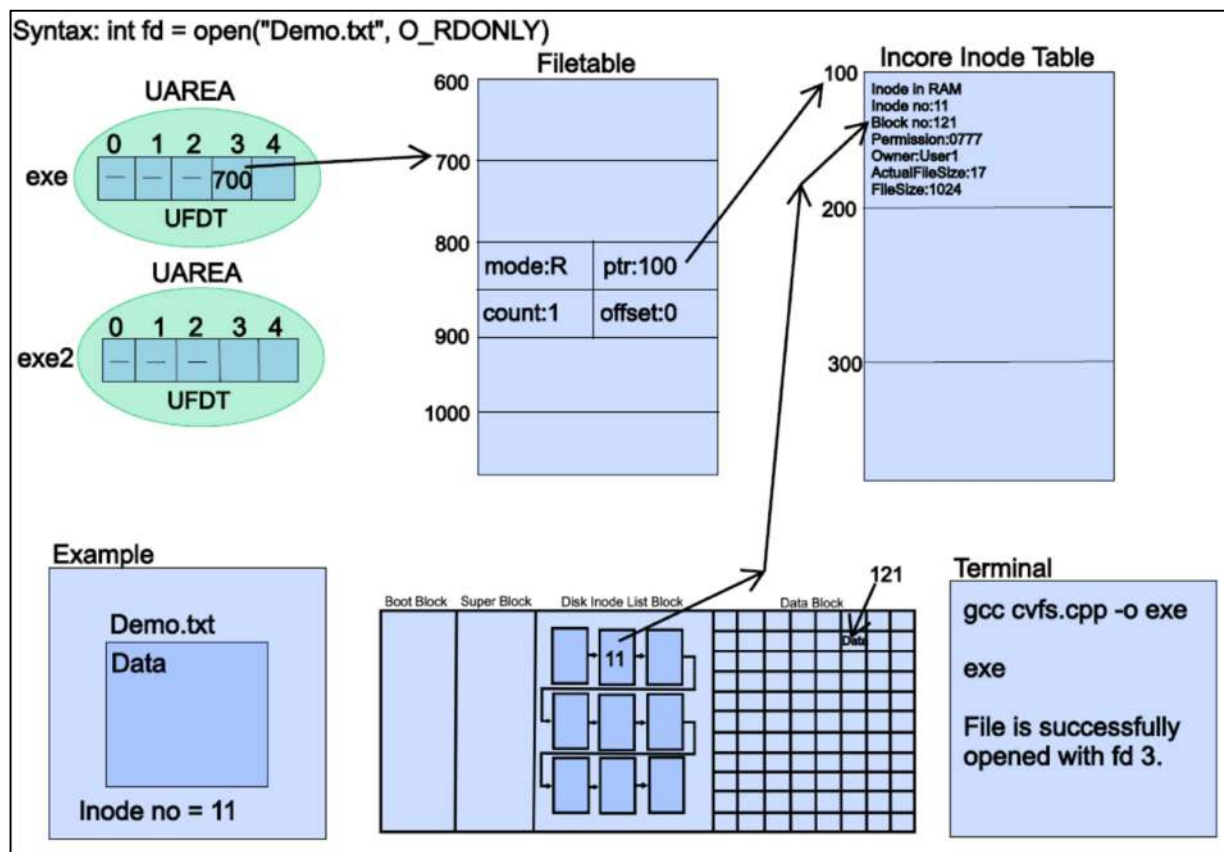
Inode Structure



Hard disk Layout:



File system working fig:



Project Features:

DisplayHelp:

This function is used to display the commands which are used to demonstrate the file system functionality.

To call the function execute the application on command prompt and type help – enter.

Below will be the output:

```
void DisplayHelp()
{
    printf("ls : To List out all files\n");
    printf("clear : To clear console\n");
    printf("open : To open the file\n");
    printf("close : To close the file\n");
    printf("closeall : To close all opened files\n");
    printf("read : To read the contents from the file\n");
    printf("write : To write contents into the file\n");
    printf("exit : To Terminate file system\n");
    printf("stat : To Display information of the file using name\n");
    printf("fstat : To Display information of the file using file descriptor\n");
    printf("truncate : To Remove all data from file\n");
    printf("rm : To Delete the file\n");
}
```

man:

This command is used to display the user manual of the commands that we can run on the terminal.

Syntax: man command, command can be any keyword listed in the help menu.

Example: man open command will given below result,

Select C:\Windows\system32\cmd.exe - E

```
CVFS : >man open
Description: Used to open existing file
Usage: open File_name mode

CVFS : >
```

man command gives brief description of the function and the command usage syntax for calling the function for performing the required functionality.

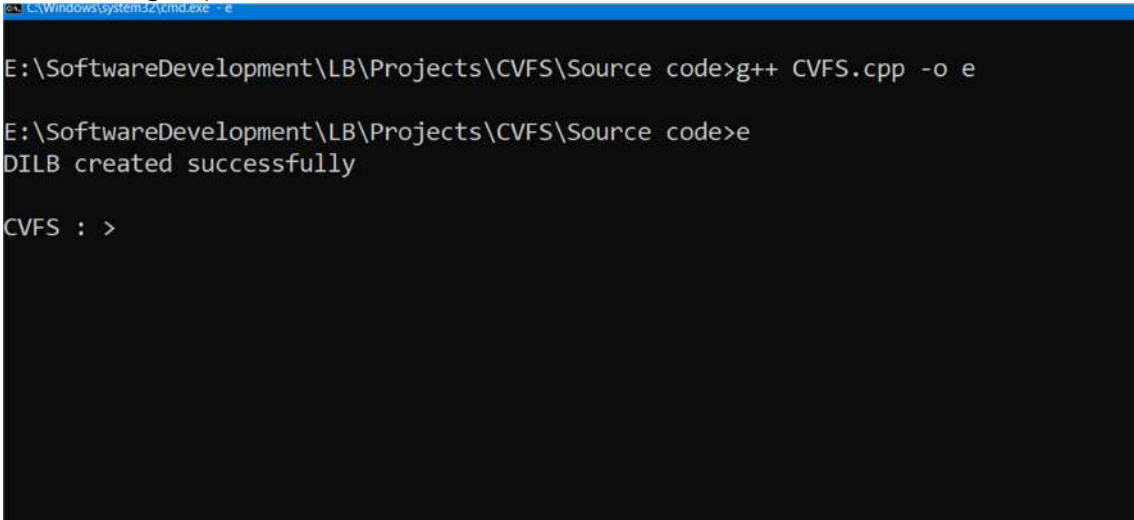
InitialiseSuperBlock():

This function is used to initialise the total inodes and free inodes to the maximum inodes of the file system. It gets called automatically when we run the application.

In this function the total inodes and the free nodes are set to the maximum node size.

DILB(): (Disk Inode List Block)

This function is used to create the inode linked list, which contains all the information of the file such as, file name, file type, file size, address of actual data, next inode address, etc. (Refer Inode structure diagram).



```
C:\Windows\system32\cmd.exe - e
E:\SoftwareDevelopment\LB\Projects\CVFS\Source code>g++ CVFS.cpp -o e
E:\SoftwareDevelopment\LB\Projects\CVFS\Source code>e
DILB created successfully
CVFS : >
```

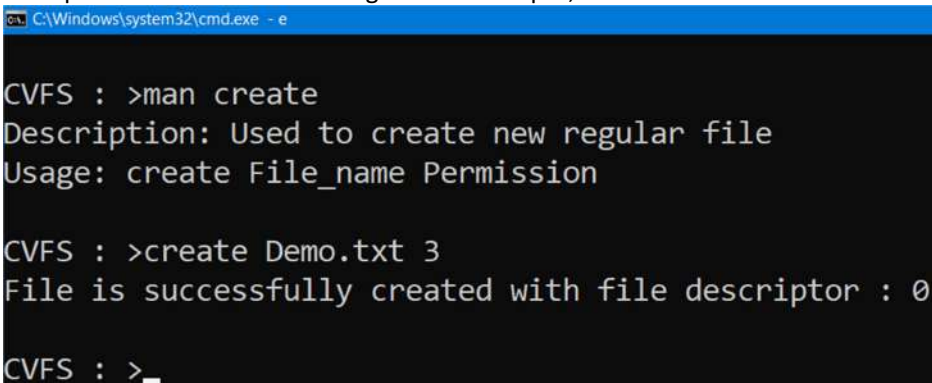
CreateFile():

This function is used to create new file.

Syntax: Create File_name Permission.

The input to this function is file name which is string and permission (Read / Write) which is integer.

Example: create Demo.txt 3 will give below output,



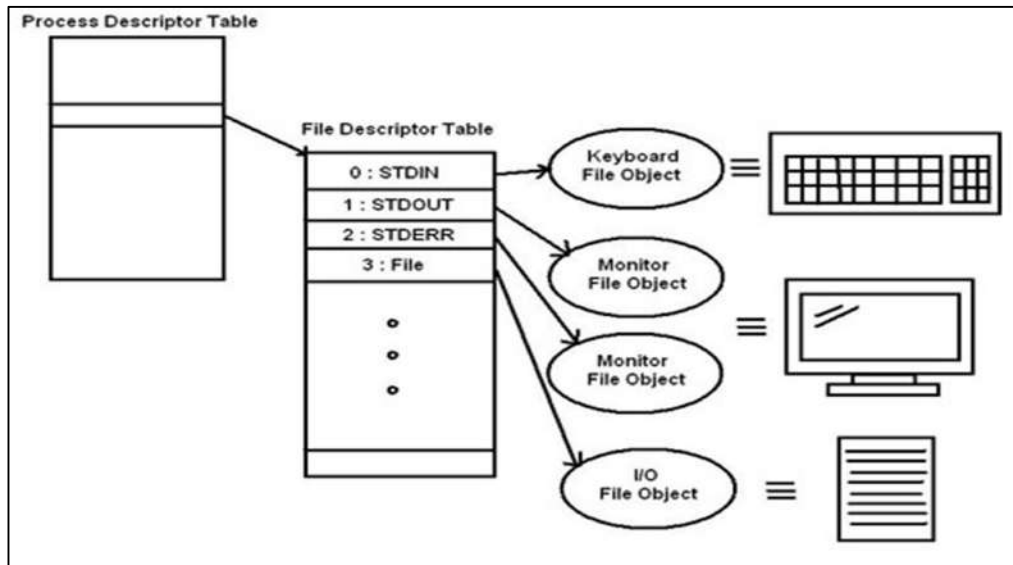
```
C:\Windows\system32\cmd.exe - e
CVFS : >man create
Description: Used to create new regular file
Usage: create File_name Permission

CVFS : >create Demo.txt 3
File is successfully created with file descriptor : 0
CVFS : >
```

File descriptor:

A file descriptor is a number that uniquely identifies an open file in a computer's operating system. It describes a data resource, and how that resource may be accessed. When a program asks to open a file — or another data resource, like a network socket — the kernel:

- I. Grants access.
- II. Creates an entry in the global file table.
- III. Provides the software with the location of that entry.



The first three file descriptors, by default, are STDIN (standard input), STDOUT (standard output), and STDERR (standard error).

OpenFile():

This function is used to open an existing file.

Syntax: open file_name mode.

The input to this function is file name which is string and mode (Read / Write) which is integer.

Example: open Demo.txt 1 will give below output,

```
C:\Windows\system32\cmd.exe - e
CVFS : >man open
Description: Used to open existing file
Usage: open File_name mode

CVFS : >open Demo.txt 1
File is successfully opened with file descriptor : 1

CVFS : >_
```

Close():

This function is used to close an open file.

The input to the function is file name which is string.

Syntax: close file_name.

Example: close Demo.txt will give below output,

```
C:\Windows\system32\cmd.exe - e

CVFS : >man close
Description: Used to close opened file
Usage: close File_name

CVFS : >close Demo.txt
File closed successfully

CVFS : >
```

Closeall():

This function is used to close all the open files.

Syntax: closeall.

Example: when closeall command is used below will be the output,

```
C:\Windows\system32\cmd.exe - e

CVFS : >man closeall
Description: Used to close all opened file
Usage: closeall

CVFS : >closeall
All files closed successfully

CVFS : >_
```

WriteFile():

This function is used to write data into file.

Syntax: write file_name

Example: write Demo.txt – enter data –

```
C:\Windows\system32\cmd.exe - e

CVFS : >man write
Description: Used to write new regular file
Usage: write File_name
After this enter the data that we want to write

CVFS : >write Demo.txt
Enter the data :
Data from Demo file

CVFS : >
```

ReadFile():

This function is used to read the contents of a file.

Syntax: read file_name no_of_bytes_to_read.

Example: read Demo.txt 19,

```
C:\Windows\system32\cmd.exe - e

CVFS : >man read
Description: Used to read new regular file
Usage: read File_name No_Of_Bytes_To_Read

CVFS : >read Demo.txt 19
Data from Demo file
CVFS : >_
```

Lseek():

This function is used to change the offset of file.

It accepts file name which is string, offset integer and start point as integer.

Syntax: lseek file_name ChangeInOffset StartPoint.

Example: lseek Demo.txt 5 1 will give output as below,

```
C:\Windows\system32\cmd.exe - e

CVFS : >write Demo.txt
Enter the data :
Information about Demo file

CVFS : >man lseek
Description: Used to change file offset
Usage: lseek File_name ChangeInOffset StartPoint

CVFS : >lseek Demo.txt 5 1

CVFS : >read Demo.txt 22
Information about Demo file
CVFS : >_
```

Truncate():

This function is used to remove all the data from the existing file.

Syntax: truncate file_name

Example: truncate Demo.txt will given below output,

```
C:\Windows\system32\cmd.exe - e

CVFS : >create Demo.txt 3
File is successfully created with file descriptor : 0

CVFS : >write Demo.txt
Enter the data :
This is demo data

CVFS : >read Demo.txt 17
This is demo data
CVFS : >man truncate
Description: Used to remove data from file
Usage: truncate File_name

CVFS : >truncate Demo.txt

CVFS : >read Demo.txt 17

CVFS : >
```

Stat():

This command is used to display all the statistical information about the file.

Syntax: stat file_name.

Example: stat Demo.txt will give below output,

```
C:\Windows\system32\cmd.exe - e

CVFS : >man stat
Description: Used to display information of file
Usage: stat File_name

CVFS : >stat Demo.txt

-----Statistical Information about file-----
File name : Demo.txt
Inode Number : 1
File Size : 2048
Actual File Size : 19
Link Count : 1
Reference Count : 1
File Permission : Read & Write
-----

CVFS : >_
```

Fstat():

This command is used to display all the statistical information about the file using file descriptor.

Syntax: fstat file_descriptor.

Example: fstat Demo.txt will give below output,

```
C:\Windows\system32\cmd.exe - e

CVFS : >man fstat
Description: Used to display information of file
Usage: fstat File_Descriptor

CVFS : >fstat 0

-----Statistical Information about file-----
File name : Demo.txt
Inode Number : 1
File Size : 2048
Actual File Size : 24
Link Count : 1
Reference Count : 1
File Permission : Read & Write
-----

CVFS : >_
```

ls():

This function is used to display all the files in the directory.

Syntax: ls.

Example: ls will give below output,

```
C:\Windows\system32\cmd.exe - e

CVFS : >ls

File Name      Inode number   File size      Link count
-----
Demo.txt       1              24             1
Hello.txt      2              0              1
-----

CVFS : >_
```

rm():

This command is used to delete the file from the directory.

Syntax: rm file_name.

Example: rm Hello.txt will give below output,

```
C:\Windows\system32\cmd.exe - e

CVFS : >ls

File Name      Inode number   File size      Link count
-----
Demo.txt       1              24             1
Hello.txt      2              0              1
-----

CVFS : >rm Hello.txt

CVFS : >ls

File Name      Inode number   File size      Link count
-----
Demo.txt       1              24             1
-----

CVFS : >_
```


Commands related to File handling:

- ls:
 - Name: List directory contents, Syntax: ls [option] [file].
 - Usage: To list information about the files (current directory by default).
- ls -l:
 - use a long listing format
- ls -a:
 - do not ignore entries starting with .
- rm:
 - Name: remove files or directories. Syntax: rm [option] [file].
 - Usage: rm removes each specified file. By default, it does not remove directories.
- Cat:
 - Name: Concatenate, Syntax: cat [option] [file].
 - Usage: concatenate files and print on the standard output.
- cd:
 - Name: Change working directory, Syntax: cd [directory]
 - Usage: The cd utility shall change the working directory of the current shell execution environment.
- Chmod:
 - Name: chmod, fchmod, fchmodat - change permissions of a file.
 - Usage: The chmod() and fchmod() system calls change a file's mode bits. (The file mode consists of the file permission bits plus the set-user-ID, set-group-ID, and sticky bits.)
- cp:
 - Name: copy files and directories, Syntax: cp [option]... source... directory.
 - Usage: Copy SOURCE to DEST, or multiple SOURCE(s) to DIRECTORY.
- df:
 - Name: report file system disk space usage, Syntax: df [option] [file].
 - Usage: df displays the amount of disk space available on the file system containing each file name argument.
- find:
 - Name: search for files in a directory hierarchy, Syntax: find [-H] [-L] [-P] [-D debugopts] [-Olevel] [starting-point...][expression].
 - Usage: find searches the directory tree rooted at each given starting-point by evaluating the given expression from left to right, according to the rules of precedence (see section OPERATORS), until the outcome is known (the left hand side is false for and operations, true for or), at which point find moves on to the next file name. If no starting-point is specified, '.' is assumed.
- grep:
 - Name: print lines that match patterns, Syntax: grep [option] patterns [file].
 - Usage: grep searches for PATTERNS in each FILE. PATTERNS is one or more patterns separated by newline characters, and grep prints each line that matches a pattern. Typically PATTERNS should be quoted when grep is used in a shell command.

- **ln:**
 - Name: make links between files, Syntax: ln [option]... target... directory.
 - Usage: create links to each TARGET in DIRECTORY. Create hard links by default, symbolic links with --symbolic.
- **mkdir:**
 - Name: make directories, Syntax: mkdir [option] [file].
 - Usage: Create the DIRECTORY(ies), if they do not already exist.
- **pwd:**
 - Name: print name of current/working directory, Syntax: pwd[option].
 - Usage: Print the full filename of the current working directory.
- **touch:**
 - Name: change file timestamps, Syntax: touch[option] file.
 - Usage: Update the access and modification times of each FILE to the current time.
- **uname:**
 - Name: print system information, Syntax: uname[option].
 - Usage: Print certain system information. With no OPTION, same as -s.
- **stat:**
 - Name: stat, fstat, lstat, fstatat - get file status.
 - Usage: These functions return information about a file, in the buffer pointed to by statbuf.
- **man:**
 - Name: an interface to the system reference manuals, Syntax: man [man options] [[section] page].
 - Usage: man is the system's manual pager. Each page argument given to man is normally the name of a program, utility or function. The manual page associated with each of these arguments is then found and displayed.
- **mkfs:**
 - Name: build a Linux filesystem, Syntax: mkfs [options] [-t type] [fs-options] device [size].
 - Usage: mkfs is used to build a Linux filesystem on a device, usually a hard disk partition. The device argument is either the device name (e.g., /dev/hda1, /dev/sdb2), or a regular file that shall contain the filesystem. The size argument is the number of blocks to be used for the filesystem.

←-----End-----→