Group 02

10 January 2025
TMN 4133

SYSTEM PROGRAMMING GROUP PROJECT

Nouf Hasya Ilma binti Badaruddin

78334

Nur Raiyani binti Mohd Yusri Azhar

80685

Rofinna Ellya Embang Anak Umar @ Richard

81056

Sheana Kasih Benedict

78525

CONTENTS

Topics that will be covered

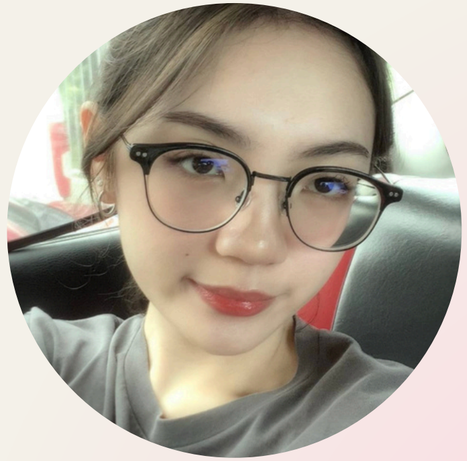
OUR TEAM

Nouf Hasya

78334

Nur Raiyani

80685



Rofinna Ellya

81056

Sheana Kasih

78525

INTRODUCTION

About Our Project

Objective: Develop Supercommand.c - a command-line tool integrating keylogger, directory management, and file operations for efficient system-level tasks.

Key Features:

File operations: Add, open, read, write, delete, change permissions.

Directory operations: Create/delete directories, view contents, current directory tracking.

Keylogger: Record keystrokes with timestamps for monitoring/testing.

Project Breakdown:

Task A: Coding Task - system calls, error handling, and implementation.
Task B: Testing and validating functionality under various scenarios.
Task C: Collaboration review, task delegation, and ChatGPT integration.
Task D: Version control using GitHub for efficient development.
Outcome: Gained expertise in advanced coding, problem-solving, and teamwork strategies.

TASK A

Coding Task

FILE OPERATIONS

getDirectoryName function uses **fgets** to read user input for **directory name** and **file name** before reading, writing, deleting, or changing file permissions, meanwhile **createOpenFile** function uses **open() system call** to create or open a file.

Source code

```c
void getDirectoryName(char *fullPath, size_t size) {
    char dirName[256], filename[256];

    // Prompt for directory name and filename
    printf("Enter directory name: ");
    fgets(dirName, sizeof(dirName), stdin);
    dirName[strcspn(dirName, "\n")] = 0;  // Remove newline character

    printf("Enter filename: ");
    fgets(filename, sizeof(filename), stdin);
    filename[strcspn(filename, "\n")] = 0;  // Remove newline character

    // Combine directory name and filename to create full path
    snprintf(fullPath, size, "%s/%s", dirName, filename);
}
void createOpenFile(const char *fullPath) {
    int fd = open(fullPath, O_CREAT | O_WRONLY, 0644);
    if (fd != -1) {
        printf("File '%s' created successfully.\n", fullPath);
        close(fd);
    } else {
        perror("Error creating file");
    }
```

Output

```
---SUPERCOMMAND OPERATIONS---:
1. File operations
2. Directory operations
3. Keylogger operations
4. Exit
Enter your choice: 1

File Operations:
1. Create a file
2. Delete a file
3. Read a file
4. Write to a file
5. Change file permissions
Enter your choice: 1
Enter directory path: /media/sf_TMN4133SP-Group02-Project/NewTestFolder
Enter filename: CreateTestFile.txt
File '/media/sf_TMN4133SP-Group02-Project/NewTestFolder/CreateTestFile.txt
' created successfully.
```

```
File Operations:
1. Create a file
2. Delete a file
3. Read a file
4. Write to a file
5. Change file permissions
Enter your choice: 2
Enter directory name: NewTestFolder
Enter filename: CreateTestFile.txt
```

FILE OPERATIONS

deleteFile function uses **unlink()** system call to delete a file, and **changeFilePerm** function utilises **chmod()** system call to change file permissions.

Source code

```c
void deleteFile(const char *path) {
    if (unlink(path) == 0) {
        printf("File '%s' deleted successfully.\n", path);
    } else {
        perror("Error deleting file");
    }
}

void changeFilePerm(const char *path, mode_t mode) {
    if (chmod(path, mode) == 0) {
        printf("Permissions for '%s' changed successfully.\n", path);
    } else {
        perror("Error changing file permissions");
    }
}
```

Output

```
File Operations:
1. Create a file
2. Delete a file
3. Read a file
4. Write to a file
5. Change file permissions
Enter your choice: 2
Enter directory name: NewTestFolder
Enter filename: CreateTestFile.txt
File 'NewTestFolder/CreateTestFile.txt' deleted successfully.
```

```
File Operations:
1. Create a file
2. Delete a file
3. Read a file
4. Write to a file
5. Change file permissions
Enter your choice: 5
Enter directory name: Nemo
Enter filename: NemoFile.txt
Enter permissions (e.g., 0644): 0444
Permissions for 'Nemo/NemoFile.txt' changed successfully.
```

FILE OPERATIONS

readFile function uses **read()** system call to read the contents a file, while **writeFile** function uses **write()** system call write contents into a file.

Source code

```c
void readFile(const char *path) {
    char buffer[1024];
    int fd = open(path, O_RDONLY);
    if (fd != -1) {
        ssize_t bytes_read;
        printf("Contents of file '%s':\n", path);
        while ((bytes_read = read(fd, buffer, sizeof(buffer) - 1)) > 0) {
            buffer[bytes_read] = '\0';
            printf("%s", buffer);
        }
        close(fd);
    } else {
        perror("Error reading file");
    }
}

void writeFile(const char *path, const char *content) {
    int fd = open(path, O_WRONLY | O_APPEND);
    if (fd != -1) {
        if (write(fd, content, strlen(content)) != -1) {
            printf("Content written to '%s' successfully.\n", path);
```

Output

```
File Operations:
1. Create a file
2. Delete a file
3. Read a file
4. Write to a file
5. Change file permissions
Enter your choice: 3
Enter directory name: Nemo
Enter filename: newFile.txt
Contents of file 'Nemo/newFile.txt':
Hello world! This is a test file content for the purpose of output demonst
ration.
```

```
File Operations:
1. Create a file
2. Delete a file
3. Read a file
4. Write to a file
5. Change file permissions
Enter your choice: 4
Enter directory name: Nemo
Enter filename: newFile.txt
Enter content to write to the file: Hello world! This is a test file conte
nt for the purpose of output demonstration.
Content written to 'Nemo/newFile.txt' successfully.
```

DEMONSTRATION

FILE OPERATIONS

Refer to presentation video

DIRECTORY OPERATIONS

create_directory function uses **mkdir** to create a directory, while **delete_directory** function uses **rmdir** to delete a directory.

Source code

```c
void create_directory(const char *path) {
    if (mkdir(path, 0755) == 0) {
        printf("Directory '%s' created successfully.\n", path);
    } else {
        perror("Error creating directory");
    }
}

void delete_directory(const char *path) {
    if (rmdir(path) == 0) {
        printf("Directory '%s' deleted successfully.\n", path);
    } else {
        perror("Error deleting directory");
    }
}
```

Output

```
---SUPERCOMMAND OPERATIONS---:
1. File operations
2. Directory operations
3. Keylogger operations
4. Exit
Enter your choice: 2

Directory Operations:
1. Create a directory
2. Delete a directory
3. Print current directory
4. List directory contents
Enter your choice: 1
Enter the directory path to create: /media/sf_TMN4133SP-Group02-Project/tryFo
lder
Directory '/media/sf_TMN4133SP-Group02-Project/tryFolder' created successfull
y.
Directory Operations:
1. Create a directory
2. Delete a directory
3. Print current directory
4. List directory contents
Enter your choice: 2
Enter the directory path to delete: tryFolder
Directory 'tryFolder' deleted successfully.
```

DIRECTORY OPERATIONS

print_current_directory function uses **getcwd** to print current working directory, while **list_directory_contents** function uses **readdir** to list all items within a directory.

Source code

```c
void print_current_directory() {
    char cwd[1024];
    if (getcwd(cwd, sizeof(cwd)) != NULL) {
        printf("Current working directory: %s\n", cwd);
    } else {
        perror("Error getting current working directory");
    }
}

void list_directory_contents(const char *path) {
    DIR *dir;
    struct dirent *entry;

    dir = opendir(path);
    if (dir == NULL) {
        perror("Error opening directory");
        return;
    }

    printf("Contents of directory '%s':\n", path);
    while ((entry = readdir(dir)) != NULL) {
        printf("%s\n", entry->d_name);
    }
```

Output

```
Directory Operations:
1. Create a directory
2. Delete a directory
3. Print current directory
4. List directory contents
Enter your choice: 3
Current working directory: /media/sf_TMN4133SP-Group02-Project

Directory Operations:
1. Create a directory
2. Delete a directory
3. Print current directory
4. List directory contents
Enter your choice: 4
Enter the directory path: /media/sf_TMN4133SP-Group02-Project
Contents of directory '/media/sf_TMN4133SP-Group02-Project':
.
..
.git
.vscode
KambingFile
keylog.txt
newFile.txt
NewTestFolder
Raiyani
README.md
supercommand
supercommand.c
TestFile
testNewFile.txt
```

DEMONSTRATION

DIRECTORY OPERATIONS

Refer to presentation video

KEYLOGGER OPERATIONS

Using **tcgetattr** system call to modify the terminal behavior for capturing raw keystrokes and using the input system call getchar to capture a single character input from the user in real-time

Menu-based system

```
---SUPERCOMMAND OPERATIONS---:
1. File operations
2. Directory operations
3. Keylogger operations
4. Exit
Enter your choice: 3

Keylogger Operations:
1. Start keylogger
Enter your choice: 1
Enter the log file name (or press Enter for default 'keylog.txt'): keylog.txt
Keylogger started. Logging keystrokes to 'keylog.txt'.
Keylogger stopped. Keystrokes saved in 'keylog.txt'.
```

keylog.txt

```
33 Session started at: Tue Jan  7 20:28:45 2025
34
35 Hello World!
```

Source Code

```c
void keylogger(char *logFile) {
    if (!logFile) {
        logFile = "keylog.txt";
    }

    printf("Keylogger started. Logging keystrokes to '%s'.\n", logFile);

    // Open the keylog file
    int fd = open(logFile, O_WRONLY | O_CREAT | O_APPEND, 0644);
    if (fd < 0) {
        perror("Failed to open keylog file");
        return;
    }

    // Add timestamp
    time_t now = time(NULL);
    dprintf(fd, "Session started at: %s\n", ctime(&now));

    // Configure terminal to raw mode for capturing keystrokes
    struct termios oldt, newt;
    tcgetattr(STDIN_FILENO, &oldt);
    newt = oldt;
    newt.c_lflag &= ~(ICANON | ECHO);  // Disable echo and canonical mode
    tcsetattr(STDIN_FILENO, TCSANOW, &newt);

    // Log keystrokes
    char c;
    while (1) {
        c = getchar();
        if (c == 27) {  // ESC key to stop keylogger
            break;
        }
        write(fd, &c, 1);
    }

    // Restore terminal settings
    tcsetattr(STDIN_FILENO, TCSANOW, &oldt);
    close(fd);

    printf("Keylogger stopped. Keystrokes saved in '%s'.\n", logFile);
}
```

DEMONSTRATION

KEYLOGGER OPERATIONS

Refer to presentation video

TASK B

Testing and Evaluation

FILE OPERATION

CREATE OR OPEN FILE

./supercommand -m 1 1 filename.txt

DELETE A FILE

./supercommand -m 1 2 filename.txt

READ A FILE

./supercommand -m 1 3 filename.txt

Source code

```
179             switch (operation) {
180                 case 1: // Create file
181                     createOpenFile(path);
182                     break;
183                 case 2: // Delete file
184                     deleteFile(path);
185                     break;
186                 case 3: // Read file
187                     readFile(path);
188                     break;
189                 case 4: // Write to file
190                     if (argc > 5) {
191                         writeFile(path, argv[5]);
192                     } else {
193                         printf("Content to write is missing.\n");
194                     }
195                     break;
196                 case 5: // Change file permissions
197                     if (argc > 5) {
198                         mode_t mode = strtol(argv[5], NULL, 8);
199                         changeFilePerm(path, mode);
200                     } else {
201                         printf("Permissions mode is missing.\n");
202                     }
203                     break;
204                 default:
205                     printf("Invalid operation for file mode.\n");
```

FILE OPERATION

WRITE INTO A FILE

./supercommand -m 1 4 filename.txt “Content to write”

CHANGE PERMISSIONS OF A FILE

./supercommand -m 1 5 filename.txt 0644

Source code

```
179             switch (operation) {
180                 case 1: // Create file
181                     createOpenFile(path);
182                     break;
183                 case 2: // Delete file
184                     deleteFile(path);
185                     break;
186                 case 3: // Read file
187                     readFile(path);
188                     break;
189                 case 4: // Write to file
190                     if (argc > 5) {
191                         writeFile(path, argv[5]);
192                     } else {
193                         printf("Content to write is missing.\n");
194                     }
195                     break;
196                 case 5: // Change file permissions
197                     if (argc > 5) {
198                         mode_t mode = strtol(argv[5], NULL, 8);
199                         changeFilePerm(path, mode);
200                     } else {
201                         printf("Permissions mode is missing.\n");
202                     }
203                     break;
204                 default:
205                     printf("Invalid operation for file mode.\n");
```

FILE OPERATION

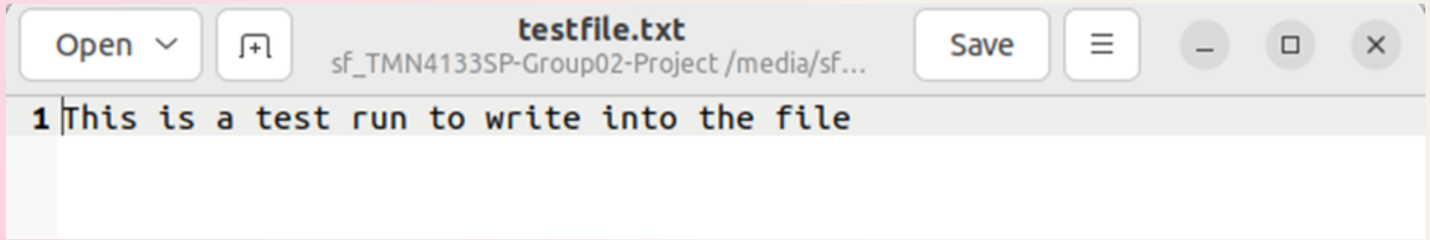
Command line-based system commands for File Operation

```
sheanakb@sheanakb-VirtualBox:/media/sf_TMN4133SP-Group02-Project$ ./supercommand -m 1 1 testfile.txt
File 'testfile.txt' created successfully.
sheanakb@sheanakb-VirtualBox:/media/sf_TMN4133SP-Group02-Project$ ./supercommand -m 1 4 testfile.txt "This
 is a test run to write into the file"
Content written to 'testfile.txt' successfully.
sheanakb@sheanakb-VirtualBox:/media/sf_TMN4133SP-Group02-Project$ ./supercommand -m 1 3 textfile.txt
Error reading file: No such file or directory
sheanakb@sheanakb-VirtualBox:/media/sf_TMN4133SP-Group02-Project$ ./supercommand -m 1 3 testfile.txt
Contents of file 'testfile.txt':
This is a test run to write into the filesheanakb@sheanakb-VirtualBox:/media
bash: ./supercom: No such file or directory
sheanakb@sheanakb-VirtualBox:/media/sf_TMN4133SP-Group02-Project$ ./supercommand -m 1 5 testfile.txt 0644
Permissions for 'testfile.txt' changed successfully.
sheanakb@sheanakb-VirtualBox:/media/sf_TMN4133SP-Group02-Project$ ./supercommand -m 1 2 testfile.txt
File 'testfile.txt' deleted successfully.
```

Commands Tested Successfully:

- Created a file (filename.txt).
- Wrote and read content successfully.
- Changed file permissions to 0644 (Read/Write for User).
- Deleted the file from directory.

Written content into file testfile.txt from command line



The command successfully writes the content into the file name entered by the user.

DIRECTORY OPERATION

CREATE A DIRECTORY

./supercommand -m 2 1 ./new_folder

DELETE A DIRECTORY

./supercommand -m 2 2 ./new_folder

PRINT CURRENT WORKING DIRECTORY

./supercommand -m 2 3

LIST DIRECTORY CONTENT

./supercommand -m 2 4 .

Source code

```
} else if (mode == 2) { // Directory operations
    int operation = atoi(argv[3]);
    const char *path = (argc > 4) ? argv[4] : ".";

    switch (operation) {
        case 1: // Create directory
            create_directory(path);
            break;
        case 2: // Delete directory
            delete_directory(path);
            break;
        case 3: // Print current directory
            print_current_directory();
            break;
        case 4: // List directory contents
            list_directory_contents(path);
            break;
        default:
            printf("Invalid operation for directory mode.\n");
    }
```

Output

```
raiyani@virtualbox:/media/sf_TMN4133SP-Group02-Project$ ./supercommand -m 2 1
 ./filebarutry
Directory './filebarutry' created successfully.
raiyani@virtualbox:/media/sf_TMN4133SP-Group02-Project$ ./supercommand -m 2 2
 ./filebarutry
Directory './filebarutry' deleted successfully.
raiyani@virtualbox:/media/sf_TMN4133SP-Group02-Project$ ./supercommand -m 2 3
Current working directory: /media/sf_TMN4133SP-Group02-Project
raiyani@virtualbox:/media/sf_TMN4133SP-Group02-Project$ ./supercommand -m 2 4
 .
Contents of directory '.':
.
..
.git
.vscode
doryFile.txt
KambingFile
keylog.txt
Raiyani
README.md
supercommand
supercommand.c
TestFile
tryhehe.txt
```

KEYLOGGER OPERATION

KEYLOGGER OPERATION COMMAND

./supercommand -m 3 1 keylog.txt

Source code

```
} else if (mode == 3) { // Keylogger operations
    int operation = atoi(argv[3]);
    char *logfile = (argc > 4) ? argv[4] : "keylog.txt";

    if (operation == 1) {
        keylogger(logfile);
    } else {
        printf("Invalid operation for keylogger mode.\n");
    }
} else {
    printf("Invalid mode.\n");
}
```

Command Line

```
rofinna@rofinna-VirtualBox:/media/sf_TMN4133SP-Group02-Project$ ./supercommand -m 3 1 keylog.txt
Keylogger started. Logging keystrokes to 'keylog.txt'.
Keylogger stopped. Keystrokes saved in 'keylog.txt'.
```

Output

```
27 Session started at: Tue Jan  7 17:39:11 2025
28
29 The keylogger is working!
```

MAN PAGE

What is the Man Page?

- A manual page (man page) was created for the supercommand tool to provide users with a quick reference.
- It details the functionality of File Operations, Directory Operations, and Keylogger using both menu-based and command-line interfaces.

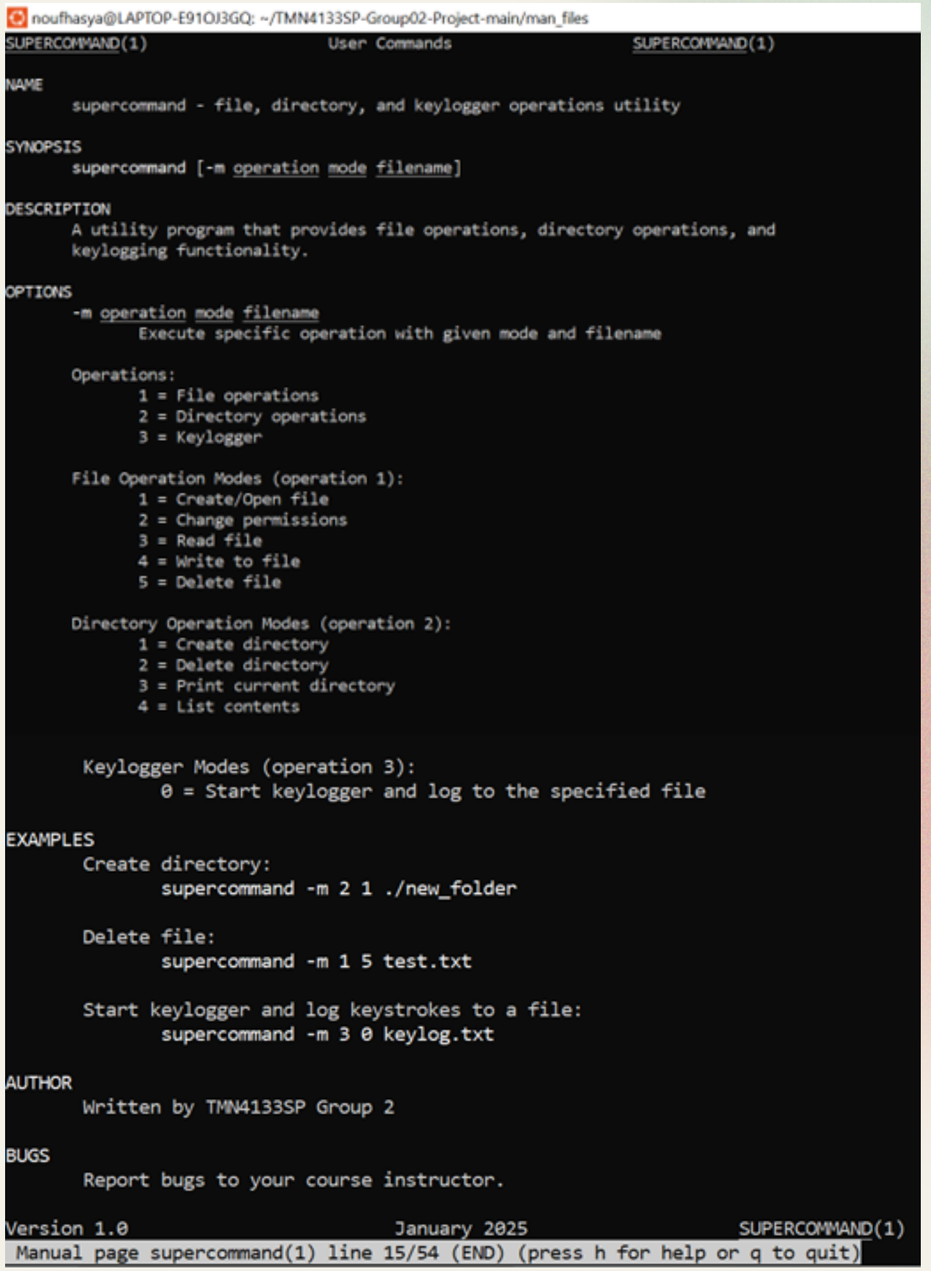
MAN PAGE COMMAND

man supercommand

Command Line

```
noufhasya@LAPTOP-E91OJ3GQ:~/TMN4133SP-Group02
02-Project-main/man_files$ man supercommand
```

OUTPUT



DEMONSTRATION

TASK B

Refer to presentation video

TASK C

Reflection

REFLECTION

USAGE OF CHATGPT AS A TOOL TO HELP STUDENTS OR PROGRAMMERS

Benefits

- The collaborative nature of ChatGPT is highly beneficial.
- It allows users to test concepts, adjust ideas, and solve problems in real time during projects or learning.
- ChatGPT helps achieve a better understanding of material
- It makes programming more approachable
- ChatGPT accelerates project completion while helping users develop self-confidence.
- It is also a helpful tool for exploring new ideas and finding innovative solutions.

Limitations

- Lacks human intuition, which may result in generated code requiring additional attention to meet real-world requirements.
- Highlights the importance of using ChatGPT as a guide rather than a definitive solution.
- Users must:
- Validate and test ChatGPT's outputs.
- Continuously improve their own skills.

TASK DISTRIBUTION & GROUP COMMUNICATION MEDIUM

Sheana Kasih Benedict (78525)

- Code for file operations

Task A

- Create/Open a file
- Change file permissions
- Read a file and print contents
- Write user input to a file
- Delete a file

Task B

- Implement command-line arguments for all file operations.

Nur Raiyani Binti Mohd Yusri Azhar (80685)

- Code for directory operations

Task A

- Create a directory
- Delete a directory
- Print the current working directory
- List directory contents

Task B

- Implement command-line arguments for all directory operations.

Rofinna Ellya Embang Anak Umar @ Richard (81056)

- Code for keylogger operations

Task A

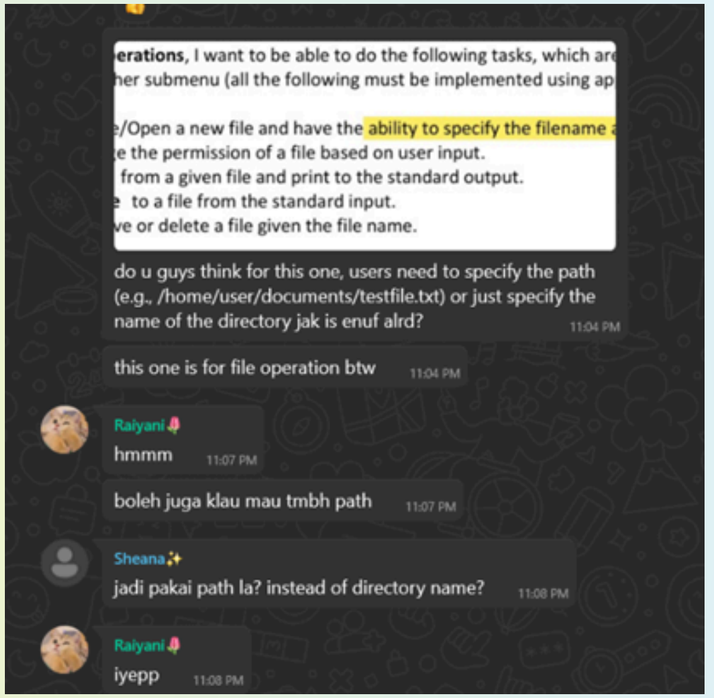
- Create the keylogger functionality
- Run in the background
- Log keystrokes to keylog.txt.
- Add a timestamp at the start of each session.

Task B

- Implement command-line arguments to start the keylogger and specify the log file.

Nouf Hasya Ilma Binti Badaruddin (78334)

- Integrate the menu system for all operations.
- Implement error checking for system calls in all tasks
- Handle invalid inputs gracefully in both menu-based and command-line modes
- Ensure the program prints success/failure messages for all operations
- Conduct final testing and debugging for both Task A and Task B
- Create the man page for the program and ensuring the overall structure is cohesive.



WhatsApp is one of the best communication medium to coordinate and communicate when it comes to group projects:

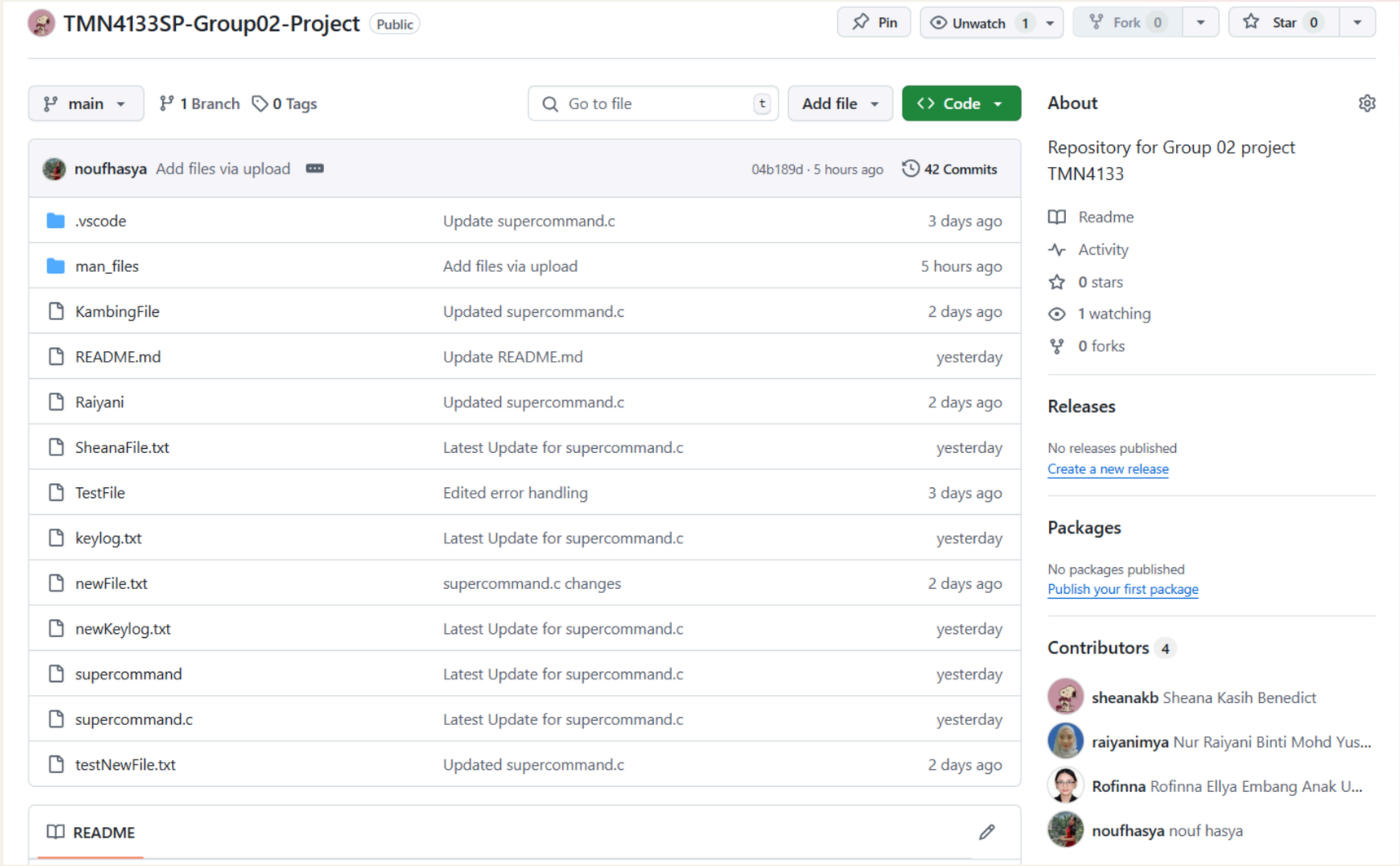
- Daily communication medium.
- Enables file sharing.
- Streamlines communication.

TASK D

Using GitHub

OUR GITHUB REPOSITORY

Link to the repository: https://github.com/sheanakb/TMN4133SP-Group02-Project



COMMIT HISTORY

Current repository
TMN4133SP-Group02-Project

Changes 3 | History

No branches to compare

Nur Raiyani Binti Mohd Yusri Azhar • 2 days ago

update supercommand.c
Nur Raiyani Binti Mohd Yusri Azhar • 2 days ago

update supercommand.c
Nur Raiyani Binti Mohd Yusri Azhar • 2 days ago

Updated supercommand.c
Sheana Kasih Benedict • 3 days ago

update supercommand.c
Nur Raiyani Binti Mohd Yusri Azhar • 3 days ago

Update supercommand.c
Rofinna Ellya Embang Anak Umar @ Richard • 3 days ago

Update supercommand.c
Rofinna Ellya Embang Anak Umar @ Richard • 3 days ago

Edited error handling
Sheana Kasih Benedict • 3 days ago

Update supercommand.c
Rofinna Ellya Embang Anak Umar @ Richard • 3 days ago

Updated supercommand.c
Sheana Kasih Benedict • 3 days ago

New update supercommand.c
Sheana Kasih Benedict • 3 days ago

Update supercommand.c
Sheana Kasih Benedict • 4 days ago

Create supercommand.c
Sheana Kasih Benedict • 4 days ago

Initial commit
Sheana Kasih Benedict • 20 days ago

Current repository
TMN4133SP-Group02-Project

Changes 3 | History

No branches to compare

Sheana Kasih Benedict • 2 days ago

update supercommand.c
Sheana Kasih Benedict • 2 days ago

update supercommand.c
Sheana Kasih Benedict • 2 days ago

update supercommand.c
Sheana Kasih Benedict • 2 days ago

Update supercommand.c
Sheana Kasih Benedict • 2 days ago

Updated supercommand.c
Sheana Kasih Benedict • 2 days ago

Updated supercommand.c
Sheana Kasih Benedict • 2 days ago

Merge branch 'main' of https://github.com/sheanakb/TMN413...
Nur Raiyani Binti Mohd Yusri Azhar • 2 days ago

update supercommand.c
Nur Raiyani Binti Mohd Yusri Azhar • 2 days ago

minor update to supercommand.c
Sheana Kasih Benedict • 2 days ago

supercommand.c changes
Sheana Kasih Benedict • 2 days ago

Updated supercommand.c
Sheana Kasih Benedict • 2 days ago

Debug supercommand.c
Sheana Kasih Benedict • 2 days ago

update supercommand.c
Nur Raiyani Binti Mohd Yusri Azhar • 2 days ago

Current repository
TMN4133SP-Group02-Project

Changes 3 | History

No branches to compare

Add files via upload
nouf hasya • 5 hours ago

Update README.md
Sheana Kasih Benedict • yesterday

Latest Update for supercommand.c
Sheana Kasih Benedict • yesterday

update
Sheana Kasih Benedict • 2 days ago

supercommand.c
Sheana Kasih Benedict • 2 days ago

latest edit supercommand.c
Sheana Kasih Benedict • 2 days ago

supercommand.c update
Sheana Kasih Benedict • 2 days ago

update supercommand.c
Sheana Kasih Benedict • 2 days ago

supercommand.c update again
Sheana Kasih Benedict • 2 days ago

Update supercommand.c
Sheana Kasih Benedict • 2 days ago

Updated supercommand.c
Sheana Kasih Benedict • 2 days ago

Update supercommand.c
Sheana Kasih Benedict • 2 days ago

supercommand.c minor fix
Sheana Kasih Benedict • 2 days ago

OUR README.MD FILE

README

TMN4133SP-Group02-Project

Repository for Group 02 project TMN4133

To compile the code, run at terminal: gcc supercommand.c -o supercommand

To execute the program, run at terminal: ./supercommand

CONCLUSION

Simplify system-level tasks with integrated keylogger functionality, directory management, and file operations in one program.

Supports both menu-based and command-line features to accommodate diverse user preferences.

Highlighted the importance of teamwork and problem-solving skills.

GitHub and thorough testing ensured program accuracy and reliability.

Enhanced programming knowledge and prepared us for more challenging future tasks.

THANK YOU !