**Project 4: File System Manager**

Kyungchan Im

Grand Canyon University

Course number: CST - 315

Professor Richardo Citro

Due Apr 16, 2023

**File System Manager**

**[ Introduction ]**

In the field of computer science, file systems play a critical role in managing the storage and organization of data on a computer's hard drive. The operating system (especially command prompt) allows users to create, modify, and delete files and directories, while also managing storage space and ensuring data integrity. This project will be tasked with implementing a file system with a hierarchical directory structure, similar to Unix/Linux, and integrating it with a shell interface. This system will allow users to execute 13 different types of commands to interact with the file system and manage the file and directory lifecycles. Through this project, you will gain valuable experience in file system design and management, as well as command-line interface development.

**[ Explanation and Design ]**

The program is designed to meet several requirements, including the ability to create and rename directories, enabling the program to run the commands simultaneously when multiple commands are given using the prompt, running a user accessing directories, and renaming. The command line interpreter provided by the operating system is a powerful component of the computer system that allows users to interact with the application using text-based line commands, and this program will perform those tasks above. Within the code, I implemented the system code for the C++ environment.

**[ Concept and Analysis ]**

The goal of this project is to develop a file system with a hierarchical directory structure, similar to Unix/Linux, and integrate it with a shell interface. The system should be able to handle various commands, such as creating and renaming directories, moving files across directories, deleting files and directories, duplicating files and directories, and searching for files in a directory tree. The implementation should also enable users to access directories and rename them. The command-line interface provided by the operating system will be used to interact with the application using text-based line commands. To achieve these objectives, the system will be designed using C++ programming language. The design will include classes and functions that represent the file system's directory structure, with the ability to create, modify, and delete directories and files. The implementation will also include a command parser that can handle multiple commands at once, allowing the user to perform various operations on the file system. The parser will take the user's input, tokenize it, and execute the corresponding command. To ensure data integrity and security, the file system will be designed to handle file permissions, ownership, and access control. This will enable users to specify who can access and modify files and directories, thereby preventing unauthorized access and modification.

**[ Implementation ]**

The implementation of the file system and shell interface will be done using C++ programming language. The file system will be represented using classes and data structures, with functions that enable creating, renaming, moving, duplicating, and deleting files and directories. The command parser will be implemented using a switch statement that maps each command to a corresponding function. The file system will be designed to handle file permissions, ownership, and access control using standard Unix/Linux file system permissions. This will ensure data integrity and prevent unauthorized access and modification of files and directories. To implement the shell interface, the program will use the standard input/output streams provided by the operating system. The program will read user input from the standard input stream and display output to the standard output stream. The command parser will take the user's input, tokenize it, and execute the corresponding command using the file system functions. Overall, the implementation will focus on creating a robust and secure file system with a user-friendly command-line interface. The program will be tested using various test cases to ensure the system is functioning correctly and as expected.

**[ Test and Validation ]**

1. **Create and rename directories.**
   1. Create directory: mkdir {directory}

Prompt:



Result:

A picture containing graphical user interface

Description automatically generated

* 1. Rename directory: mv {source} {destination}

Prompt:

Text

Description automatically generated

Result:

A picture containing graphical user interface

Description automatically generated

1. **Delete directories, with special handling of non-empty directories.**
   1. Delete directories: rm -r dir3/dir2/

Prompt:

Text

Description automatically generated

Result:

A picture containing graphical user interface

Description automatically generated

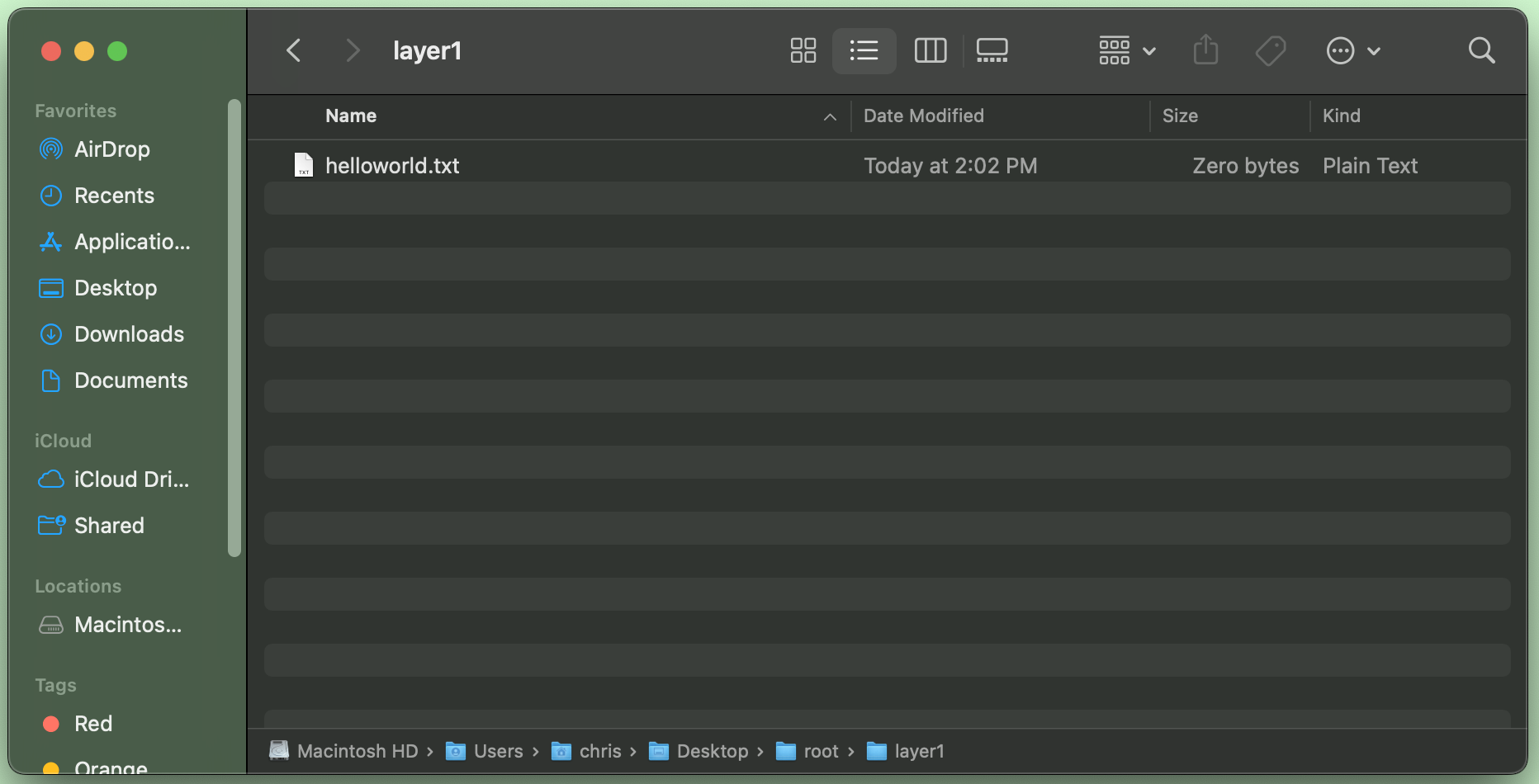
1. **Create, rename, edit, and delete files.**
   1. Create file: touch /path/to/directory/filename.txt

Prompt:

Graphical user interface, text, application

Description automatically generated

Result:



* 1. Rename file: mv oldfile.txt newfile.txt

Prompt:

Text

Description automatically generated

Result:

Graphical user interface, application

Description automatically generated

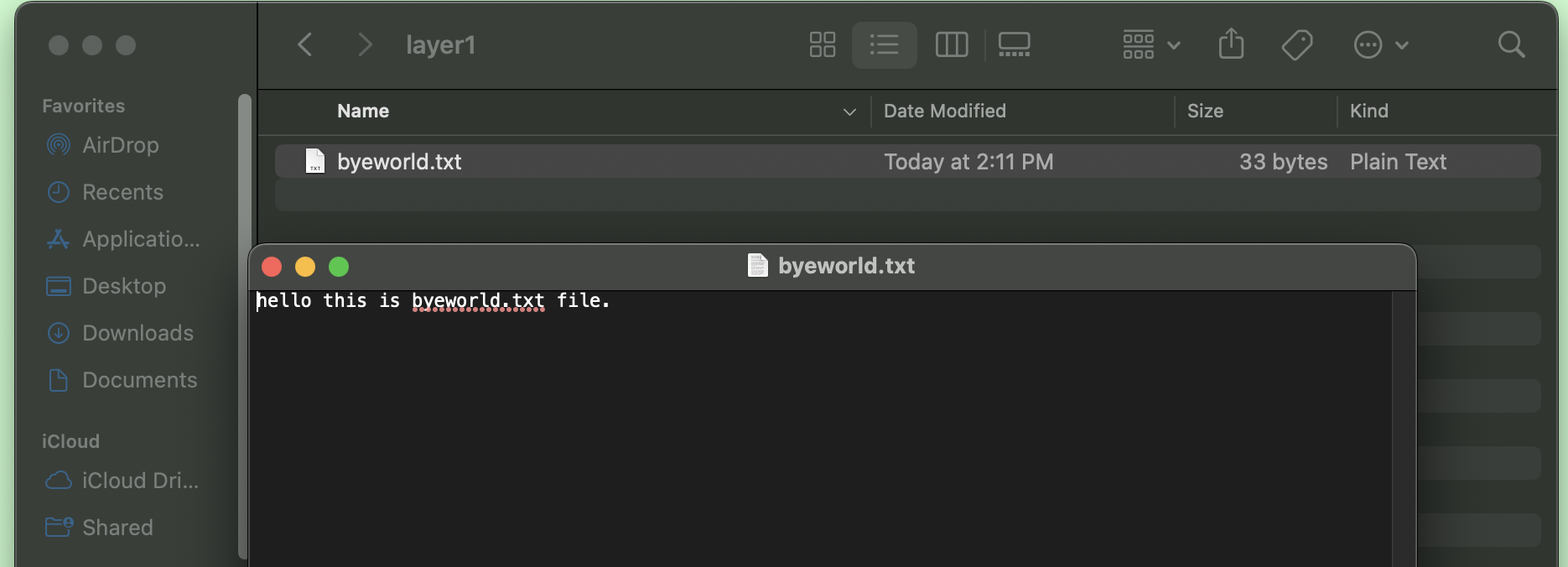
* 1. Edit file: nano filename.txt

Prompt:

Text

Description automatically generated

Result:



* 1. Delete file: rm filename.txt

Prompt:

Text

Description automatically generated

Result:

Graphical user interface

Description automatically generated

1. **When a file is created, a special flag will indicate the amount of data (bytes) to (randomly) generate for it (in order to eliminate the need to open and edit each file)**
2. **Move files across directories:** mv /path/to/source/file /path/to/destination/

Prompt:

Graphical user interface, text, application

Description automatically generated with medium confidence

Result:

Graphical user interface, text, application

Description automatically generated

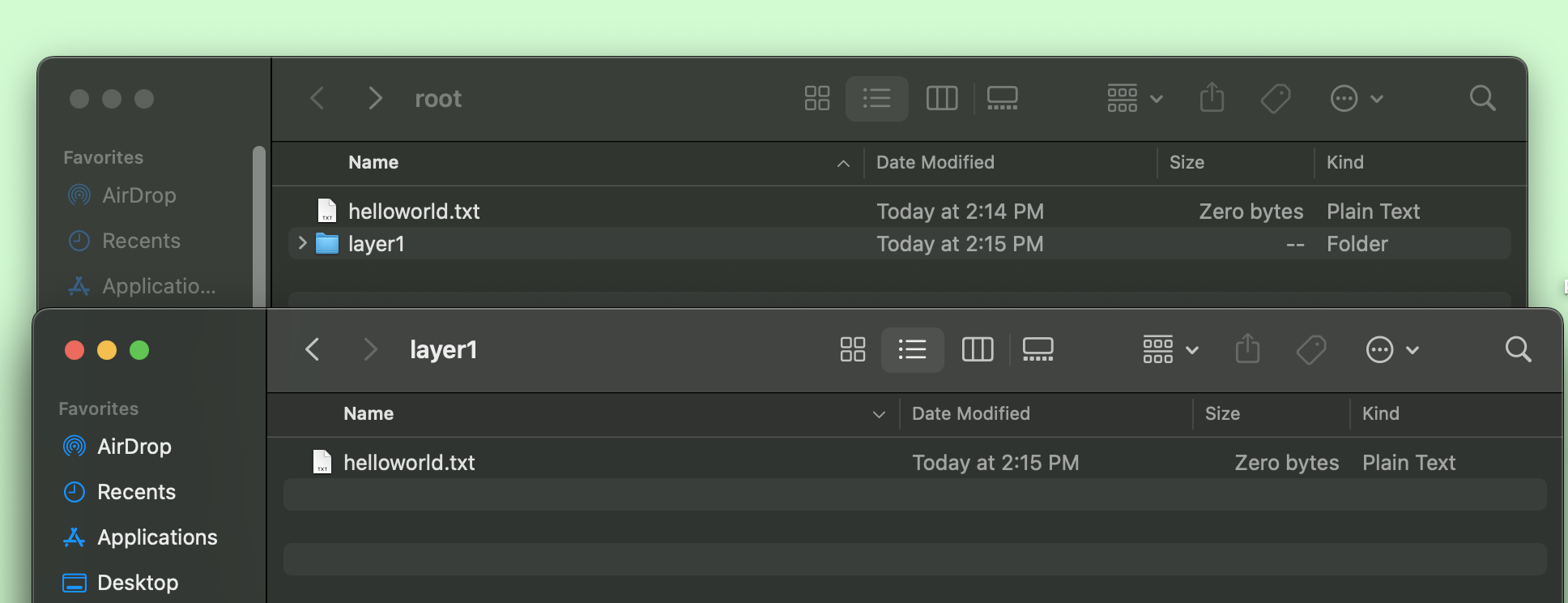
1. **Duplicate files:** cp /path/to/source/file /path/to/destination/

Prompt:

Text

Description automatically generated

Result:

****

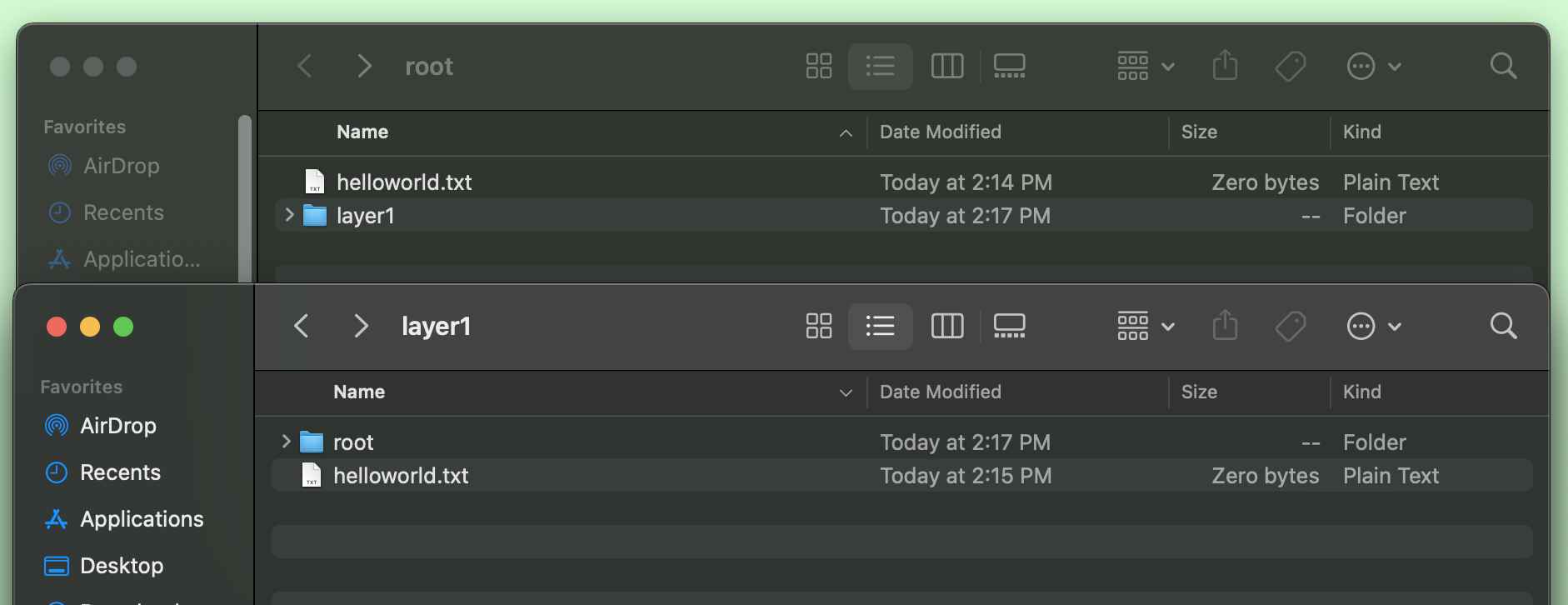
1. **Duplicate directories:** cp -r /path/to/source/directory /path/to/destination/

Prompt:

Graphical user interface, application

Description automatically generated

Result:



1. **Search for a file in a directory tree:** find /path/to/search -name filename.txt

Prompt:

Graphical user interface, application

Description automatically generated

1. **Display a directory tree given a starting node:** ls -RT ./root

Prompt:

Graphical user interface, text, application

Description automatically generated

1. **Get basic information about a file:** ls -l filename.txt

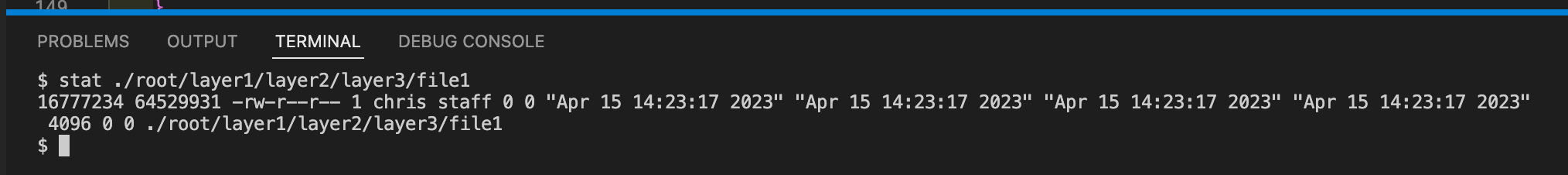
Prompt:

Text

Description automatically generated

1. **Get detailed information about a file (e.g. using a special flag):** stat filename.txt

Prompt:



1. **Get basic information about a directory:** ls -ld directoryname

Prompt:

Text, application

Description automatically generated

1. **Get detailed information about a directory (e.g. using a flag)**

Prompt:

Graphical user interface, text

Description automatically generated

**[ Conclusion ]**

In conclusion, a file system is a critical component of a computer system that manages the storage and organization of data on a hard drive. In this project, we implemented a hierarchical directory structure similar to Unix/Linux and integrated it with a shell interface to enable users to interact with the file system using command-line interface commands. Through this project, we gained valuable experience in file system design and management, as well as command-line interface development. The ability to create, modify, and delete files and directories, as well as to manage storage space and ensure data integrity, are crucial skills in the field of computer science. By developing a file system and shell interface, we have gained a deeper understanding of these concepts and their practical applications.