Question:

1. **What is mean by file system?**

Ans. File system means making environment for storing our data of our file on hard disk in such a way that the application can retrieve it back from the hard disk and display it to the end user in an understandable format.

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

   Ans. NTFS (New Technology File System), FAT32 (File Allocation Table 32) are windows' file systems.

   ext3 (Extended 3) is used for Linux distros (i.e. Linux versions)

3. **What are the parts of the file systems?**

   Ans. A file system consists of a superblock (SB), a collection of inodes (DILB), a collection of data blocks allocated on a disk(DB).

4. **Explain UAREA and it's contents.**

   Ans. The user area (UAREA) of each process contains information that the process uses when it is running.

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

   Ans. The information about the current state of the file is maintained in the file table. For example if the file is being written the information about the current cursor position is kept in the file table. This file table also checks whether the accessing process has access to that file or not.

   The contents of file table: offset of the file, mode of opening the file, pointer to the IIT, etc.

6. **Explain use of In Core inode Table and its use**.

Ans. Each inode for an open file is stored in an "in-core" inode table entry with a reference count. Process accesses file through file descriptors which point to open file table entries which point to inode table entries. There is one inode table entry for each opened file. Forked processes sharing offset into an open file have file descriptors pointing to same opened file table entry. Multiple processes with the same file opened point to separate opened file table entries, but they point to the same inode table entry.

7. What is mean by inode?

Ans. The inode is the indexed node, which maps the address of a file to the logical block number in the hard disk.

8. What are the contents of super block?

Ans.

- Size of File System.
- Number of free blocks in File System.
- List of available free blocks.
- Index of next free blocks.
- Size of inode list.
- Number of free inode.
- List of free inode.
- Index of next free inode.
- Lock fields for free blocks and free inodes.
- A flag which indicates that the super block is modified or not.

9. What are the types of files?

Ans. Regular files and special files.

10. What are the contents of inodes?

Ans.

- Owner identifier.
- Group identifier.
- Permission.
- Last file access time.
- inode last modified time.
- File last modified time.
- Link count.
- File size.
- Table of disk address. (13 member array)
- File Type.

11. What is the use of the directory files?

Ans. The directory contain a list of file names, each of which maps to an inode number.

12. How Operating system maintains the security of files?

Ans.

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

Ans. When an user wants to open a file, a process is generated. The process accesses the file through file descriptors from UFDT. The file descriptor points to opened file table entries. Each file table entry points to incore inode table entry. Each IIT entry is for a separate opened file, while each FT entry is for a separate forked process, the forked processes sharing an offset into an opened file have file descriptor pointing to the same opened FT entry.

14. What happens when user calls lseek system call?

Ans.

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

Ans. System calls: (functions provided by the kernel)

Library calls: (functions within program libraries)

16. What is the use of this project?

Ans. This is a research based project which let us understand the working of an operating system. There are 5 main tasks of any OS - Process Management, Memory Management, CPU Scheduling, Hardware abstraction and File Management, of which file management is what we are trying to learn through this project. Here, we tried to make a rough replica of Unix File System.

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

Ans. Basically this project is very simple coding wise, as we have not included any complex algorithm in it, however working on the concepts and to understand it is troublesome. I had to gather lots of information about the OS and its working.

18. Is there any improvement needed in this project?

Ans.

Internal working of the system calls.

1. **Open**: The **open**() system call opens the file specified by *pathname*. If the specified file does not exist, it may optionally (if **O_CREAT** is specified in *flags*) be created by **open**(). The return value of **open**() is a file descriptor, a small, nonnegative integer that is used in subsequent system calls (read(2), write(2), lseek(2), fcntl(2), etc.) to refer to the open file. The file descriptor returned by a successful call will be the lowest-numbered file descriptor not currently open for the process.

2. **Close**: It closes a file descriptor, so that it no longer refers to any file and may be reused. Any record locks held on the file it was associated with, and owned by the process, are removed (regardless of the file descriptor that was used to obtain the lock). If *fd* is the last file descriptor referring to the underlying open file description, the resources associated with the open file description are freed; if the file descriptor was the last reference to a file which has been removed using unlink(), the file is deleted.

3. **Read**: It attempts to read up to *count* bytes from file descriptor *fd* into the buffer starting at *buffer.*

4. **Write**: It writes up to *count* bytes from the buffer starting at *buffer* to the file referred to by the file descriptor *fd*.

5. **Lseek**: It allows the file offset to be set beyond the end of the file (but this does not change the size of the file). If data is later written at this point, subsequent reads of the data in the gap return null bytes ('\0') until data is actually written into the gap.

Explain use of below commands

1) ls

ls is a command to list computer files in Unix and Unix-like operating system.

2) ls - l

long format, displaying Unix file types, permissions, number of hard links, owner, group, size, last-modified date and filename.

3)ls -a

4) rm

rm command is used to remove objects such as files, directories, symbolic links and so on from the file system like UNIX**.**
syntax :- rm fileName

5) cat

The cat (short for "concatenate") command is one of the most frequently used command in Linux/Unix like operating systems. cat command allows us to create single or multiple files, view contain of file, concatenate files and redirect output in terminal or files.
syntax :- cat [option] [FILE]

6) cd

The cd command, also known as chdir (change directory), is a command-line OS shell command used to change the current working directory in operating systems such as Unix
syntax :- cd path

7) chmode

chmod is the command and system call which may change the access permissions to file system objects.
Syntax**:** chmod [-Rcvf] MODE[,MODE]... FILE...

8) cp

cp is a command in various Unix and Unix-like operating systems for copying files and directories.
Syntax**:** cp [-fip] [-R [-H|-L|-P]] SOURCE DEST

9) df

df is a standard Unix command used to display the amount of available disk space for file systems on which the invoking user has appropriate read access.
Syntax**:** df [-hiklnP] [-t type] [[file|file_system] ...]

10) find

find is a command-line utility that searches one or more directory trees of a file system, locates files based on some user-specified criteria and applies a user-specified action on each matched file.
Syntax**:** find [-H] [-L] [-P] [-D debugopts] [-Olevel] [starting-point...] [expression]

11) grep

The grep command which stands for "global regular expression print," processes text line by line and prints any lines which match a specified pattern. The grep command is used to search text or searches the given file for lines containing a match to the given strings or words.
Syntax**:** grep [-HhrilLnqvsoweFEABCz] PATTERN [FILE]...

12) ln

The ln command is a standard Unix command utility used to create a hard link or a symbolic link to an existing file.
Syntax**:** ln [OPTIONS] TARGET... LINK|DIRECTORY

13) mkdir

mkdir is used to make a new directory.
Syntax**:** mkdir [-p] [-m mode] directory ...

14) pwd

the pwd command writes the full pathname of the current working directory to the standard output. The command is a shell builtin in most Unix shells.

Syntax**:** pwd [OPTION]...

15) touch

ouch is a command in Unix and Unix-like operating systems used to update the access date and/or modification date of a computer file or directory.

Syntax**:** touch [-c] [-d DATE] FILE [FILE]...

16) uname

uname is a computer program in Unix and Unix-like computer operating systems that prints the name, version and other details about the current machine and the operating system running on it.

Syntax**:** uname [-amnrspv]

17) stat

stat is a Unix system call that returns file attributes about an inode.

Syntax**:** stat [-FLnq] [-f format|-l|-r|-s|-x] [-t timefmt] [file ...]

18) man

man stands for Manual. it display the all information about specific command.

man CommandName

19) mkfs

The mkfs (i.e., make filesystem)command is used to create a filesystem.

Syntax mkfs [-V] [-t fstype] [fs-options] filesys [blocks]