CVFS (Custom Virtual File System) Documentation

**Project Overview**

* **Name of the Project**

CVFS (Custom Virtual File System)

* **Technology Used**

Programming Language: C

File System Concepts

Operating System Concepts

* **User Interface Used**

Command Line Interface (CLI)

* **Platform Required**

Linux

* **Hardware Requirements**

Standard x86/x64 architecture

Minimum 2GB RAM

Minimum 100MB disk space

* **Description of the Project**

The Custom Virtual File System (CVFS) is designed to simulate a file system for educational purposes. It includes basic file operations such as creation, deletion, reading, and writing files. CVFS allows users to understand the underlying mechanisms of a file system by providing a simple and interactive CLI.

* **Data Structures**

**Data Structures Used in the Project**

Superblock

Inode

Directory

File Table

User File Descriptor Table

* **Diagram of Data Structures**
* **Project Flow**

**The Flow of the Project**

Initialization of the file system

User interacts with the CLI to perform file operations

Each command triggers a function to manipulate the data structures

The changes are reflected in the virtual file system

* **Screenshot of Output**

To demonstrate each feature of the project, here are some example commands and their corresponding outputs:

## Questions and Answers

### 1. What is meant by file system?

A file system is a method and data structure that an operating system uses to manage files on a disk or partition; it organizes data into files and directories.

### 2. Which file systems are used by Linux and Windows operating systems?

* Linux: Ext4, Btrfs, XFS, and others.
* Windows: NTFS, FAT32, exFAT.

### 3. What are the parts of the file system?

* Superblock
* Inode table
* Directory structure
* Data blocks
* File descriptor table

### 4. Explain UAREA and its contents.

UAREA (User Area) contains user-specific data such as open file tables, user ID, and other process-specific information. It is used by the OS to manage user processes and their interactions with the file system.

### 5. Explain the use of the File Table and its contents.

The File Table keeps track of all open files, containing information such as inode number, file pointer, and other file-specific data. It allows the OS to manage file operations efficiently.

### 6. Explain the use of InCore inode Table and its use.

The InCore inode Table stores the in-memory representations of inodes, allowing quick access to file metadata without constantly reading from disk.

### 7. What does inode mean?

An inode (index node) is a data structure on a filesystem that stores information about a file or directory, such as its size, ownership, permissions, and data block locations.

### 8. What are the contents of Superblock?

The Superblock contains metadata about the filesystem, including total inodes, free inodes, total blocks, free blocks, and other critical filesystem information.

### 9. What are the types of files?

* Regular files
* Directories
* Character devices
* Block devices
* FIFOs (named pipes)
* Sockets
* Symbolic links

### 10. What are the contents of the inode?

The inode contains:

* File type and permissions
* Owner and group IDs
* File size
* Timestamps (creation, modification, access)
* Link count
* Pointers to data blocks

### 11. What is the use of a directory file?

A directory file maps filenames to inodes, organizing files into a hierarchical structure that allows users to navigate and manage their files efficiently.

### 12. How does the operating system maintain security for files?

The OS maintains security through file permissions, user and group ownership, and access control lists (ACLs), ensuring only authorized users can access or modify files.

### 13. What happens when a user wants to open the file?

When a user wants to open a file, the OS locates the file's inode, creates an entry in the file table, and returns a file descriptor to the user.

### 14. What happens when a user calls lseek system call?

The lseek system call changes the file offset for the open file descriptor, allowing subsequent read/write operations to occur at a different location within the file.

### 15. What is the difference between library function and system call?

* Library function: High-level API provided by libraries (e.g., fopen, fread).
* System call: Low-level function provided by the OS kernel (e.g., open, read).

### 16. What is the use of this project?

The CVFS project provides an educational tool for understanding file system concepts and operations, allowing users to interact with a simulated file system.

### 17. What are the difficulties that you faced in this project?

* Managing memory and data structures efficiently
* Ensuring data consistency and integrity
* Simulating file system behavior accurately

### 18. Is there any improvement needed in this project?

Possible improvements include:

* Implementing persistent storage for the file system
* Enhancing the CLI with more features and better error handling
* Adding support for more file operations and metadata

### 19. Explain the internal working of below system calls

#### open

* **Internal Working**: Validates the file path, allocates an inode if the file does not exist (for create mode), initializes a file descriptor, and returns it.

#### close

* **Internal Working**: Releases the file descriptor, updates file metadata, and ensures all pending writes are flushed.

#### read

* **Internal Working**: Reads data from the file's data blocks into a buffer, updates the file pointer, and returns the number of bytes read.

#### write

* **Internal Working**: Writes data from the buffer to the file's data blocks, updates the file pointer and file size, and returns the number of bytes written.

#### lseek

* **Internal Working**: Modifies the file pointer based on the offset and whence parameters, allowing for random access within the file.

#### stat

* **Internal Working**: Retrieves and returns the file's metadata stored in its inode, such as size, permissions, and timestamps.

#### chmod

* **Internal Working**: Updates the file's inode to change its permissions based on the specified mode.

#### unlink

* **Internal Working**: Removes the file from the directory, decreases the link count in the inode, and deallocates the inode if the link count reaches zero.

### Explanation and Demonstration of Commands

#### ls

* **Use**: Lists directory contents
* **Command**: ls

#### ls -l

* **Use**: Lists directory contents in long format
* **Command**: ls -l

#### ls -a

* **Use**: Lists all directory contents, including hidden files
* **Command**: ls -a

#### rm

* **Use**: Removes files or directories
* **Command**: rm filename

#### cat

* **Use**: Concatenates and displays file content
* **Command**: cat filename

#### cd

* **Use**: Changes the current directory
* **Command**: cd dirname

#### chmod

* **Use**: Changes file permissions
* **Command**: chmod 755 filename

#### cp

* **Use**: Copies files or directories
* **Command**: cp source destination

#### df

* **Use**: Displays disk space usage
* **Command**: df

#### find

* **Use**: Searches for files in a directory hierarchy
* **Command**: find /path -name "filename"

#### grep

* **Use**: Searches for patterns in files
* **Command**: grep "pattern" filename

#### ln

* **Use**: Creates hard and symbolic links
* **Command**: ln -s source linkname

#### mkdir

* **Use**: Creates directories
* **Command**: mkdir dirname

#### pwd

* **Use**: Prints the current working directory
* **Command**: pwd

#### touch

* **Use**: Updates file timestamps or creates empty files
* **Command**: touch filename

#### uname

* **Use**: Prints system information
* **Command**: uname -a

#### stat

* **Use**: Displays file or filesystem status
* **Command**: stat filename

#### man

* **Use**: Displays the manual for commands
* **Command**: man command

#### mkfs

* **Use**: Creates a filesystem on a device
* **Command**: mkfs -t ext4 /dev/sdX