MIT | Academy of Engineering

(An Autonomous Institute affiliated to Savitribai Phule Pune University)

PROJECT REPORT

FILE-SYSTEM FORMATTING

By

Nishant Kumar
Chetan Pawar

Guided by
Mr. Mayur Patil

DEPARTMENT OF COMPUTER ENGINEERING
MIT ACADEMY OF ENGINEERING
ALANDI (D), PUNE
2017 - 2018
.

MIT ACADEMY OF ENGINEERING
ALANDI (D), PUNE

DEPARTMENT OF COMPUTER ENGINEERING

CERTIFICATE

This is to certify that the seminar entitled "FILE-SYSTEM FORMATTING" has been carried out by NISHANT KUMAR and CHETAN PAWAR under my guidance in partial fulfillment of Third Year Computer Engineering of Savitribai Phule Pune University, Pune during the academic year 2017-2018.

Mr . Mayur Patil	Dr. Shitalkumar. A. Jain
Seminar Guide	**Head of the Department**

ACKNOWLEDGEMENT

I take this opportunity to express my sincere appreciation for the cooperation given by Dr. ShitalKumar. A. Jain, HOD (Department of Computer Engineering) and need a special mention for all the motivation and support.

I am deeply indebted to my guide Mr . Mayur Patil for completion of this project for which she has guided and helped me going out of the way.

For all efforts behind the Project report, I would also like to express my sincere appreciation to staff of department of Computer Engineering, MIT Academy of Engineering, Pune for their extended help and suggestions at every stage.

Nishant Kumar
Chetan Pawar

INDEX

ABSTRACT

In computing, a ***file system*** *or* ***filesystem*** *is used to control how data is stored and retrieved. Without a file system, information placed in a storage medium would be one large body of data with no way to tell where one piece of information stops and the next begins. By separating the data into pieces and giving each piece a name, the information is easily isolated and identified. Taking its name from the way paper-based information systems are named, each group of data is called a "file". The structure and logic rules used to manage the groups of information and their names is called a "file system".*

There are many different kinds of file systems. Each one has different structure and logic, properties of speed, flexibility, security, size and more. Some file systems have been designed to be used for specific applications. For example, the ISO 9660 file system is designed specifically for optical discs.

The file system provides the mechanism for online storage and access to file contents, including data and programs. This paper covers the high-level details of file systems, as well as related topics such as the disk cache, the file system interface to the kernel, and the user-level APIs that use the features of the file system. It will give you a thorough understanding of how a file system works in general. The main component of the operating system is the file system. It is used to create, manipulate, store, and retrieve data. At the highest level, a file system is a way to manage information on a secondary storage medium. There are so many layers under and above the file system. All the layers are to be fully described here. This paper will give the explanatory knowledge of the file system designers and the researchers in the area

1. INTRODUCTION

1.1 Problem Statement

Implement file system format program which will copy host OS file system and formats secondary storage such as Flash drives, USB drives with same file system.

1.2 Project Background

File systems can be used on numerous different types of storage devices that use different kinds of media. The most common storage device in use today is a hard disk drive. Other kinds of media that are used include flash memory, magnetic tapes, and optical discs. In some cases, such as with tmpfs, the computer's main memory (random-access memory, RAM) is used to create a temporary file system for short-term use.

1.3 Objective

- The purpose of this document is to Design and Implement file system format program which will copy host OS file system and formats secondary storage such as Flash drives, USB drives with file system such as-

  1)ext4
  2)NTFS
  3)Fat32
  4)btrfs
  5)reiserfs

- Provides a concrete description of the project's effect at the outcome level;
- Was developed in a participatory process;
- Is accepted by the target group and other stakeholders;
- Is clear and concise.

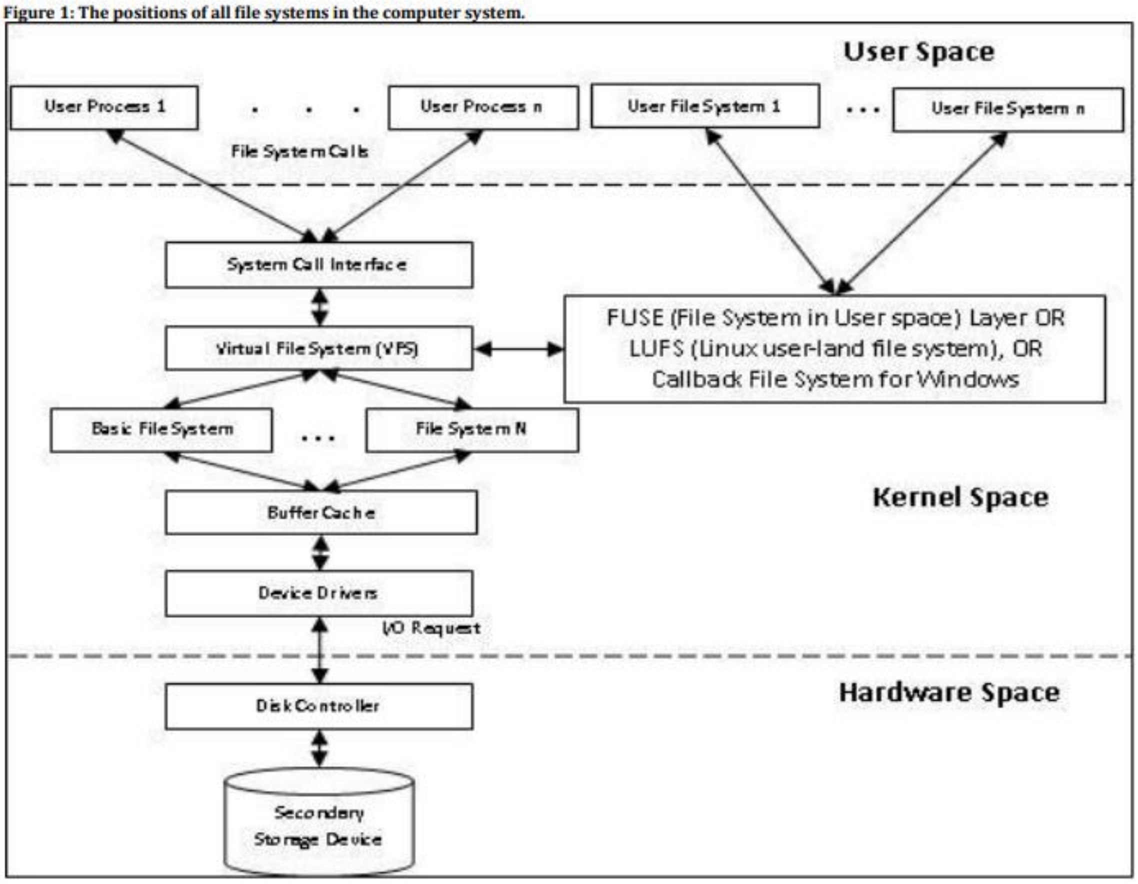
1.4 Motivation

Project motivation is a key aspect to a successful project, yet few studies exist that shed light on this important subject. This article reviews the current literature and theoretical aspects of motivation and provides an analysis of the data.

2. Design Details:

2.1 UML-Diagram:

Figure 1: The positions of all file systems in the computer system.



3.Implementation Details

3.1 Hardware and Software requirements

- Apache2
- Python3
- Django Framework
- Bash Commands
- Html
- Css
- Gksudo
- Linux OS

3.2 Environmental Setup Steps

- sudo apt-get -y install python3-pip
- pip3 install -y Django
- sudo apt-get -y install gksudo
- sudo apt-get -y install libguestfs-reiserfs
- sudo apt-get update
- sudo apt-get -y install btrfs-tools

3.3 Code

```
import sys
import os
def Unmounting(loc):
unmount_cmd = 'umount '+loc
os.system('gksudo '+unmount_cmd)

def MakeExt4System(loc):
ext_format_cmd = 'mkfs.ext4 '+loc
os.system('gksudo '+ext_format_cmd)

def MakeNtfsSystem(loc):
ntfs_format_cmd = 'mkfs.ntfs '+loc
```

```
os.system('gksudo '+ntfs_format_cmd)
def MakeFat32System(loc):
fat32_format_cmd = 'mkfs.vfat '+loc
os.system('gksudo '+fat32_format_cmd)
def MakeBtrfsSystem(loc):
btrfs_format_cmd = 'mkfs.btrfs -f '+loc
os.system('xterm -e sudo '+btrfs_format_cmd)
def MakeReiserfsSystem(loc):
reiserfs_format_cmd = 'mkfs.reiserfs '+loc
os.system('xterm -e sudo '+reiserfs_format_cmd)
def main(type1,loc):
if type1 == "ext4":
Unmounting(loc)
MakeExt4System(loc)
elif type1 == "ntfs":
Unmounting(loc)
MakeNtfsSystem(loc)
elif type1 == "fat32":
Unmounting(loc)
MakeFat32System(loc)
elif type1 == "reiserfs":
Unmounting(loc)
MakeReiserfsSystem(loc)
elif type1 == "btrfs":
Unmounting(loc)
MakeBtrfsSystem(loc)
if __name__ == '__main__':
main(sys.argv[1],sys.argv[2])
```

3.4 Compilation Steps

1)cd filesystem.

2)python3 manage.py runserver 8000

3)goto ipaddress localhost:8000

4.Testing

4.1 Test Cases

Basically the steps are

1. mke2fs
2. mount the filesystem
3. Create a file called "format.complete" in the filesystem
4. unmount the filesystem

So we need to put some tests before this. The logic would be:

1. Attempt to mount the filesystem at `$tmpmount` forcing `ext2`
2. If `mount` returned error code ==> Goto NOT FORMATTED
3. If `$tmpmount/lost+found` does not exist then an odd filesystem mounted; should not happen but... `umount`. Goto NOT FORMATTED
4. If `$tmpmount/format.complete` does not exist then format was interrupted; `umount`. Goto NOT FORMATTED
5. `umount` ==> FORMATTED, skip to next disk.

The "NOT FORMATTED" would be the original 4 steps.

We can add those structures together. The result would be that disks would only be formatted if they don't have a `format.complete` file on them.
Once all disks have been formatted you can optionally then remount each one and delete the `format.complete` file.
Essentially we maintain a small amount of state on each disk and use that to determine if formatting was successful.

5.Applications

5. 1 Ext4

ext4 was born as a series of backward-compatible extensions to ext3, many of them originally developed by Cluster File Systems for the Lustre file system between 2003 and 2006, meant to extend storage limits and add other performance improvements.[4] However, other Linux kernel developers opposed accepting extensions to ext3 for stability reasons,[5] and proposed to fork the source code of ext3, rename it as ext4, and perform all the development there, without affecting the current ext3 users. This proposal was accepted, and on 28 June 2006, Theodore Ts'o, the ext3 maintainer, announced the new plan of development for ext4

5.2 Fat32

In order to overcome the volume size limit of FAT16, while at the same time allowing DOS real mode code to handle the format, Microsoft designed a new version of the file system, **FAT32**, which supported an increased number of possible clusters, but could reuse most of the existing code, so that the conventional memory footprint was increased by less than 5 KiB under DOS.[39] Cluster values are represented by 32-bit numbers, of which 28 bits are used to hold the cluster number. The boot sector uses a 32-bit field for the sector count, limiting the FAT32 volume size to 2 TiB for a sector size of 512 bytes and 16 TiB for a sector size of 4,096 bytes.[40][41]FAT32 was introduced with MS-DOS 7.1 / Windows 95 OSR2 in 1996, although reformatting was needed to use it, and DriveSpace 3(the version that came with Windows 95 OSR2 and Windows 98) never supported it.

5.3 NTFS

NTFS ("**New Technology File System**")[1] is a proprietary file system developed by Microsoft.[1] Starting with Windows NT 3.1, it is the default file system of the Windows NT family.[7]

NTFS has several technical improvements over the file systems that it superseded – File Allocation Table (FAT) and High Performance File System (HPFS) – such as improved support for metadata and advanced data structures to improve performance, reliability, and disk space use. Additional extensions are a more elaborate security system based on access control lists (ACLs) and file system journaling.

5.4 BtrFs

Btrfs (**B-tree file system**, pronounced as "butter fuss",[10] "better F S",[7] "b-tree F S",[11] or simply by spelling it out) is a file systembased on the copy-on-write (COW) principle, initially designed at Oracle Corporation for use in Linux. The development of Btrfs began in 2007, and since August 2014 the file system's on-disk format has been marked as stable.

Btrfs is intended to address the lack of pooling, snapshots, checksums, and integral multi-device spanning in Linux file systems.[7] Chris Mason, the principal Btrfs author, has stated that its goal was "to let Linux scale for the storage that will be available. Scaling is not just about addressing the storage but also means being able to administer and to manage it with a clean interface that lets people see what's being used and makes it more reliable".

5.5 ReiserFS

ReiserFS is a general-purpose, journaled computer file system formerly designed and implemented by a team at Namesys led by Hans Reiser. ReiserFS is currently supported on Linux (without quota support) licensed as GPLv2. Introduced in version 2.4.1 of the Linux kernel, it was the first journaling file system to be included in the standard kernel. ReiserFS is the default file system on the Elive, Xandros, Linspire, and YOPER Linux distributions. ReiserFS was the default file system in Novell's SUSE Linux Enterprise until Novell decided to move to ext3 on October 12, 2006 for future releases.

6.Conclusion

This paper gives a light on the layer of file system in the computer system. All the upper layers and the lower layer are shown in it that gives a sound knowledge about the file system interaction between them. This review paper also tells about the view of file systems that may be the user file systems which has the interface kernel module like FUSE, LUFS, and Callback File System. These interface modules are intermediate between the VFS and the user space file systems.

7.REFERENCES

Web sources:

- Wikipedia
- www.stackoverflow.com
- Ubuntu Forum
- the Linux Man.[Youtube Channel]

.

Books:

Pate, Steve D. (2003). UNIX Filesystems: Evolution, Design, and Implementation. Wiley. ISBN 0-471-16483-6