Customised Virtual File System

- **Documentation for the Customised VFS project:**

1.Name of the project:

“Customised Virtual File System”.

2.Technology used:

System Programming using C

3.User interface used

Command User Interface

4.Platform required:

Windows NT platform OR Linux Distributions

5.Architectural requirements:

Intel 32 bit processor

6.Description of the project:

In this project we emulate all data structures which are used by operating system to manage File system oriented tasks. • As the name suggest its virtual because we are going to maintain all records in Primary storage. • In this project we create all data structures which required for File Subsystems as 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 our own data structures by referring Algorithms of UNNIX operating system. • By using this project we can get overview of UFS (UNIX File System) on any platform.

- # Questions and answers regarding File system

1. What is mean by file system?

A file system is a process of managing how and where data on a storage disk, which is also referred to as file management or FS. It is a logical disk component that compresses files separated into groups, which is known as directories. It is abstract to a human user and related to a computer; hence, it manages a disk's internal operations. Files and additional directories can be in the directories.

A file is a collection of correlated information which is recorded on secondary or non-volatile storage like magnetic disks, optical disks, and tapes. It is a method of data collection that is used as a medium for giving input and receiving output from that program. In general, a file is a sequence of bits, bytes, or records whose meaning is defined by the file creator and user. Every File has a logical location where they are located for storage and retrieval.

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

- File systems used by Linux OS:

When we install linux OS, Linux offers the file systems such as Ext, Ext2, Ext3,Ext4, JSP, ReiserFS, XFS, btrfs ans swap.

1. Ext, Ext2, Ext3 and Ext4 file System:
   The file system Ext stands for Extended File System
2. JSP file System:
   JFS stands for Journaled File System, and it is developed by IBM for AIX Unix. \
3. ReiserFS file System:
   ReiserFS is an alternative to the Ext3 file system. It has improved performance and advanced features.
4. XFS file system:
   XFS file system was considered as high-speed JFS, which is developed for parallel I/O processing. NASA still using this file system with its high storage server (300+ Terabyte server).

5. Btrfs file system:
   stands for the B tree file system. It is used for fault tolerance, repair system, fun administration, extensive storage configuration, and more. It is not a good suit for the production system.
6. swap file system :
   The swap file system is used for memory paging in Linux operating system during the system hibernation. A system that never goes in hibernate state is required to have swap space equal to its RAM size.

- File systems used by Windows OS:

five types of Windows file system, such as FAT12, FAT16, FAT32, NTFS and exFAT.

1. FAT32: In order to overcome the limited volume size of FAT16 (its supported maximum volume size is 2GB) Microsoft designed a new version of the file system FAT32, which then becomes the most frequently used version of the FAT (File Allocation Table) file system.

2. NTFS: NTFS is the newer drive format. Its full name is New Technology File System. Starting with Windows NT 3.1, it is the default file system of the Windows NT family. Microsoft has released five versions of NTFS, namely v1.0, v1.1, v1.2, v3.0, and v3.1.

3. exFAT : exFAT (Extended File Allocation Table) was designed by Microsoft back in 2006 and was a part of the company's Windows CE 6.0 operating system. This file system was created to be used on flash drives like USB memory sticks and SD cards, which gives a hint for its precursors: FAT32 and FAT16.

4.Explain UAREA and its contents.

Explanation by taking an example:
We have written one program named “marvellous.c “which opens “demo.txt” file in READ mode.(Program manipulation)

1. after compiling “marvellous.c” will creates one executable file i.e “marvellous.exe” (this will considered as a process)

2. All record regarding(which are needed) this executable file will be stored in one location which known as UAREA.(User area)
3. (This is not an address space) happens Through UAREA O.S does interaction with the peocess
4. Inside UAREA there is an array of pointers wich is know as UFDT(User file descriptor table). Inside this array. First 3 File descriptors are reserved(index no. 0 for stdin, 1 for stdout, 2 for stderror in java system.in(Keyboard), system.out(monitor/screen) and system.err)
5. If we open any file in the process, will get first index number as 3 to use

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

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

IIT contains inode number, block number and permission
(Read/write/readwrite)

The inode is a data structure that describes everything about a file other than their name. When a file is opened then the kernel copies the inode into memory. As the file changes, the in-core inode is updated usually more often than the on-disk copy. And the in-core inode has a few extra fields that are only needed while the file is opened.

7.What is mean by inode?

Index wise stored nodes means Inode. Inode is most important data structure which stored metadata of file

8.What are the contents of Superblock?

(tatal inode, total free inode, total block, total free blocks)

Size of super block is 1Kb.

It stores all information of O.S (reletd with hard disc), total count of inodes and count of free inodes from those inodes(e.g if you have 100 inodes and ypu have created 20 files then number of free inodes are 80)

Total number of inodes are fixed, We can create 1 file per inode (To create a file we need inode), we can get these all information from Super block

When we create one new file count of free inode will decrease by one.

Superblock also contains total number of block and number free block. But there can be multiple blocks with one file().

9.What are the types of files?

S_ISREG(): regular file

S_ISDIR() :directory file

S_ISCHR(): character special file

S_ISBLK() :block special file

S_ISFIFO(): pipe or FIFO

S_ISLNK() :symbolic link

S_ISSOCK(): socket

10.What are the contents of the inode?

i-node contains the owner of the file, the size of the file, pointers to where the actual data blocks for the file are located on disk, and so on.DILB(Disk inode list block): Internally its doubly circular linked list. Initially, if we have total inodes 100. Then LL will contain 100 ready inodes which are empty.

11.What is the use of a directory file?

A file that contains the names of other files and pointers to information on these files. Any process that has read permission for a directory file can read the contents of the directory, but only the kernel can write directly to a directory file. Processes must use the functions described in this chapter to make changes to a directory.

12.How the operating system maintains security for files?

1. Access Control plays a huge part in file system security.The system should only allow access to files that the user is permitted to access ¬ Almost all major file systems support ACL's or capabilities in order to prevent malicious activity on the file system ¬ Depending on the users rights they can be allowed to read, write and/or execute and object. In some file systems schemes only certain users are allowed to alter the ACL on a file or see if a file even exists. ¬ Ultimately the less the user has access to the less that can go wrong and the integrity of the disk can be more guaranteed.
2. Encryption is also a method used by file systems to secure data File system encryption has a few advantages over full disk encryption for example, File based key management

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

A successful call to open returns a "file descriptor" which is merely an integer in the context of the user process. This descriptor is consequently passed to any call that interacts with the opened file

It is important to note that the call to open acts like a validation point at which various checks are made. If not all of the conditions are met, the call fails by returning -1 instead of the descriptor, and the kind of error is indicated in Error number.

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

lseek() system call repositions the read/write file offset i.e., it changes the positions of the read/write pointer within the file. In every file any read or write operations happen at the position pointed to by the pointer. lseek() system call helps us to manage the position of this pointer within a file.

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

System call is a request to the kernel to access a resource while library call is a request to use a function defined in a programming library.

16.System calls and their work:

1.Open: Opening or creating a file can be done using the system call open. The syntax is:

```
#include <sys/types.h>
#include <sys/stat.h>
#include <fcntl.h>
int open(const char *path,
        int flags,... /* mode_t mod */);
```

2.Close: For closing a file and thus eliminating the assigned descriptor we use the system call *close*.

```
#include <unistd.h>
int close(int fd);
```

The function returns 0 in case of success and -1 in case of an error. At the termination of a process an open file is closed anyway.

3.read : When we want to read a certain number of bytes starting from the current position in a file, we use the *read* call. The syntax is:

```
#include <unistd.h>
ssize_t read(int fd, void* buf, size_t noct);
```

4.write: For writing a certain number of bytes into a file starting from the current position we use the *write* call. Its syntax is:

```
#include <unistd.h>
ssize_t write(int fd, const void* buf, size_t noct);
```

5.lseek: To position a pointer (that points to the current position) in an absolute or relative way can be done by calling the *lseek* function. Read and write operations are done relative to the current position in the file. The syntax for *lseek* is:

```
#include <sys/types.h>
#include <unistd.h>
off_t lseek(int fd, off_t offset, int ref);
```

6. STAT, LSTAT and FSTAT: In order to obtain more details about a file the following system calls can be used: *stat*, *lstat* or *fstat*.

```
#include <sys/types.h>
#include <sys/stat.h>
int stat(const char* path, struct stat* buf);
int lstat(const char* path, struct stat* buf);
int fstat(int df, struct stat* buf);
```

7.chmod: To modify the access rights for an existing file we use:

```
#include <sys/types.h>
#include <sys/stat.h>
int chmod(const char* path, mode_t mod);
```

8.unlink: To delete a link (a path) in a directory we can use the *unlink* system call. Its syntax is:

```
#include <unistd.h>
int unlink(const char* path);
```

17.Commands with explanation:

1. **Ls:** This command will list the files stored in a directory. To see a brief, multi-column list of the files in the current directory

2. **ls – l**: displays the detailed information of all files and folders that are not hidden, and displays them as a list.

3. **ls – a:** displays all files and folders in the current directory, including all hidden files and folders

4. **rm:** This command will remove (destroy) a file. You should enter this command with the -i option, so that you'll be asked to confirm each file deletion. To remove a file named junk, enter:

5. **cat**: use to read brief files or to concatenate files together

6. **cd**: to change current directory location

7. **chmod**: This command changes the permission information associated with a file

8. **cp**: This command copies a file, preserving the original and creating an identical copy

9. **df** :This command reports file system disk usage (that is, the amount of space taken up on mounted file systems). For each mounted file system, df reports the file system device, the number of blocks used, the number of blocks available, and the directory where the file system is mounted.

10. **Find**: The find command lists all of the files within a directory and its subdirectories that match a set of conditions. This command is most commonly used to find all of the files that have a certain name.

11. **Grep**: used to search for a string of characters in a specified file

12. **Ln**: The **use** of a hard link allows multiple filenames to be associated with the same file since a hard link points to the inode of a given file, the data of which is stored on disk.

13. **mkdir**: To create a subdirectory in the current directory

14. **pwd**: This command reports the current directory path.

15. **Touch**: use to change and modify timestamps of a file

16. **Uname**: used to determine the processor architecture, the system hostname and the version of the kernel running on the system

17. **Stat**: use to displays detailed information about given files or file systems.

18. **man**: This command displays the manual page for a particular command. If you are unsure how to use a command or want to find out all its options, you might want to try using man to view the manual page.

19. **mkfs :** The mkfs command stands for “make file system” is utilized to make a file system (which is, a system for organizing a hierarchy of directories, subdirectories, and files) on a formatted storage device usually, a partition on a hard disk drive (HDD) or it can also be a USB drive, etc.