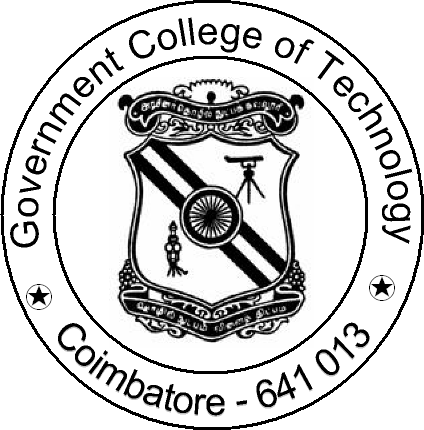
**IMPLEMENTATION OF FUSE**

**Dropbox File System in UNIX**



PROJECT SUBMITTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS THE AWARD OF THE DEGREE OF **BACHELOR OF ENGINEERING IN COMPUTER SCIENCE AND ENGINEERING** OF THE ANNA UNIVERSITY

Submitted by

**S.ASHWIN(1317105)**

**PROJECT WORK**

Under the Guidance of

**Mrs. Dr.RATHI APCSE.,**

**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING**

**GOVERNMENT COLLEGE OF TECHNOLOGY**

(An Autonomous Institution affiliated to Anna University of Technology Coimbatore)

**COIMBATORE - 641 013**

**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING**

**GOVERNMENT COLLEGE OF TECHNOLOGY**

(An Autonomous Institution affiliated to Anna University)

**COIMBATORE - 641 013**

PROJECT WORK

## BONAFIDE RECORD

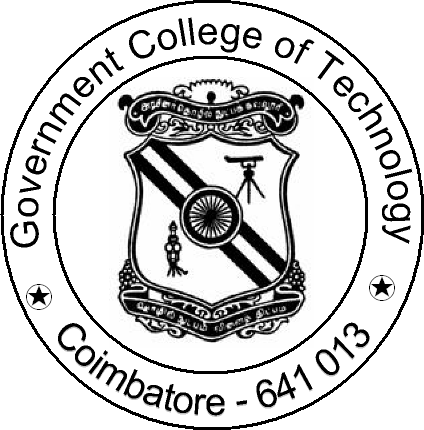
**NOVEMBER 2015**

This is to certify that this project work entitle

**IMPLEMENTATION OF FUSE**

**DROPBOX FILE SYSTEM IN UNIX**

is the bonafide record of project work done by

 **S.ASHWIN(1317105)**

**S.AARTHI(1317101)**

**K.NISHANTHINI(1317131)**

of B.E.(Computer Science & Engineering) during the year 2015-2016

Project GuideHead of the Department

**Mrs. Dr.RATHI APCSE., Prof. S. Kumaresan, M.E.,**

Submitted for the Project Viva-Voce examination held on

**Internal Examiner**   **External Examiner**

## GCTACKNOWLEDGEMENT

Great achievements are not possible without standing on the shoulders of giants. Without the active involvement of the following experts this project would not have been a reality.

We express our sincere gratitude to **Dr.V.Lakshmi Prabha, M.E., Ph.D.,** Principal, Government College of Technology, Coimbatore for providing us all facilities that we needed for the completion of this project.

We whole-heartedly express our thankfulness and gratitude to **Prof. S. Kumaresan M.E.,** Professor andHead of the Department ofComputer Science and Engineering, Government College of Technology, for helping us to successfully carry out this project.

Our thankfulness and gratitude to our respectable project guide **Mrs. Dr.RATHI APCSE.,** Assistant Professor,Department of Information Technology, who has been of immense help through the various phases of the project. With her potent ideas and excellent guidance we were able to comprehend the essential aspects involved.

We extend our sincere thanks to **Dr.M.L.Valarmathi ,** Associate Professor **Dr.R.Shanmugalakshmi**, Associate Professor**, Dr.T.Purusothaman,** Associate Professor**, Dr.J.C.Miraclin Joyce Pamila,** Associate Professor, **Dr.K.Kumar** ,Associate Professor**,** and all other faculty members of Computer Science and Information Technology for their valuable suggestions to the completion of the project successfully.

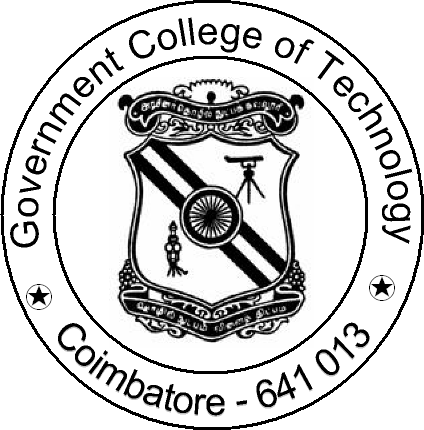
We also thank all the non-teaching staff for their support and to all our friends for their co-operation and suggestions towards the successful completion of the project.

## GCTSYNOPSIS

The cloud computing marks the beginning of the Internet age of things. The rate of the secondary storage are high in the market while purchasing the equipment. Most of the equipment in the market uses a UNIX backbone. The motivation of the project is to utilise the Cloud services and be able to transport and utilise it as a UNIX file system. By making this possible it will be possible to directly utilise shell commands on the cloud storage and making it really helpful in this Internet of Things(IOT) era.

There a quiet a range of cloud service providing companies in the market which provides a free service of upto 2-15GB at the maximum. By this even a phone with only few MB of RAM can have a big storage capacity and reduce the manufacturing cost.

The complete utilisation of Cloud service rely on the internet and it can be defined as an amalgam of cloud based services and UNIX operating system.It provides the cloud file structure in our UNIX system and can be portable when this code is run on another system. It is highly adaptive to the environment and has the capability to run on any flair of UNIX operating system.

Table of Contents

[BONAFIDE RECORD 1](#__RefHeading___Toc801_559449821)

[ACKNOWLEDGEMENT 2](#__RefHeading___Toc693_559449821)

[SYNOPSIS 3](#__RefHeading___Toc695_559449821)

[​LIST OF FIGURES: 5](#__RefHeading___Toc5453_1424099345)

[LIST OF ABBREVIATIONS 6](#__RefHeading___Toc1010_1256751980)

[​CHAPTER 1-INTRODUCTION 8](#__RefHeading___Toc5455_1424099345)

[1.1. Filesytem 8](#__RefHeading___Toc909_559449821)

[1.2. Dropbox 8](#__RefHeading___Toc911_559449821)

[1.3. Dropbox API 9](#__RefHeading___Toc913_559449821)

[1.4 FUSE 10](#__RefHeading___Toc915_559449821)

[1.5 Filesystem 10](#__RefHeading___Toc917_559449821)

[​CHAPTER 2-FUSE 11](#__RefHeading___Toc5457_1424099345)

[2.1. Introduction 11](#__RefHeading___Toc919_559449821)

[2.2 Installation 12](#__RefHeading___Toc921_559449821)

[2.3. Sample FUSE structure 13](#__RefHeading___Toc923_559449821)

[2.4. Project WorkFlow 13](#__RefHeading___Toc925_559449821)

[2.4.1.Import Headers 13](#__RefHeading___Toc927_559449821)

[2.4.2.Declare Variables – 13](#__RefHeading___Toc929_559449821)

[2.4.3. Syscall Declaration 13](#__RefHeading___Toc931_559449821)

[2.4.4. Syscall Functions 13](#__RefHeading___Toc933_559449821)

[2.4.5. Main() 14](#__RefHeading___Toc935_559449821)

[​CHAPTER 3-API 15](#__RefHeading___Toc5459_1424099345)

[3.1. Introduction 15](#__RefHeading___Toc937_559449821)

[3.2. Types of API 15](#__RefHeading___Toc939_559449821)

[3.3. Popular API 15](#__RefHeading___Toc941_559449821)

[​ Chapter 4: IMPLEMENTATION AND RESULT 17](#__RefHeading___Toc5461_1424099345)

[4.1. Basic File Set: 17](#__RefHeading___Toc943_559449821)

[4.2. Authentication: 17](#__RefHeading___Toc945_559449821)

[4.3. Libraries 17](#__RefHeading___Toc947_559449821)

[4.4. API Interface 18](#__RefHeading___Toc949_559449821)

[4.5. Cache 18](#__RefHeading___Toc951_559449821)

[4.6.Screenshots in order of execution: 19](#__RefHeading___Toc953_559449821)

[File structure for the project 21](#__RefHeading___Toc955_559449821)

[4.7. LogFiles: 22](#__RefHeading___Toc957_559449821)

[4.7.1. ls Command 22](#__RefHeading___Toc959_559449821)

[4.7.2. Command cd 22](#__RefHeading___Toc961_559449821)

[4.7.3 mkdir command 22](#__RefHeading___Toc963_559449821)

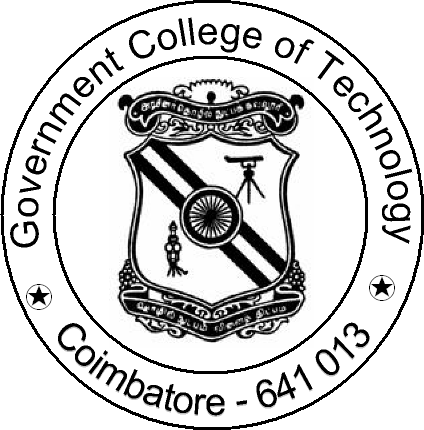
[4.7.4. mv Command 23](#__RefHeading___Toc965_559449821)

[​Chapter 5- Conclusion 26](#__RefHeading___Toc1146_2106615733)

[5.1. Technical Conclusion: 26](#__RefHeading___Toc967_559449821)

[5.2. Future Work 26](#__RefHeading___Toc969_559449821)

[​Bibliography 27](#__RefHeading___Toc5465_1424099345)

****

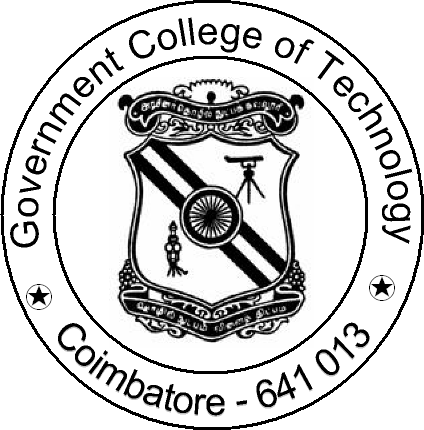
## LIST OF FIGURES:

FigureNo FigureName

1.1 Dropbox Growth

1.2 Fuse File Structure

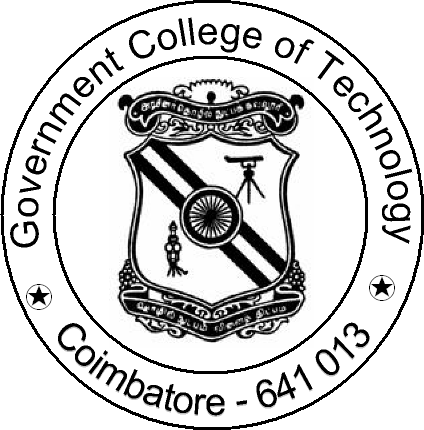
2.2 Fuse Installation



## LIST OF ABBREVIATIONS

FUSE: Filesystem in Userspcae

API: Application Program Interface



## 

**IMPLEMENTATION OF FUSE**

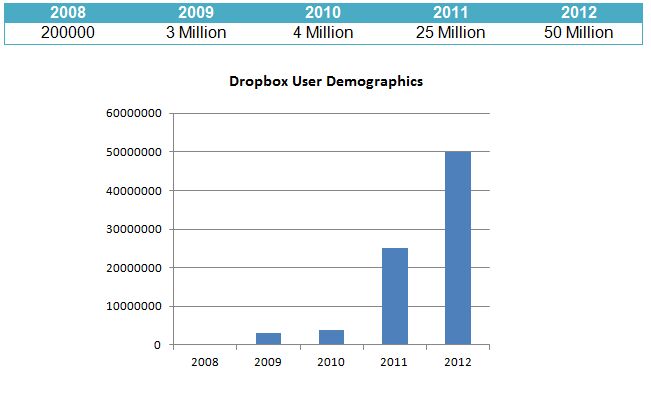
**Dropbox File System in UNIX**

### GCTCHAPTER 1-INTRODUCTION

### 1.1. Filesytem

A file system is used to control how data is stored and retrieved. Without a file system information placed in a storage area would be one large body of data with no way to tell where one piece of information stop and the next begins. There are many kinds of fie system. Each one has different structure and logic, properties of speed, flexibility, security, size and more. The directories have specific purposes and generally hold the same types of information for easily locating files.

### 1.2. Dropbox

Dropbox is the most popular online storage service which focuses on online backup, file synchronization and file sharing. To synchronize files between the online storage and a mobile device or pc, dropbox offers a special application. The main advantage of dropbox is that initially offers only 2 GB of free storage, compared to one drive, which offers 15 GB of free storage. This application is compatible with Mac, windows and Linux system.

### GCT1.3. Dropbox API

API, an abbreviation of application program interface, is a set of routines, protocols, and tools for building software applications. The API specifies how software components should interact and APIs are used when programming graphical user interface (GUI) components.

The Core API provides a flexible way to read and write to Dropbox. It includes support for advanced functionality like search, revisions, and restoring files.

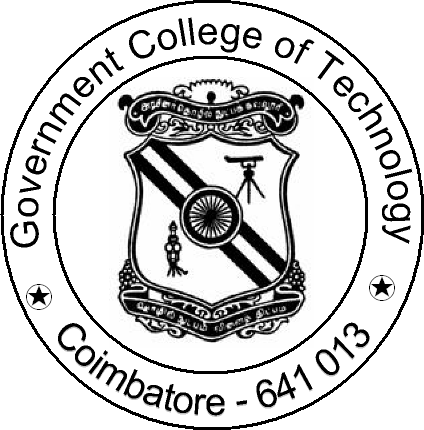
For more details on Core API for Dropbox- <https://www.dropbox.com/developers-v1/core/docs>

### GCT1.4 FUSE

File system in userspace(FUSE) is an operating system mechanism for unix-like computer operating systems that lets non privileged users create their own file systems without editing kernel code. This is achieved by running file system code in user space while the FUSE module provides only a bridge to the actual kernel interfaces. FUSE is particularly used for writing virtual file system.

### 1.5 Filesystem

A file system is used to control how data is stored and retrieved. Without a file system information placed in a storage area would be one large body of data with no way to tell where one piece of information stop and the next begins. There are many kinds of fie system. Each one has different structure and logic, properties of speed, flexibility, security, size and more.

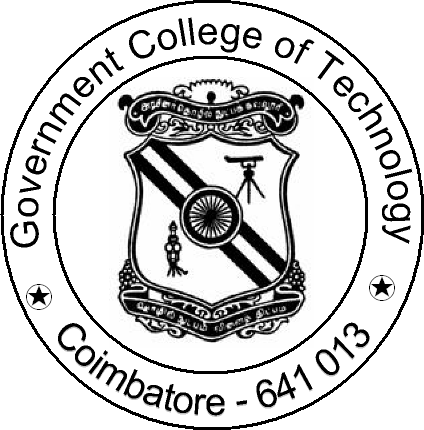


## CHAPTER 2-FUSE

### 2.1. Introduction

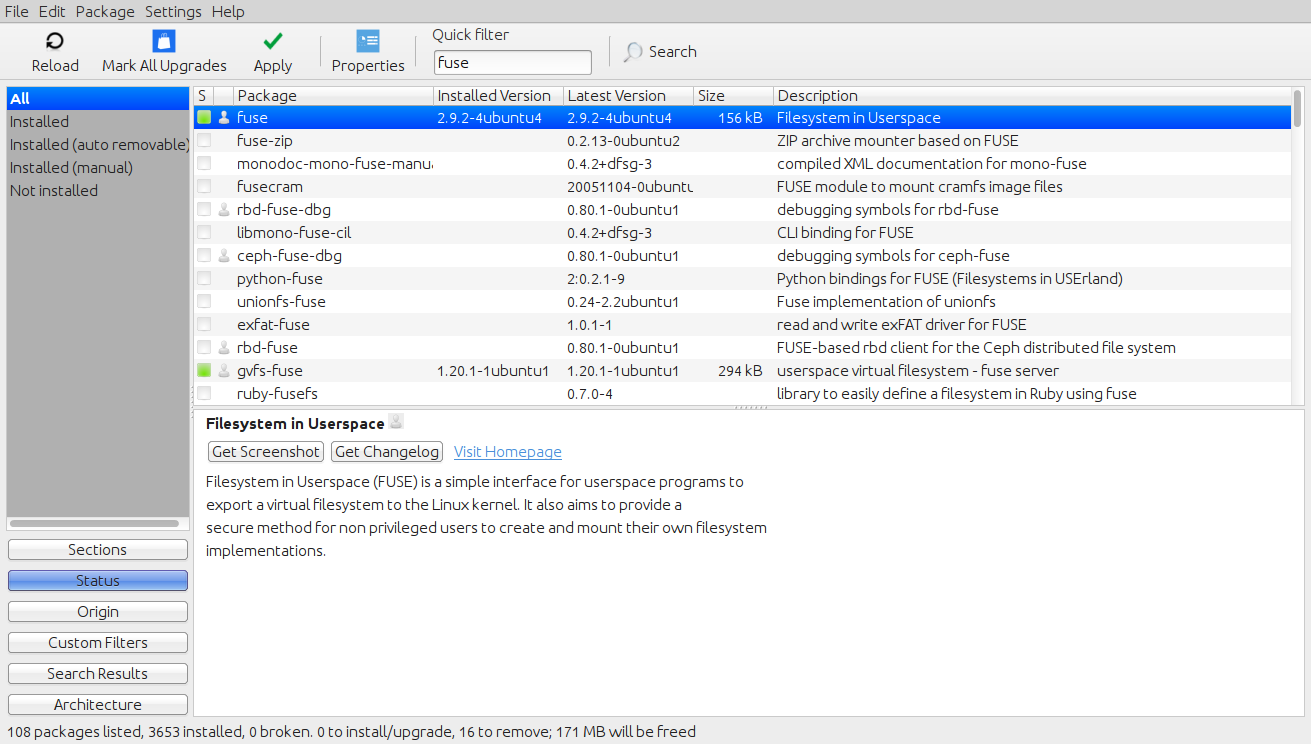
The Filesystem in Userspace (FUSE) is a special part of the Linux kernel that allows regular users to make and use their own filesystems without needing to change the kernel or have Root privileges. The filesystems used in FUSE are virtual filesystems. Not all virtual filesystems use FUSE. The code for FUSE itself is in the kernel, but the filesystem is in userspace. However, typical filesystems exist in the kernelspace.

FUSE allows users to make their own filesystems, modify filesystems, or use a special filesystem temporarily. FUSE can also be used to add extra features and abilities to a system or software. For example, GVFS (GNOME Virtual FileSystem) is a filesystem that allows applications to access remote files as if they were local. FUSE makes this possible. Also, at the time this article was written, the Linux kernel did not natively support exFAT (also called FAT64). If users wish to access a storage unit with a exFAT/FAT64 filesystem, users can mount the filesystem in FUSE. However, FUSE may need some extra packages to be installed to run some filesystems or virtual filesystems. For instance, if a user wanted to use exFAT like in the example above, then the user would need to install the "exfat-fuse" package which extends FUSE's abilities. The "exfat-fuse" package is a FUSE driver. Many drivers/extensions are available for FUSE.

FUSE not only mounts virtual filesystems, but also "real" filesystems like ZFS and exFAT. FUSE can also mount files like ISOs, CD track files, and compressed files (zip, gzip, tar, etc.). FUSE's abilities extend to network filesystems like HTTP-FS, aptfs (apt repos), and others. FUSE can also be used to transfer files to Apple devices like the ipod and iphone (iFUSE). Amazingly, FUSE can also be used to convert FLAC to MP3 via the MP3 filesystem (mp3fs).

### 2.2 Installation

FUSE is hosted at http://fuse.sourceforge.net/. FUSE is open-source freeware that anyone may obtain and use. FUSE is stable, secure, and reliable. However, it is more efficient to use a "real" filesystem if possible. FUSE is compatible and works on Solaris, FreeBSD, Darwin, GNU/Hurd, OS X, Opensolaris, and others. Some operating systems do not support FUSE. Such systems can use a fork of FUSE or use an alternative. For example, NetBSD uses PUFFS and Windows uses "fuse4win".



### GCT2.3. Sample FUSE structure

Sample FUSE structure variable code -

Code:

struct fuse\_operations {

int (\*mknod) (const char \*, mode\_t, dev\_t);

int (\*getdir) (const char \*, fuse\_dirh\_t, fuse\_dirfil\_t);

int (\*mkdir) (const char \*, mode\_t);

int (\*rmdir) (const char \*);

int (\*readlink) (const char \*, char \*, size\_t);

int (\*symlink) (const char \*, const char \*);

int (\*unlink) (const char \*);

int (\*rename) (const char \*, const char \*);

int (\*link) (const char \*, const char \*);

int (\*chmod) (const char \*, mode\_t);

int (\*chown) (const char \*, uid\_t, gid\_t);

int (\*truncate) (const char \*, off\_t);

int (\*utime) (const char \*, struct utimbuf \*);

int (\*open) (const char \*, struct fuse\_file\_info \*);

int (\*read) (const char \*, char \*, size\_t, off\_t, struct fuse\_file\_info \*);

int (\*write) (const char \*, const char \*, size\_t, off\_t,struct fuse\_file\_info \*);

int (\*statfs) (const char \*, struct statfs \*);

int (\*fsync) (const char \*, int, struct fuse\_file\_info \*);

int (\*getattr) (const char \*, struct stat \*);

int (\*setxattr) (const char \*, const char \*, const char \*, size\_t, int);

int (\*getxattr) (const char \*, const char \*, char \*, size\_t);

int (\*listxattr) (const char \*, char \*, size\_t);

int (\*removexattr) (const char \*, const char \*);

};

### 2.4. Project WorkFlow

Here is a table to help summarize the parts of a FUSE filesystem's code -

#### 2.4.1.Import Headers

All C/C++ code imports headers and other programming language import some kind of library.

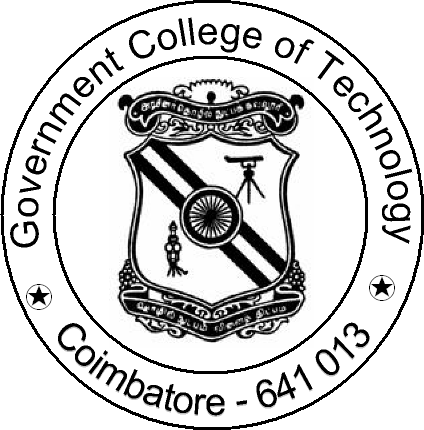
#### 2.4.2.Declare Variables –

Any variables that are used a lot in the code should be declared near the topic of the source code so the programmers can easily find and change global variables.

#### 2.4.3. Syscall Declaration

A variable structure titled “fuse\_operations” declares a syscall and needed parameters.

#### 2.4.4. Syscall Functions

Next, programmers would write code for the action or actions that should occur when a particular syscall is declared. Developers may have a function for open(), write(), read(), and many other syscalls or needed features.

#### 2.4.5. Main()

Obviously, if the FUSE filesystem is coded in C/C++, you will have a “int main()” which is where the code “starts” after the functions and variables are set.

When the filesystem “executes”, FUSE will be a mediator and communicate with the kernel on behalf of the FUSE filesystem.

## GCTCHAPTER 3-API

### 3.1. Introduction

API, an abbreviation of ***a****pplication* ***p****rogram* ***i****nterface*, is a set of routines, protocols, and tools for building software applications. The API specifies how software components should interact and APIs are used when programming graphical user interface (GUI) components.  A good API makes it easier to develop a program by providing all the building blocks. A programmer then puts the blocks together.

### 3.2. Types of API

There are many different types of APIs for operating systems, applications or for websites. Windows, for example, has many API sets that are used by system hardware and applications — when you copy and paste text from one application to another, it is the API that allows that to work.

Most operating environments, such as MS-Windows, provide an API so that programmers can write applications consistent with the operating environment. Today, APIs are also specified by websites. For example, Amazon or eBay APIs allow developers to use the existing retail infrastructure to create specialized web stores. Third-party software developers also use Web APIs to create software solutions for end-users.

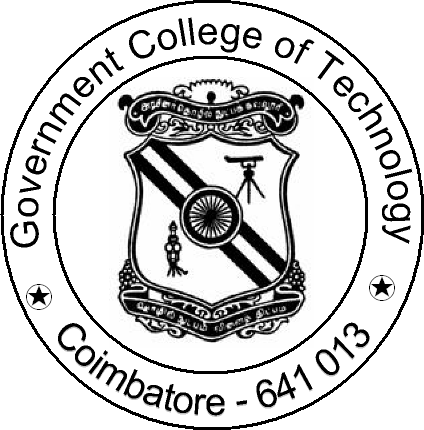
### 3.3. Popular API

Programmable Web, a site that tracks more than 13,000 APIs, lists Google Maps, Twitter, YouTube, Flickr and Amazon Product Advertising as some of the the most popular APIs. The following list contains several examples of popular APIs:

1. Google Maps API: Google Maps APIs lets developers embed Google Maps on webpages using a JavaScript or Flash interface. The Google Maps API is designed to work on mobile devices and desktop browsers.

2. YouTube APIs: YouTube API: Google's APIs lets developers integrate YouTube videos and functionality into websites or applications. YouTube APIs include the YouTube Analytics API, YouTube Data API, YouTube Live Streaming API, YouTube Player APIs and others.

3. Flickr API: The Flickr API is used by developers to access the Flick photo sharing community data. The Flickr API consists of a set of callable methods, and some API endpoints.

4. Twitter APIs: Twitter offers two APIs. The REST API allows developers to access core Twitter data and the Search API provides methods for developers to interact with Twitter Search and trends data.

5. Amazon Product Advertising API: Amazon's Product Advertising API gives developers access to Amazon's product selection and discovery functionality to advertise Amazon products to monetize a website.

## GCT Chapter 4: IMPLEMENTATION AND RESULT

### 4.1. Basic File Set:

FUSE is a system that allows the users to

* Move files(ls and cd)
* Create and Delete Folders(mkdir and rmdir)
* Create ,delete, read, append files(rm,cat,touch redirection input)

### 4.2. Authentication:

Authentication is provided by the OAuth version 12 in which the following process:

1. Client Credentials required and associated with the application (the project Dropbox-

fuse in this case) by a server at Dropbox. This step is performed

once to create the project.

1. Application Programming Interface

2. Core also supports OAuth API Version 2 Request Token is requested by the application, which connects to the Client Credentials his request.

3 URL is constructed from the Request Token received and directs User the website of Dropbox. After authenticated, it allows the application to access o his account.

4. An Access Token is finally asked. Associates Customer Credentials, they serve

to identify the transactions between the client and the server and give Dropbox

and access to the account of the user. This token can be stored and reused during

a subsequent program execution.

### 4.3. Libraries

Several libraries are used by theAPIDropbox to simplify the implementation supply. The needs they cover are:

1. client authentication with OAuth protocol version 1;

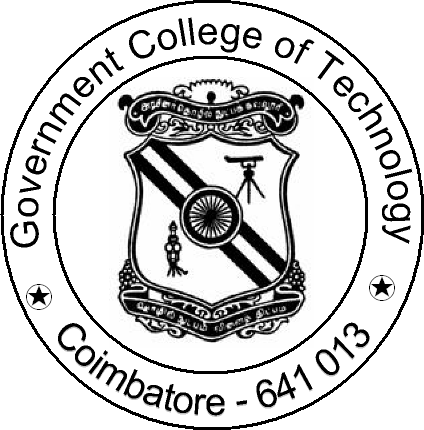
2. communication between the client and the server with HTTP;

3. serialization and JSON structure.

**Authentication is provided by the library liboauth**

3The only to provide suchservice to the C language, however, only a few functions are exploited becausemany of them are denominated deprecated and do not offer sufficient flexibility.

This weakness is filled by cURL

4)A query library URL to sup- with multiple protocols including HTTP and elsewhere used by oauth.h. His \ her choicehas established itself through its reputation and its versatility.

Regarding the reading of responses in JSON format, the choice fell on Jannson library is its free to use.

### 4.4. API Interface

The purpose of the C implementation of the API Dropbox is to offer a simple interface

flexible, as complete as possible at the level of functionality offered by the Core

API and closeAPI existing for other languages.

To achieve this, the functions are constructed as follows:

3. http://liboauth.sourceforge.net/

4. http://curl.haxx.se/

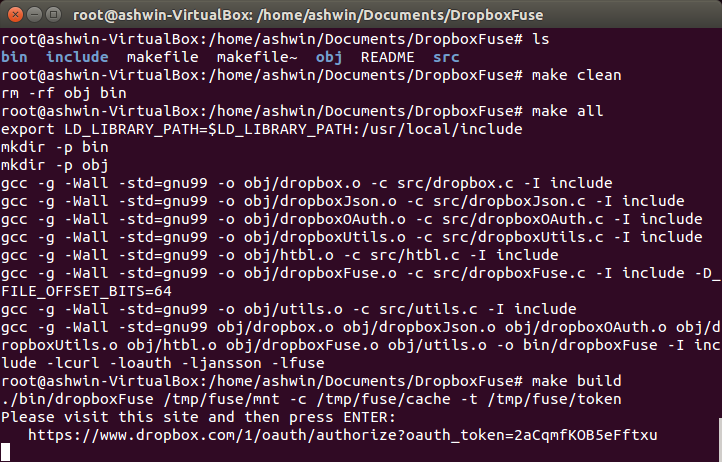
5. <http://www.digip.org/jansson/>

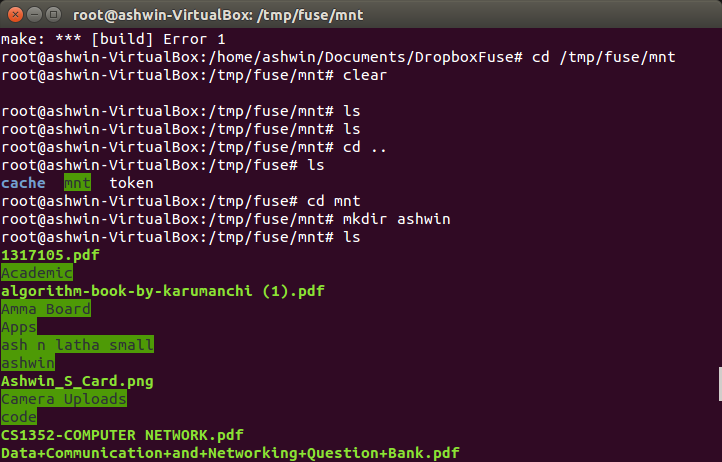
### 4.5. Cache

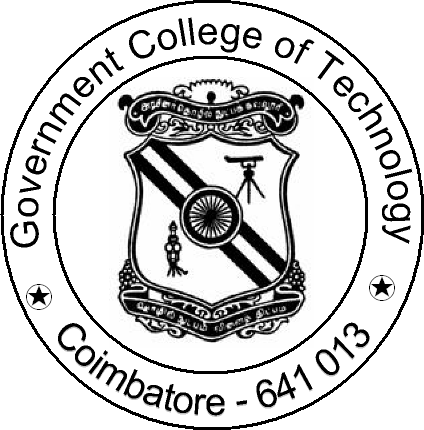
The requests sent to the Core API take between half a second and two secondsto be executed. This delay is problematic when a system command requires several queries to be performed. This situation occurs in particular for com-ls control which, after a call to readdir to get the list of file and folder name getattr for each of them. If getattr is implemented with a systematic call the metadata based on the Core API, the ls function will take several seconds to run(see minute if the folder contains several hundred files or folder). For thisBecause, file attributes are stored in a cache memory, which takes the form a hash table.Downloading files is also quite slow. These are kept on the local disk when read for the first time (or when the file version

### GCT4.6.Screenshots in order of execution:

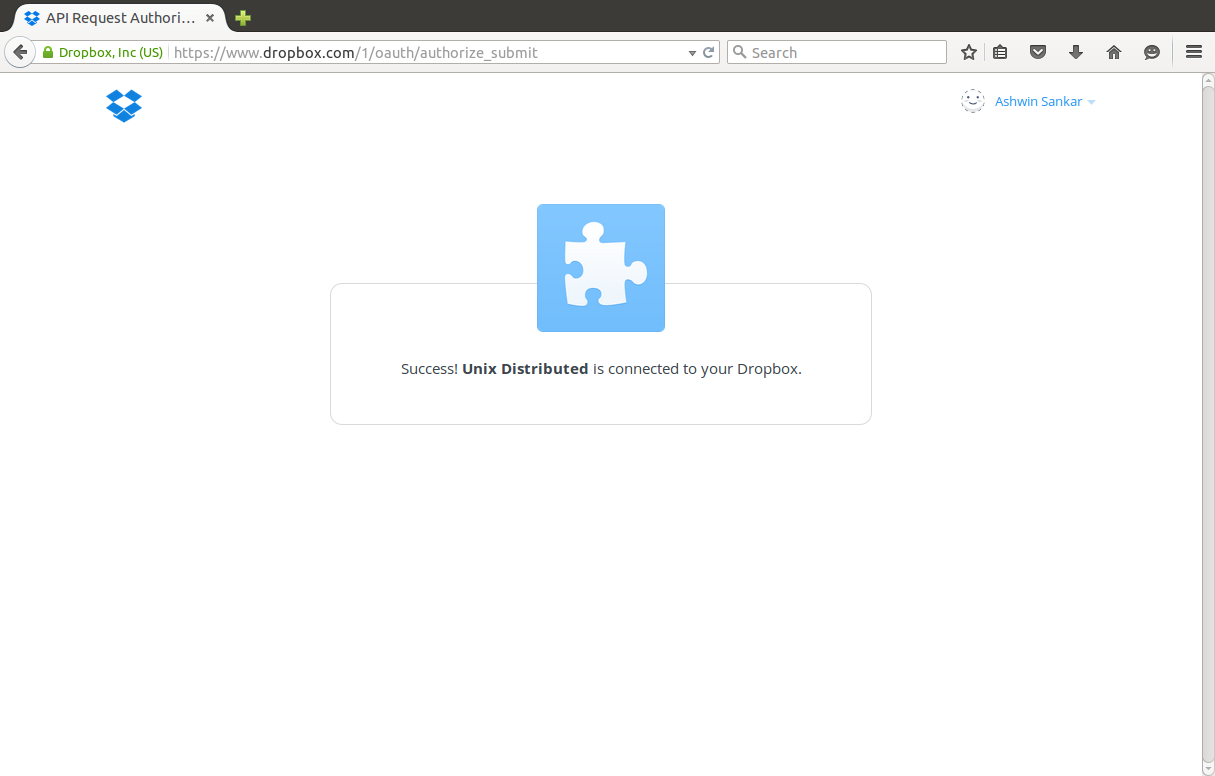
LISTING OF THE FILE,MAKING OF THE FILE AND AUTHORISATION



CREATING A FILE, AND LISTING THE FILE IN DROPBOX



AUTHORISATION(2 STEP)



CODE FOR MAKEFILE:

CC=gcc

LD\_LIBRARY\_PATH=/usr/local/include

OBJ\_PATH=obj

SRC\_PATH=src

BIN\_PATH=bin

MOUNT\_PATH=/tmp/fuse/mnt

CACHE\_PATH=/tmp/fuse/cache

INCLUDE\_PATH=include

FLAGS=-g -Wall -std=gnu99

OBJ=$(addprefix $(OBJ\_PATH)/,dropbox.o dropboxJson.o dropboxOAuth.o dropboxUtils.o dropboxUtils.o htbl.o dropboxFuse.o utils.o)

BIN=$(BIN\_PATH)/dropboxFuse

DROPBOX\_H = $(addprefix $(INCLUDE\_PATH)/, dropbox.h dropboxOAuth.h dropboxJson.h dropboxUtils.h)

DROPBOX\_JSON\_H = $(addprefix $(INCLUDE\_PATH)/, dropboxJson.h)

DROPBOX\_OAUTH\_H = $(addprefix $(INCLUDE\_PATH)/, dropboxOAuth.h dropboxUtils.h)

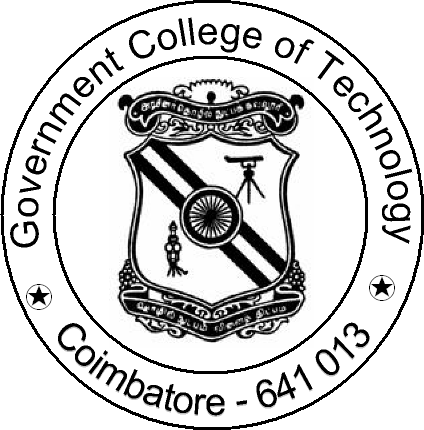
DROPBOX\_UTILS\_H = $(addprefix $(INCLUDE\_PATH)/, dropboxUtils.h)

HTBL\_H = $(addprefix $(INCLUDE\_PATH)/, htbl.h)

UTILS\_H = $(addprefix $(INCLUDE\_PATH)/, utils.h)

DROPBOX\_FUSE\_H = $(addprefix $(INCLUDE\_PATH)/, dropbox.h htbl.h utils.h)

all: EXPORT\_VAR $(BIN\_PATH) $(OBJ\_PATH) $(CACHE\_PATH) $(MOUNT\_PATH) $(BIN)



build:

./bin/dropboxFuse $(MOUNT\_PATH) -c $(CACHE\_PATH) -t /tmp/fuse/token

EXPORT\_VAR:

export LD\_LIBRARY\_PATH=$$LD\_LIBRARY\_PATH:$(LD\_LIBRARY\_PATH)

$(BIN): $(OBJ)

$(CC) $(FLAGS) $^ -o $@ -I $(INCLUDE\_PATH) -lcurl -loauth -ljansson -lfuse

$(OBJ\_PATH)/dropbox.o : $(SRC\_PATH)/dropbox.c $(DROPBOX\_H)

$(CC) $(FLAGS) -o $@ -c $< -I $(INCLUDE\_PATH)

$(OBJ\_PATH)/dropboxJson.o : $(SRC\_PATH)/dropboxJson.c $(DROPBOX\_JSON\_H)

$(CC) $(FLAGS) -o $@ -c $< -I $(INCLUDE\_PATH)

$(OBJ\_PATH)/dropboxOAuth.o : $(SRC\_PATH)/dropboxOAuth.c $(DROPBOX\_OAUTH\_H)

$(CC) $(FLAGS) -o $@ -c $< -I $(INCLUDE\_PATH)

$(OBJ\_PATH)/dropboxUtils.o : $(SRC\_PATH)/dropboxUtils.c $(DROPBOX\_UTILS\_H)

$(CC) $(FLAGS) -o $@ -c $< -I $(INCLUDE\_PATH)

$(OBJ\_PATH)/htbl.o : $(SRC\_PATH)/htbl.c $(HTBL\_H)

$(CC) $(FLAGS) -o $@ -c $< -I $(INCLUDE\_PATH)

$(OBJ\_PATH)/utils.o : $(SRC\_PATH)/utils.c $(UTILS\_H)

$(CC) $(FLAGS) -o $@ -c $< -I $(INCLUDE\_PATH)

$(OBJ\_PATH)/dropboxFuse.o : $(SRC\_PATH)/dropboxFuse.c $(DROPBOX\_FUSE\_H)

$(CC) $(FLAGS) -o $@ -c $< -I $(INCLUDE\_PATH) -D\_FILE\_OFFSET\_BITS=64

$(OBJ\_PATH) $(BIN\_PATH) $(CACHE\_PATH) $(MOUNT\_PATH) :

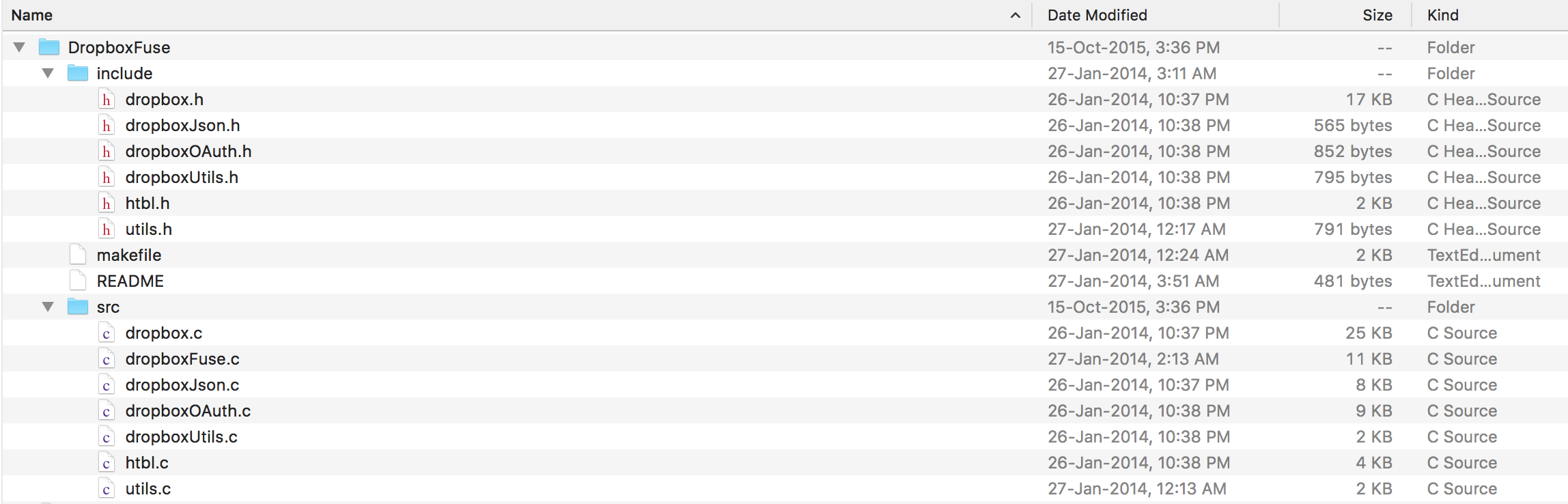
mkdir -p $@

clean:

rm -rf $(OBJ\_PATH) $(BIN\_PATH)

rebuild: clean $(BIN)

### File structure for the project

****

### GCT4.7. LogFiles:

#### 4.7.1. ls Command

The following log results from the execution of the command ls / tmp / fuse (the mount point).

getattr (/): ok

readdir (/): ok

getattr (/ lost + found): ok

getattr (/ root): ok

getattr (/ usr): ok

getattr (/ tmp): ok

getattr (/ lib): ok

...

Analysis:

- The ls command starts with identifying the attributes of the requested path, either

root / (as the mount point overlooking it). This task is performed

getattr by the handler.

- Being identified as a file, ls demand content, resulting in the call

the readdir handler.

- Finally, ls demand attributes of found files and folders.

#### 4.7.2. Command cd

The following log results from the execution of the command cd / tmp / fuse / sys / dev.

getattr (/ sys): ok

getattr (/ sys / dev): ok

access (/ sys / dev)): ok

Analysis:

- Cd ensures that succession / sys and / sys / dev are many issues through

the handler getattr seen before.

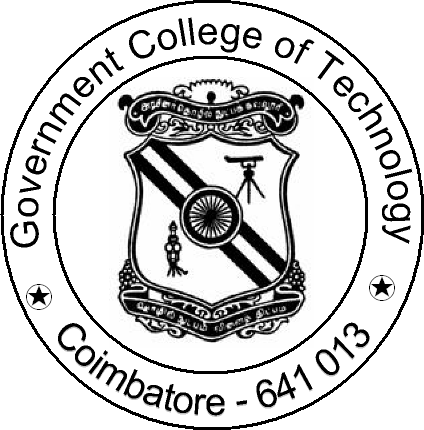
- The order finally asks whether access to the designated path is possible (handler

access).

#### 4.7.3 mkdir command

The following log results from the execution of the command mkdir / tmp / fuse / tmp / foo / bar,

knowing that the file / tmp / foo exists.

getattr (/ tmp): ok

getattr (/ tmp / foo): ok

getattr (/ tmp / foo / bar): No such file or directory

mkdir (/ tmp / foo / bar): ok

getattr (/ tmp / foo / bar): ok

Analysis:

- Mkdir successively ensures that files exist until path / tmp / foo.

- The order notes that the folder does not exist bar and then decided to create it,

causing the call to mkdir handler.

#### 4.7.4. mv Command

The behavior of the mv command is interesting to observe, because it varies

the source and destination location and the type (file or folder). The log

three executions are analyzed (each time to a folder and file):

1. mv / tmp / fuse / tmp / foo / tmp / fuse / tmp / bar

record

getattr (/ tmp): ok

getattr (/ tmp / bar): No such file or dir

getattr (/ tmp / foo): ok

getattr (/ tmp / bar): No such file or dir

rename (/ tmp / foo, / tmp / bar): ok

file

getattr (/ tmp): ok

getattr (/ tmp / bar): No such file or dir

getattr (/ tmp / foo): ok

getattr (/ tmp / bar): No such file or dir

rename (/ tmp / foo, / tmp / bar): ok

Analysis: The movement taking place between the same two locations

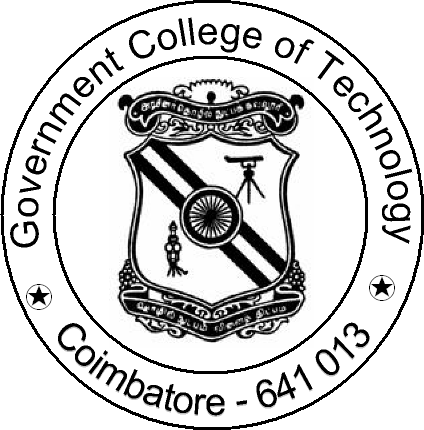
FS

Mv is

only verifies that the destination path is clear and the existing source before

rename using the handler rename.

2. mv / tmp / fuse / tmp / foo / tmp / bar

record

getattr (/ tmp): ok

getattr (/ tmp / foo): ok

readdir (/ tmp / foo): ok

getattr (/ tmp / foo): ok

readdir (/ tmp / foo): ok

rmdir (/ tmp / foo): ok

file

getattr (/ tmp): ok

getattr (/ tmp / foo): ok

open (/ tmp / foo): ok

read (/ tmp / foo): ok

getattr (/ tmp / foo): ok

unlink (/ tmp / foo): ok

Analysis: Here the displacement occurs since

FS FUSE

in the direction of another. There

part performed on the destination (/ tmp / bar) figure by the log as it is

treated with

FS

the destination. The source side, mv proceeds according to the

Type the item to move. We distinguish the use of rmdir and unlink handler

for the removal of the source and open and read the file for playback.

3. mv / tmp / foo / tmp / fuse / tmp / bar

record

getattr (/ tmp): ok

getattr (/ tmp / bar): No such file or dir

getattr (/ tmp / bar): No such file or dir

getattr (/ tmp / bar): No such file or dir

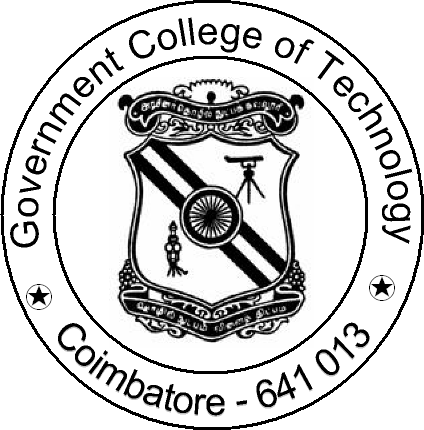
mkdir (/ tmp / bar): ok

getattr (/ tmp / bar): ok

chmod (/ tmp / bar): ok

getattr (/ tmp / bar): ok

chown (/ tmp / bar): ok

getattr (/ tmp / bar): ok

chmod (/ tmp / bar): ok

getattr (/ tmp / bar): ok

file

getattr (/ tmp): ok

getattr (/ tmp / bar): No such file or dir

getattr (/ tmp / bar): No such file or dir

getattr (/ tmp / bar): No such file or dir

mknod (/ tmp / bar): ok

getattr (/ tmp / bar): ok

open (/ tmp / bar): ok

write (/ tmp / bar): ok

chown (/ tmp / bar): ok

getattr (/ tmp / bar): ok

chmod (/ tmp / bar): ok

getattr (/ tmp / bar): ok

Analysis: This time, copying from another place

FS

. So these are actions

companies on the source that are not visible, as performed on the other

FS

. The framework shall in turn create the folder (mkdir handler) or copy the file (handler mknod, open and write) and then copy their attributes (handler chown and chmod).

Furthermore, executing this command displays the following error, allowing us to see the effect of a handler (utimens) which is not implemented: mv: can 't preserve times of' / tmp / fuse / tmp / bar ': Function not implemented

## GCTChapter 5- Conclusion

### 5.1. Technical Conclusion:

# Initially this term paper focuses primarily on the study of the frame-work FUSE to understand how it works. However, it quickly became apparent that the difficulty appear that bit at that level. Indeed, FUSE provides an interface quick to understand and simple to use. The developer can concentrate on the must behave filesystem.

This is the functional needs are on the Dropbox file system - that FUSE

sought the largest share of the work including the development of

API Dropbox C and setting up the cache, necessary for fluidity of the file system.

### 5.2. Future Work

This work properly for Dropbox and major contribution lies on the interfacing the Filesystem with UNIX. Now on thinking about this , any phone working with UNIX as any of its flavour can implement this mode in Super user mode and have additional cloud storage and it does not requuire any app to provide the same. It will embed the cloud storage to the user purpose and can make the utilisation more useful.

## GCTBibliography

1. <https://github.com/Dwii/Dropbox-C>
2. Maurice J. Bach, “The Design of the Unix Operating System”, Prentice Hall of India, 2010.
3. Developing Enterprise Web Services: An Architects Guide (Hewlett-Packard Professional Books by Sandeep Chatterjee
4. Linux libraries-<http://www.linux.org/threads/fuse.6211/>
5. www.dropbox.com