MODULE - II

The UNIX File System

MODULE 2

The Unix Files System

Module Description

The main goal of studying internal representation of files is to acquire conceptual understanding of file system, especially data structure and related terms. This knowledge will help you become a successful system administrator. In this module we will describe internal representation of UNIX file system, how files are organised in UNIX system, how system calls interact with file system. To get familiar with UNIX file system, commands and system calls related file system are introduced.

By the end of this module, students will learn organization of UNIX file system. At the end of this module a user would be able to create and manipulate files using shell commands and system calls and mounting the file system. The conceptual knowledge can be applied to restore file system in an emergency situation.

Chapter Table of Contents

Chapter 2.1

Working with UNIX File System

Aim

To acquire basics and structure of UNIX file system

Instructional Objectives

After completing this chapter, you should be able to:

- Explain the different types of files in UNIX*
- Describe the structure of a regular file
- Explain the directory structure of the UNIX file system*
- Explain the various disk allocation methods
- Describe inode along with its importance
- Define superblock and the process of freeing a superblock
- Illustrate the steps to convert pathname to an inode
- Discuss in detail how to assign inode to a new file

Learning Outcomes

At the end of this chapter, you are expected to:

- Identify the differences between various file types available in UNIX
- Outline the features of a regular file
- Illustrate how UNIX file system can be represented with hierarchical tree structure
- Describe how each block on a disk is assigned
- Prepare a list of various types of information stored in inode
- Discuss the purpose of super block
- Write an algorithm to convert pathname to an inode
- Identify the need of inode assignment

2.1.1 Types of Files

The types of files available in UNIX are

- Regular files
- Directory files
- Device Files
- Hidden files

The best way to learn about files is to play with them. UNIX looks everything as a file. If you write a program, you add one more file in the system. Files grow rapidly in a system, and if they are not organised properly, you will find it difficult to locate them. UNIX organises its file in different directories.

UNIX file system is simple and conceptually clean. It allow user to access other user's file but at the same time it is secure too. In this chapter, you will conceptually understand UNIX file system.

File: Files are collection of data items stored on disk. Whatever you store in computer is in the form of a file. Files are always associated with storage devices like hard disk, CD-ROM etc. File is the last object in your file system tree.

Directory: Directory is a group of files. Directories are used to organize your data files more efficiently. Directory can be classified in into two types:

- **Root directory**- It is the top directory in file system and is denoted by / (forward slash). There is only one root directory in the file system which can't be renamed or deleted.
- **Sub directory** – These Directories can be created and renamed. These directories exists under the root (/) directory.

UNIX/Linux File system:

A UNIX file system is nothing but a logical collection of files and directories stored in a partition or disk. Everything in UNIX is considered as a file, including physical devices such as DVD-ROMs, USB devices, floppy drives. Apart from storing the user data which is available in the form of files, a file system also stores some structural information about file system in the form of superblock, inodes, and directories.

Features of UNIX File System:

The UNIX system organizes its files using an upside-down hierarchical tree structure. This hierarchical component adds dynamic flexibility of the file –system. Size of a file is not determined by any rule other than the amount of disk-storage that is available on the system.

Files are protected by file-ownership mechanism. Only a specific class of user can access certain files. Internal format of a file is dictated by the utility that creates it. So, UNIX files are structure less.

Type of Files:

In UNIX files are divided into different categories:

- **Regular files:**

It is also known as ordinary files. An ordinary file consists of sequential series of bytes, which occupy disk space. These files are the 'leaf-node' in the tree structured hierarchy. UNIX imposes no rules regarding internal format of a regular file. This type of file is structure less, which is a major advantage of UNIX system. All the programs that you write belong to this type. An ordinary file can be divided into two types:

- **Text file:**

Contains only printable character and you can often view the contents Binary file: contains both printable and unprintable character. Most of the UNIX commands, executables, pictures, sound and video files are binary files. If you try to display the content of binary file using cat command, the output would be unreadable.

- **Directory files:**

Directory contains files and other directories. UNIX file system is organized with a number of directories, subdirectories and you can also create them as and when needed.

A directory file maintains the information about every file and subdirectory it contains. For each file, a separate entry is maintained. If you have 40 files in a directory, there will be 40 entries in directory file. Each entry gives you information about the filename and inode number (A unique identification number for the file or directory).A directory contains filename and not the file's content: you cannot write a directory file, but you can perform some action like

when you create or remove a file, kernel automatically updates the corresponding directory by adding or removing the entry (inode number and filename).

- **Device Files:**

UNIX/Linux treats hardware device as a file. This device file acts as an interface to the user so that he need not to get in technical details about hardware. These special files are interface for a device driver that appears in a file system. Some device files reads/writes a character by character (1 byte at a time). These files are known as character device/special files. Example of character special files are: Virtual terminals, terminals and serial modems etc. The devices that read one block at a time (One block – 512 bytes to 32 KB) are called block device/special files. The example of block special files are HDD and other memory regions.

All device files are stored in /dev directory. Use cd and ls command to browse the directory: $ cd /dev/ $ ls -l

- **Hidden files:**

Hidden files are the one that are not listed by ls command. In case of a hidden file, the filename begin with a dot (.). Example of hidden file is .profile which is executed every time you log in to the system. A user can see hidden files in the system by using –a option to the ls command. For example:

$ ls –a

Self-assessment Questions

1) Binary files contain contains only printable character.
   a) True b) False

2) Regular/ ordinary files are structured files
   a) True b) False

3) A directory contains.....
   a) Filename and its contents b) Filename and its inode
   c) Filename d) Inode

4) Filename of hidden files starts with.....
   a) . b) /
   c) ./ d) ..

2.1.2 Structure of regular files:

In the above sections we discussed about the different files in UNIX file system and also observed the directory structure in a UNIX file system. As we have already discussed that In UNIX a file and its attributes are identified by an inode. Inode is also known as index number. Actually, an inode is a data structure on a traditional UNIX file system that stores basic information and description of the file data and its layout on disk.

Now, we will discuss in detail about information stored in inode and the way it is represented in the kernel.

Inodes are stored on the disk. The kernel reads an inode into a memory which we can call as in-core inodes. Disk inodes contains the following information:

- File access permissions and time (last access / modified etc.)
- File ownership information.
- Type of the file (regular / directory / block special / pipe)
- Number of links to the file

- File size and organization on disk (the file data may spread across several different and far-spaced disk location)

The In-core copy of inodes contains all of the above information, but it also contains the following additional information:

- Status (locked / process is waiting for it to become unlocked / in-core copy has been modified and thus differs from the copy on the disk / mounted)
- Logical Device number of the file system
- Inode Number. Since inodes are stored in a sequential manner on the disk, the kernel uses an identifier of that array to refer to its in-core copy.
- Pointer to other in-core inodes. Kernel maintains a hash queue of inodes according to the logical device number and the inode numbers. Kernel also maintains a list of free inodes.
- Reference count which indicates the number of instances of the file that are currently active.

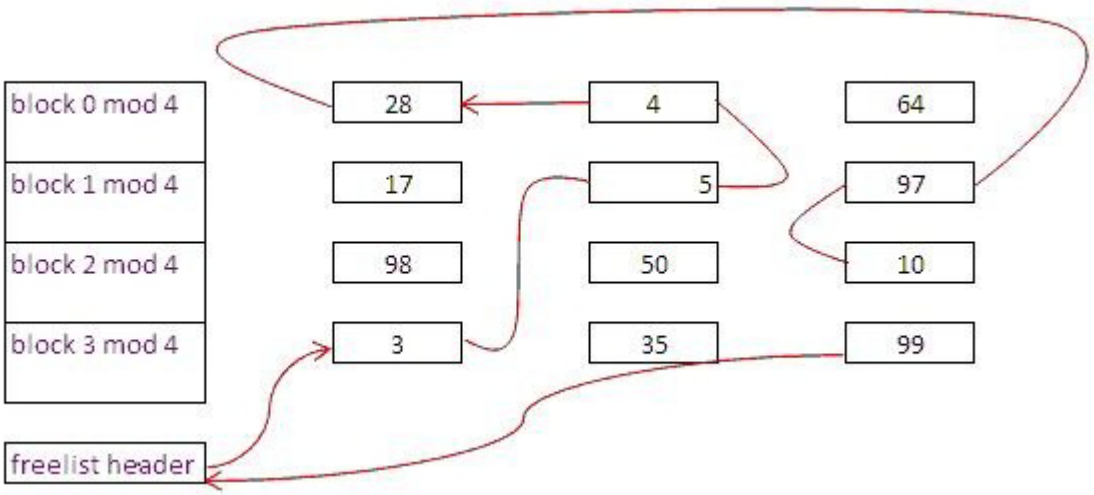


Figure 2.1.1: Inodes

Structure of a Regular file on Disk

(Reference: *The Design of the UNIX Operating System - by Maurice J. Bach)* As stated previously inodes contains the table of content of the file data on disk. Table of content is a sequence of disk block numbers. Generally the file data is not stored in contiguous memory locations. Therefore we need to keep track of all the block numbers on the disk.

Suppose a UNIX systems have the following entries as the table of contents:

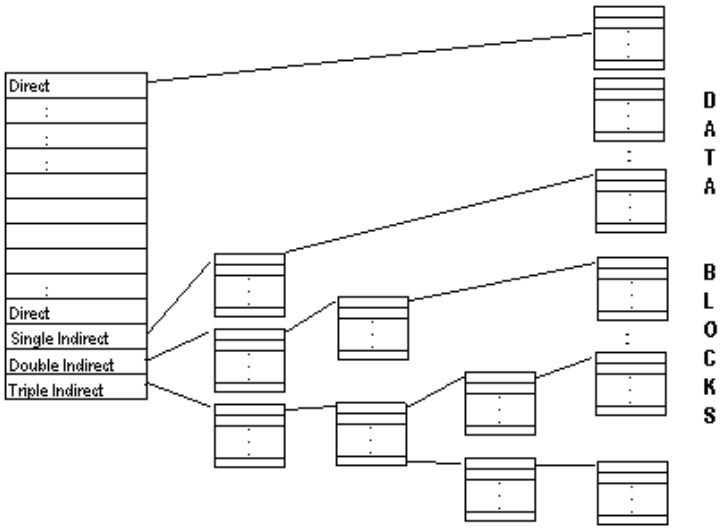


Figure 2.1.2: (Reference: *The Design of the UNIX Operating System - by Maurice J. Bach)*

"direct": the block marked direct can link to a single disk block that contains the real data. "Single indirect": the block marked single indirect contains the number of a disk block which in itself contains a list of block numbers that we can reference and they have the real data. "double indirect" and "triple indirect" : in these blocks, kernel read the indirect block, find the appropriate direct block entry, and then read the direct block to find the data. The block marked "double indirect" contains a list of indirect block numbers, and the block marked "triple indirect" contains a list of double indirect block numbers.

Self-assessment Questions

5) An inode contains information about:
   a) File access permissions and time
   b) File ownership information
   c) Number of links to the file
   d) All of these

6) The blocks marked "direct" can refer to a _____ disk block that contains the real data
   a) Single
   b) Double
   c) Triple
   d) All of these

7) An inode is a data structure on a traditional UNIX file system that stores basic information and description of the file data and its layout on disk.
   a) File data and its layout on disk
   b) Only file data
   c) Layout on disk
   d) None of these

2.1.3 .Directory Structure in a UNIX file system:

UNIX uses an upside-down tree like hierarchical file system. In this structure root (/) is at the base of the file system and all other directories spreading from there.

The directories hold the same types of information so that the files can be located easily.. Following are the directories that exist on the major versions of UNIX –

Directory	Description
/	This is the root directory. It contains only the directories needed at the top level of the file structure.
/bin	This directory contains executable files.
/dev	This directory represents the device drivers.
/etc	Supervisor directory commands, configuration files, disk configuration files, valid user lists, groups, ethernet, hosts, where to send critical messages.

/lib	Contains shared library files and sometimes other kernel-related files.
/boot	Contains files for booting the system.
/home	Contains the home directory for users and other accounts.
/mnt	Used to mount other temporary file systems, such as cdrom and floppy for the CD-ROM drive and floppy diskette drive, respectively
/proc	Contains all processes marked as a file by process number or other information that is dynamic to the system.
/tmp	Holds temporary files used between system boots
/usr	Used for miscellaneous purposes, or can be used by many users. Includes administrative commands, shared files, library files, and others
/var	Typically contains variable-length files such as log and print files and any other type of file that may contain a variable amount of data
/sbin	Contains binary (executable) files, usually for system administration. For example fdisk and ifconfig utlities.
/kernel	Contains kernel files

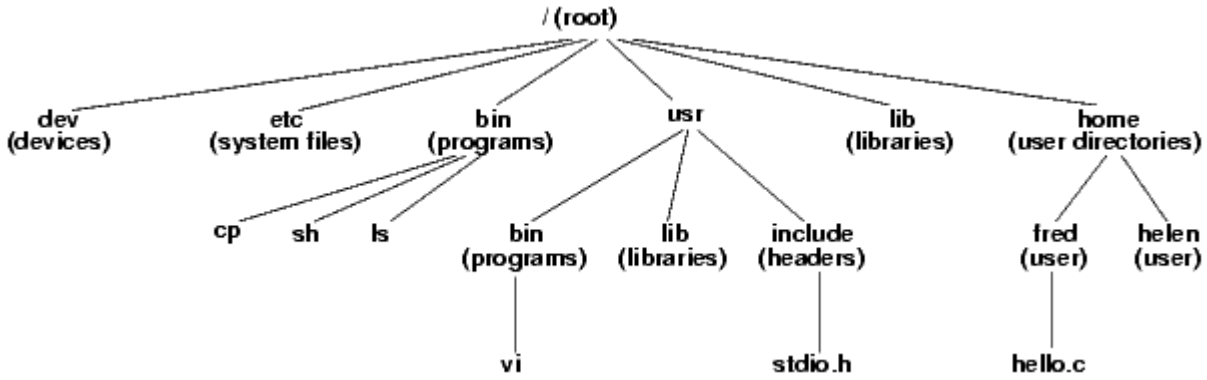


Figure 2.1.3: shows the directory structure in UNIX file system:

A UNIX file system has the following properties −

- Root directory is at the top of directory structure. Root directory (/) contains other files and directories.
- In the UNIX file system every file or directory is uniquely identified by an inode.
- A user can see inode number of a file by using –i option in the ls command.
    - $ ls –i
    - Inode numbers 1 and 0 are not used. Root directory has inode number 2 and the lost/found directory has an inode number of 3.
- There are no dependencies between one file system and any other.

Self-assessment Questions

8) The UNIX system organizes its files using an upside-down hierarchical tree structure hierarchical tree structure.

a) True b) False

9) UNIX file system have multiple root directories

a) True b) False

10) /home directory is for

a) Only for owner b) Users and other accounts

c) For group d) For users and group

2.1.4 Allocation of Disk Blocks:

The UNIX file system structure is a randomly addressable collection of characters. The maximum file system size is same as the size of disk-storage device. The file system is contained on disk devices that can be found in '/dev' directory. The block size is normally of 512 bytes to 32KB.

Following figure illustrates a typical structure of file system:

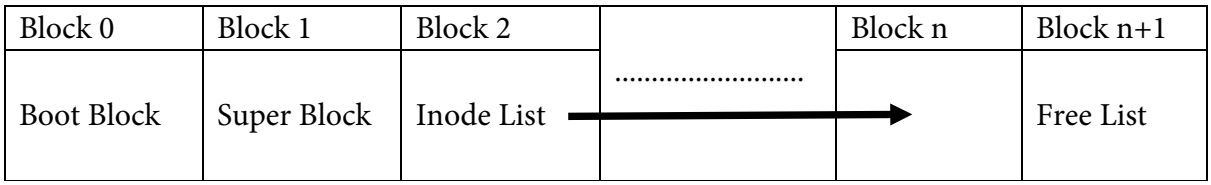


Figure 2.1.4: overall structure of a typical file system

Disk blocks:

A disk is broken down into four zones:

The boot block

The super block. This contains the size of disk being used, and its boundaries i.e. the allocated disk space already used.

The i-list. The i-list simply conatins a list of i-nodes.

The free- list (a list of unused blocks).

The initial block of a disk is boot block and it contains bootstrapping program.

The Superblock: The super block has the fields like : the file system size, number of free blocks in the file system, list of available free blocks on the file system, the index of the next free block in the free block list, the inode list size, number of free inodes in the file system, list of free inodes in the file system, index of the next free inode in the free inode list, lock fields for the free block and free inode lists, A flag indicating that the super block has been modified.

The kernel periodically writes the super block to disk if it had been modified so that it is consistent with the data in the file system.

The super block have a list of free disk block numbers. The list points to a data block. The data block points to some other free blocks and a next pointer. Index points to the next free disk block in the free disk block list as shown in the following figure 2.1

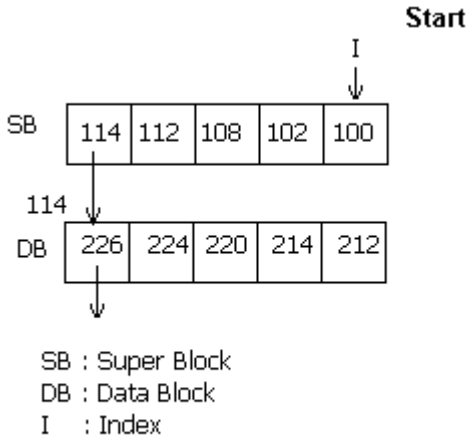


Figure 2.1.5: mapping of index to data block

2.1.5 Assignment of New Inodes

As discussed in above section super block also contains an array to represent free inodes. The kernel searches the free inode list to assign a free inode to a file as shown in figure 2.1.3.

If one free inode is found: it is returned to kernel.

If the list that holds free inode is empty then the inode list for free inodes is searched. Each inode contains a type field. If the type field value is 0, then that inode is free. Then it fills the free inode list of super block as much as possible with number of free inodes from inode list. Then it returns one of these. It then remembers the highest inode number. Next time it scans the inode list for free inodes and it starts from this remembered one. Hence it is not required to scan those are already scanned. This results in improved efficiency. Index indexes to the next free inode in the free disk block list.

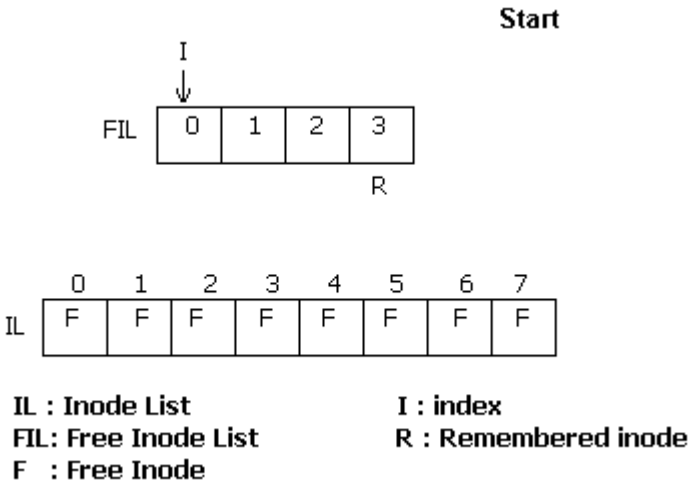


Figure 2.1.6

The I-node list is a list of inodes, which holds the information for Owner, Type, Last modified time, Last accessed time, Last inode modified time, Access Permissions, No of links to the file, Size of the file, Data blocks owned etc.

Owner indicates who is the owner of the file(s) corresponding to this inode.

The Type indicates that whether the inode represents a file, a directory, a FIFO, a character device or a block device. If the type value is 0 then the inode is free.

The times represent the modification time for the file i.e. when it was last accessed, or when the inode has been modified last. Whenever the file contents are changed, the "inode modified time" also changes. It also changes when the parameters of inode as permission, link creation etc. are changed.

Each file has permissions to read, write and execute. These will be for owner, group and others. There are nine access permissions. The format for these permissions is rwx rwx rwx.

Self-assessment Questions

11) Maximum size of the file system is size of disk-storage device

a) TRUE b) FALSE

12) The i-list simply conatins a list of

a) Unused blocks b) i-nodes.

c) Used blocks d) All of these

13) The UNIX file system structure is a _____addressable collection of characters.

a) Randomly b) Sequentially

c) Continuous d) All of these

2.1.6 Conversion of a pathname to inode:

In a UNIX pathname, slashes(/) are used to separate names of pathname components. All the components from left to rightmost slash are directories.

Suppose

$ /home/ankit/ foobar

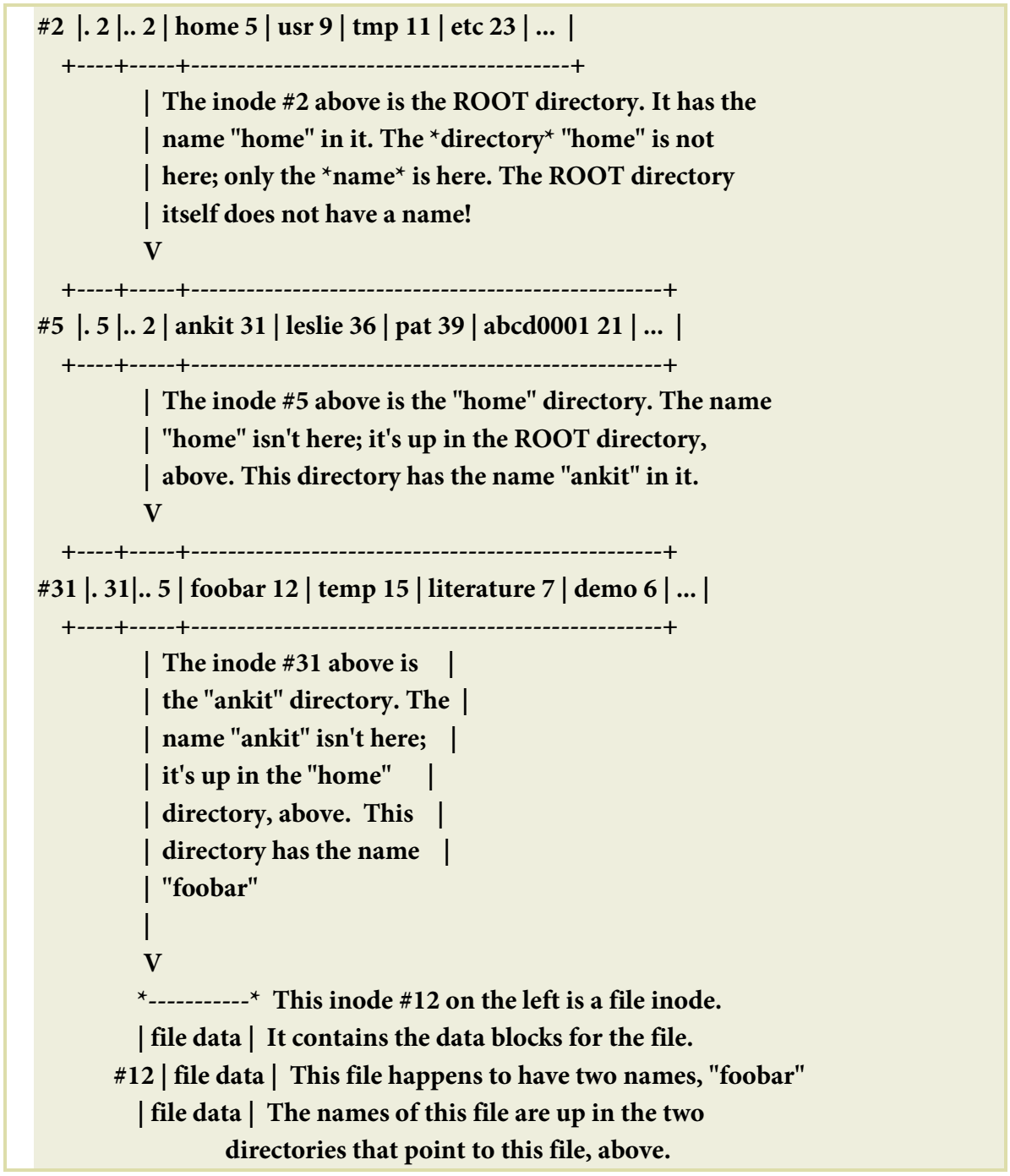
In this pathname first slash(/) represents root directory. The ROOT directory does not have a name.

- Inside the ROOT directory, the home directory is present.
- Inside the home directory, there is ankit directory
- Inside the ankit directory, there is there is a file named foobar

The last pathname component can be a file or directory. In above example, foobar is the filename.

Let us trace a filename to an inode. In below diagram each box represents an inode; inode numbers for the box are given in the left of box. Inside the directory inodes, names and inode numbers are paired.

For example, we will trace inode for /home/ankit/ foobar



The pathname /home/alex/foobar starts at the nameless ROOT directory, inode #2. It travels through two more directory inodes and stops at file inode #12. Using all four inode numbers, /home/alex/foobar could be written as #2->#5->#31->#12.

Let's examine each of the above inodes.

The box below represents the layout of names and inode numbers inside the actual disk space given to the nameless ROOT directory, inode #2:

```
   +----+-----+---------------------------------------+
#2  |. 2 |.. 2 | home 5 | usr 9 | tmp 11 | etc 23 | ... |
   +----+-----+---------------------------------------+
```

The above ROOT directory has the name home in it, paired with inode #5. The actual disk *space* of the directory home is not here; only the *name* home is here, alongside of its own inode number #5. To read the actual contents of the home directory, you have to find the disk space managed by inode #5somewhere else on disk and look there.

The above ROOT directory pairing of home with inode #5 is what gives the home directory its name. The name home is separate from the disk space for home. The ROOT directory itself does not have a name; because, it has no parent directory to give it a name!

The ROOT directory is the only directory that is its own parent. If you look at the ROOT directory above, you will see that both the name and the name.. in this ROOT directory are paired with inode #2, the inode number of the ROOT directory. Following either name . or .. will lead to inode #2 and right back to this same ROOT inode.

Let us move to the storage space for the home directory at inode #5.

The box below represents the layout of names and inode numbers inside the actual disk space given to the home directory, inode #5:

```
   +----+-----+-------------------------------------------------+
#5  |. 5 |.. 2 | alex 31 | leslie 36 | pat 39 | abcd0001 21 | ... |
   +----+-----+-------------------------------------------------+
```

The name home for this inode isn't in this inode; the name home is up in the ROOT directory. This homedirectory has the name ankit in it, paired with inode #31. The *directory* ankit is not here; only the *name* ankit is here. To read the ankit directory, you have to find inode #31 on disk and look there. (In fact, until you look up inode #31 and find out that it is a directory, you have no way of even knowing that the name ankit is a name of a directory!)

Let us move to the storage space for the ankit directory at inode #31.

The box below represents the layout of names and inode numbers inside the actual disk space given to the ankit directory, inode #31:

```
  +----+-----+-------------------------------------------------+
#31 |. 31|.. 5 | foobar 12 | temp 15 | literature 7 | demo 6 | ... |
  +----+-----+-------------------------------------------------+
```

The name ankit for this inode isn't in this inode; the name ankit is up in the home directory. This ankit directory has the name foobar in it, paired with inode #12. The *file* foobar is not here; only the *name* foobar is here. To read the data from file foobar, you have to find inode #12 on disk and look there. (In fact, until you look up inode #12 and find out that it is a plain file, you have no way of even knowing that the name foobar is a name of a plain file!)

Let us move to the storage space for the foobar file at inode #12.

The box below represents the actual disk space given to the foobar file, inode #12:

```
  *-----------*
#12 | file data |
  *-----------*
```

The name foobar for this inode isn't in this inode; the name foobar is up in the ankit directory. This foobarinode is a file inode, not a directory inode, and the attributes of this inode will indicate that.

The inode for a file contains pointers to disk blocks that contain file data, not directory data. There are no special directory names . and .. in files. There are no names here at all; the disk block pointers in this inode point to just file data (whatever is in the file).

This completes the inode trace for /home/alex/foobar: #2->#5->#31->#12

Summary

- Everything is treated as file in UNIX. Files have been assumed to be of four types. An ordinary/regular file contains what you write into it. A directory maintains the filename and its associated inode number. A device file contains no data but kernel uses the attributes of device file to operate the device.
- A file system is a hierarchical structure, and top most directories is called root. Files and directories have a parent-child relationship
- Internal format of a file is dictated by the utility that creates it. So, UNIX files are structure less.
- An inode is a data structure on a traditional UNIX file system that stores basic information and description of the file data and its layout on disk.
- An inodes contains the table of content of the file data on disk. As each disk block can be referenced by a number, the table of content is nothing but a sequence of disk block numbers.
- The super block. This contains the size of disk being used, and its boundaries i.e. the allocated disk space already used.
- In a UNIX pathname, slashes (/) separate names of pathname components. All the components from left to rightmost slash are directories.

Terminal Questions

1. Explain the different types of files in UNIX. Also describe the structure of a regular file.

2. Explain the directory structure of the UNIX file system.

3. Define superblock and the process of freeing a superblock. Also explain the various disk allocation methods.

4. Describe inode along with its importance. Also discuss in detail how to assign inode to a new file.

5. Illustrate the steps to convert pathname to an inode.

Answer Keys

Self-assessment Questions	
Question No.	Answer
1	b
2	b
3	b
4	a
5	d
6	a
7	a
8	a
9	b
10	b
11	a
12	b
13	a

Activity

Activity Type: Online/Offline **Duration: 30 Minutes**

Description:

Prepare a presentation (min 15 sides) on inode.

Bibliography

e-References

- This website was referred on 3rd May 2016 while developing content for Unix file system http://edusagar.com/articles/view/23/Inode-file-structure-on-Unix
- This website was referred on 3rd May 2016 while developing content for Unix file system https://www.google.co.in/url?sa=t&rct=j&q=&esrc=s&source=web&cd=4&cad=rja&uact=8&ved=0ahUKEwjf976JgobKAhWLPhQKHZWoBI0QFggwMAM&url=http%3A%2F%2Fcau.ac.kr%2F~bongbong%2Flinux09%2Flinux08.ppt&usg=AFQjCNHo5jb7p5uYXr-wGgsN8iTmHk6Ibg&sig2=655gCYP9xV8c63oOHWoekg
- This website was referred on 3rd May 2016 while developing content for Unix file system http://www.linfo.org/inode.html
- This website was referred on 3rd May 2016 while developing content for Unix file system http://www.cyberciti.biz/tips/understanding-unixlinux-filesystem-superblock.html

External Resources

- Maurice J. Bach, The Design of Unix Operating System, (2010) Pearson Education
- S. Prata, Advance UNIX, a Programmer's Guide, (2011), BPB Publications, and New Delhi,
- B.W. Kernighan & R. Pike, The UNIX Programming Environment, (2009) Prentice Hall of India.
- Jack Dent Tony Gaddis, Guide to UNIX Using LINUX, (2010) Vikas/ Thomson Pub. House Pvt. Ltd.

Video Links

Topic	Link
The Linux File System	https://www.youtube.com/watch?v=2qQTXp4rBEE
Directory structure of the UNIX file system	https://www.youtube.com/watch?v=PEmi550E7zw

Notes: