FILE EXPLORER

A report on package submitted by

ANANTHA RAAM G

Roll no. 18PT03

VINAY KUMAR

Roll no. 18PT10

18XT44 - OPERATING SYSTEMS

April 2020

M.Sc. THEORETICAL COMPUTER SCIENCE

DEPARTMENT OF APPLIED MATHEMATICS AND COMPUTATIONAL SCIENCES

PSG COLLEGE OF TECHNOLOGY

COIMBATORE – 641 004.

Table of Contents

Abstract 3

Introduction 4

Description 5

System Calls 6

Tools and Technology 8

Workflow 9

Results and Discussions 11

Conclusion 11

Bibliography 11

Abstract

Every operating system has dominantly two types of interfaces namely Graphical User Interface (GUI) and Command Line Interface (CLI) through which we can access the file system and their contents.In general every operating system by default has a file manager or file explorer that is developed based on GUI.Similarly we can also access the file system with the help of CLI using a set of system commands.But we don't have a built in file explorer based on CLI like that of other file explorers based on GUI.The aim of this project is to develop a file explorer that couples different file operations together so that we can easily access the files from the command line interface in more simplified form.

Introduction

The file explorer we developed is based on the linux ubuntu operating system.In the following report we will be discussing the whereabouts of file explorer which includes project description like tools and technologies used ,system calls used and so on.Next we will see the basic workflow of the project and how to install and use it in local machines.

Description

This file explorer is developed using C and C++ programming languages that work on the Ubuntu operating system.So first we will look at the file structure of our project.The project mainly has three types of files.

- headers.h
- cpp files
- makefile

The headers.h file contains all the header files that are necessary for the project.Additionally it contains a list of global variables and functions.

The cpp files are a list of files that are built to do specific tasks related to the file explorer such as creating a file or folder,deleting a file or folder,renaming a file and so on.

It is more difficult to compile each cpp file separately and combine the final output as we have a large number of files.So instead we use a makefile that is simply a text file which contains all the commands to run the project.So if we run the makefile we can have our desired output.

This is the file structure of our project and now we will discuss about various options provided by file explorer.This file explorer provides us the option of moving through the directories present in the file system.We can also view the file properties like size and permissions of each file.We can be able to create a new file or folder along with the option of deletion.We can also rename the file if necessary along with the option of opening each file with their associated default applications.The major advantage is that all these tasks can be done in the command line by not using the commands but by using the options provided.So this reduces the task of giving commands as input with files as argument.Moreover since it is developed using the most atomic programming language it is an added advantage to the execution time of the file explorer.

System Calls

- **readdir()**

  The readdir() system call function is used to read into a directory.It takes one argument of type DIR* that is the name of the directory to be read and returns a directory pointer.The associated header file is dirent.h.

  Syntax: readdir(DIR*)

- **opendir()**

  The opendir() system call function is used to open a directory and return a pointer to this directory.It takes one argument of type const char * that is the name of the directory to be opened.The associated header file is dirent.h.

  Syntax: opendir(const char*)

- **closedir()**

  The closedir() system call function is used to close the directory that is associated with DIR*.It takes one argument of type DIR* namely the directory name to be closed.The associated header file is dirent.h.

  Syntax: closedir(DIR*)

- **Stat()**

  The stat() system call function is used to check the status of a file like last modified time of a file, file size, permissions and so on.It takes two argument of type const char* and another of type struct stat* that is the corresponding file name and structure details that are associated with the file.The associated header file is sys/stat.h.

Syntax: stat(const char *, struct stat *)

- **system()**

The system() system call function is used to execute native linux commands in c or c++ program.It takes one argument of type const char* that is the linux command to be executed.The associated header file is stdlib.h.

Syntax: system(const char*)

Tools and Technology

- Visual Studio code editor
    - A text editor that is used to develop the project with suitable plugins.

- Github
    - A version control system used to handle the project with speed and accuracy and to monitor the changes made in project

- gcc or g++ compiler
    - A gcc or g++ compiler to compile the project to get the output.

- GNU debugger
    - A gnu debugger called gdb is used to debug the errors that are mainly related to segmentation faults.

Workflow

This part of the report deals with the workflow of the file explorer project.The first step is to run the makefile by using make command.This creates the object file or output file of the file explorer.If we run the object file we will get the list of files and directory available in the current directory. This part of the file explorer is done by DirectoryOperations.cpp.

Then we can easily navigate through the list displayed by using the keys provided so that we can select that particular file or directory using its position in the terminal.The navigation part is taken care of by the object file created from NavigateContent.cpp.

Then we can move into a selected directory (indicated by green color) to explore its content or we can go back to the parent directory to explore its content by using the keys provided.This part of exploration of directory is taken care of by FileExploration.cpp.

Other than directory normal files are also present in the list (indicated by red color) for which file properties can be displayed if or can be renamed if necessary.This is done by FileOperations.cpp.If we want to delete a particular file or folder we can select it and input required key.Finally we can also open the selected file using its default application.This is taken care of by ApplicationViewer.cpp.

Now we will discuss how to install and use the file explorer in local systems.

- To install the file explorer, click the link provided below to clone the repository to your preferred location on your machine.

  https://github.com/ananthu-16/file-explorer.git

- Next cd to the directory of the project.
- $ cd file-explorer

- Next run the makefile to produce an output file.
- $ make
- Next run the output file
- $ ./main

How to use the file-explorer?

- Use ‘ w ‘ or ‘ s ‘ to navigate through the content displayed.
- Press enter to display file properties (if it is a file) or go to the child directory (if it is directory).
- Press backspace to go to the parent directory.
- Use ‘ > ‘ to rename the selected file or directory.
- Use ‘ d ‘ to delete selected files or directories.
- Use ‘ c ‘ to create a directory.
- Use ‘ f ‘ to create a file.
- Use ‘ o ‘ to open the default application for the selected file.
- Use ‘ q ‘ to quit file explorer.

This is the detailed workflow of how the file explorer internally works and how to use it in other local machines.

Results and Discussions

We normally use GUI based file explorer.But this project gives us the brief idea of how a terminal based file explorer works in a more simplified form.We can almost perform all file operations using this file explorer that used the most basic programming language.So if you have any query regarding the file explorer or if you find out any bug in the project you can file an issue in the github repository.

Conclusion

So this is how the file explorer works behind the screen.The main reason to develop this project is to produce a file explorer that works on a terminal.This will be relatively faster than a normal GUI based file explorer because here we have no need to render graphical elements. Moreover we used system calls in most necessary cases so as to avoid mode switching time which also relatively increases the execution time of the file explorer.

Bibliography

- Operating System concepts by Abraham Silberschatz and Peter Baer Galvin.
- Beginning Linux programming by Neil Matthew and Richard Stones.