FILE EXPLORER

(Operating System)

~RA2211003010438
RA2211003010414

- Objective
- Problem Statement Architecture/ Flow chart
- Hardware/Software requirements
- Implementation- Code snippet
- Results- Screen Shots of Output
- Conclusion

Objective

The objective is to develop a robust and efficient system file manager that utilizes the CPU, memory and Disk.

The objective is to provide an efficient framework to the end user to efficiently manage files in the system and to easily access, edit, modify and deleted them.

Problem Statement

In today's fast-paced technological landscape, efficient utilization of computing filesis critical for the optimal performance of the organization. This poses the need for an efficient file management system which users can rely on to manage all their files with a secured layout and easily locate, edit, modify, delete and relocate them. Such a file manager would prove really useful on the following grounds and may help organizations in functioning more efficiently by reducing their workloads.

REQUIREMENTS

SRM INSTITUTE OF SCIENCE & TECHNOLOGY
(Deemed to be University u/s 3 of UGC Act, 1956)

Non-Functional Requirements:

These requirements describe the quality attributes of the system. Non-functional requirements for your project might include:

- Usability
- Performance
- Reliability
- Security
- Cross-platform Compatibility
- Scalability
- Accessibility

Functional Requirements:

These requirements define what the system should do and include features and capabilities. For a directory and file management project, functional requirements might include:

- Navigation Directories and subdirectories
- Listing the contents of a directory
- Creating new directories
- Deleting directories
- Error handling for various scenarios

Implementation (code snippets)


```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <dirent.h>
#include <unistd.h>
#include <sys/stat.h>
#include <sys/types.h>
#include <errno.h>

// Function to list the contents of a directory
void list_directory(const char *path) {
    struct dirent *entry; // Structure for directory entry
    struct stat statbuf;  // Structure to store file
information
    char filepath[1024];  // Array to store the file path

    DIR *dir = opendir(path); // Open the directory

    if (dir == NULL) {
        perror("Unable to read directory"); // Print an error
message if the directory cannot be opened
        return;
    }
```

```
    while ((entry = readdir(dir)) != NULL) {
        snprintf(filepath, sizeof(filepath), "%s/%s", path, entry->d_name); //
Construct the complete file path
        if (stat(filepath, &statbuf) == -1) {
            perror("Error getting file status"); // Print an error message if
file information cannot be obtained
            continue; // Skip to the next entry
        }

        if (S_ISDIR(statbuf.st_mode)) {
            printf("[DIR]  %s\n", entry->d_name); // Print the directory name
        } else {
            printf("[FILE] %s (Size: %lld bytes)\n", entry->d_name, (long
long)statbuf.st_size); // Print the file name and size
        }
    }

    closedir(dir); // Close the directory
}

// Function to create a text file
```

```c
void create_text_file(const char *filename) {
    FILE *file = fopen(filename, "w"); // Open the file in write mode
    if (file == NULL) {
        perror("Error creating the text file"); // Print an error message if the file cannot
be created
        return;
    }
    fclose(file); // Close the file
}

int main() {
    char current_directory[1024]; // Array to store the current directory path
    getcwd(current_directory, sizeof(current_directory)); // Get the current directory path

    char choice[256]; // Array to store the user's choice

    while (1) {
        printf("Current directory: %s\n", current_directory);
        list_directory(current_directory); // List the contents of the current directory
```

```
        printf("\n[Enter 'q' to quit, '..' to go up a level, 'mkdir' to create a directory,");
        printf(" 'rmdir' to remove a directory, 'rmfile' to remove a file,");
        printf(" 'create' to create a text file, or a directory name to navigate into]\n");
        printf("Choose an action: ");
        if (scanf("%255s", choice) != 1) {
            fprintf(stderr, "Input error.\n"); // Print an error message for input error
            break;
        }

        if (strcmp(choice, "q") == 0) {
            break; // Exit the program if the user enters 'q'
        } else if (strcmp(choice, "..") == 0) {
            if (chdir("..") == -1) {
                perror("Error navigating up"); // Print an error message if navigation fails
            } else {
                if (getcwd(current_directory, sizeof(current_directory)) == NULL) {
                    perror("Error getting current directory"); // Print an error message if getting the
current directory fails
                    break;
                }
```

```
} else if (strcmp(choice, "mkdir") == 0) {
    printf("Enter directory name to create: ");
    if (scanf("%255s", choice) != 1) {
        fprintf(stderr, "Input error.\n"); // Print an error message for input error
        break;
    }
    if (mkdir(choice) == -1) {
        perror("Error creating directory"); // Print an error message if directory creation fails
    }
} else if (strcmp(choice, "rmdir") == 0) {
    printf("Enter directory name to remove: ");
    if (scanf("%255s", choice) != 1) {
        fprintf(stderr, "Input error.\n"); // Print an error message for input error
        break;
    }
    if (mkdir(choice) == -1) {
        perror("Error creating directory"); // Print an error message if directory creation fails
    }
} else if (strcmp(choice, "rmdir") == 0) {
    printf("Enter directory name to remove: ");
    if (scanf("%255s", choice) != 1) {
```

```
                fprintf(stderr, "Input error.\n"); // Print an error message
for input error
                break;
            }
            if (remove(choice) == -1) {
                perror("Error removing file"); // Print an error message if
file removal fails
            }
        } else if (strcmp(choice, "create") == 0) {
            printf("Enter text file name to create: ");
            if (scanf("%255s", choice) != 1) {
                fprintf(stderr, "Input error.\n"); // Print an error message
for input error
                break;
            }
            create_text_file(choice); // Create a text file with the provided
name
        } else {
            if (chdir(choice) == -1) {
```


```
                perror("Error navigating to directory"); // Print an
error message if navigation to a directory fails
            } else {
                if (getcwd(current_directory, sizeof(current_directory))
== NULL) {
                    perror("Error getting current directory"); // Print
an error message if getting the current directory fails
                    break;
                }
            }
        }
    }

    return 0;
}
```

Algorithm

1. Start
2. Initialize variables and data structures.
3. Display the current directory and its contents.
4. Offer a menu of actions to the user.
5. Accept the user's choice.
6. Process the choice:
    - 6.1 Navigate up or down directories.
    - 6.2 Create or remove directories/files.
    - 6.3 Quit the program.
7. Repeat until the user quits.
8. End

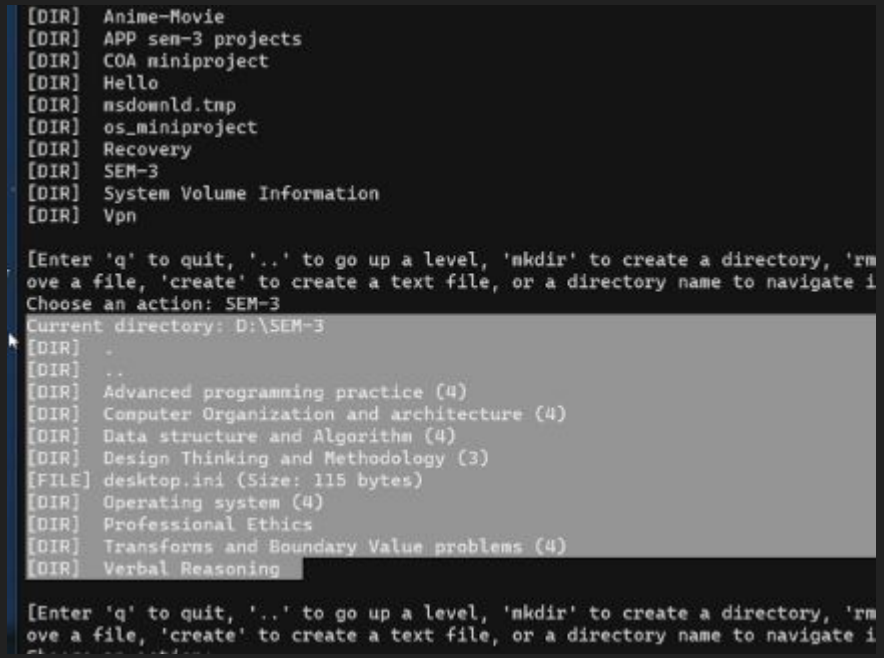
Result and Analysis

SRM INSTITUTE OF SCIENCE & TECHNOLOGY
(Deemed to be University u/s 3 of UGC Act, 1956)

```
Current directory: D:\os_miniproject
[DIR]  .
[DIR]  ..
[DIR]  .vscode
[FILE] directory_manager.c (Size: 3469 bytes)
[FILE] directory_manager.exe (Size: 432896 bytes)

[Enter 'q' to quit, '..' to go up a level, 'mkdir' to create a directory, 'rmdir' to remove a directory, 'rmfile' to remove a file, or a directory name to n
avigate into]
Choose an action: ..
Current directory: D:\
[DIR]  $RECYCLE.BIN
[DIR]  1.Old_data
[DIR]  2.Games
[DIR]  3.Wallapaer
[DIR]  4.ComputerScience_EXP
[DIR]  5.Illustrator
[DIR]  6.Udemy
[DIR]  7.Premiere pro
[DIR]  Anime-Movie
[DIR]  APP sem-3 projects
[DIR]  COA miniproject
[DIR]  msdownld.tmp
[DIR]  os_miniproject
[DIR]  Recovery
[DIR]  SEM-3
[DIR]  System Volume Information
[DIR]  Vpn

[Enter 'q' to quit, '..' to go up a level, 'mkdir' to create a directory, 'rmdir' to remove a directory, 'rmfile' to remove a file, or a directory name to n
avigate into]
Choose an action:
```


```
[Enter 'q' to quit, '..' to go up a level, 'mkdir' to create a directory, 'rmdir' to remove a directory, 'rmfile' to remove a file, or a directory name to n
avigate into]
Choose an action: mkdir
Enter directory name to create: Hello
Current directory: D:\
[DIR]  $RECYCLE.BIN
[DIR]  1.Old_data
[DIR]  2.Games
[DIR]  3.Wallapaer
[DIR]  4.ComputerScience_EXP
[DIR]  5.Illustrator
[DIR]  6.Udemy
[DIR]  7.Premiere pro
[DIR]  Anime-Movie
[DIR]  APP sem-3 projects
[DIR]  COA miniproject
[DIR]  Hello
[DIR]  msdownld.tmp
[DIR]  os_miniproject
[DIR]  Recovery
[DIR]  SEM-3
[DIR]  System Volume Information
[DIR]  Vpn

[Enter 'q' to quit, '..' to go up a level, 'mkdir' to create a directory, 'rmdir' to remove a directory, 'rmfile' to remove a file, or a directory name to n
avigate into]
Choose an action: rmdir
Enter directory name to remove: Hello
Current directory: D:\
```

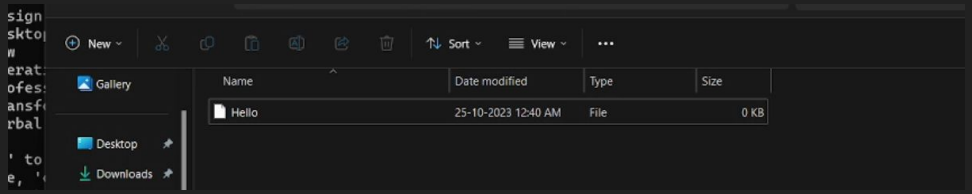
.. to go up a level



Mkdir > name to create and rmdir to remove

Name of the Folder can be written to get into it

```
Enter text file name to create: Hello
Current directory: D:\SEM-3\New
[DIR]  .
[DIR]  ..
[FILE] Hello (Size: 0 bytes)
```

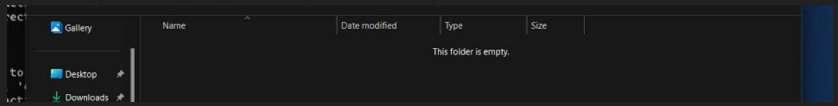


‘Create’ followed by name to create a file:

rmfile>name to delete it.

Q to quit.

```
Choose an action: rmfile
Enter file name to remove: Hello
Current directory: D:\SEM-3\New
```



Conclusion

In conclusion, this directory and file management project provides a straightforward yet functional interface for users to navigate directories, create, remove directories, and manage files. It offers a practical utility for basic file operations in a user-friendly manner.

SRM
INSTITUTE OF SCIENCE & TECHNOLOGY
(Deemed to be University u/s 3 of UGC Act, 1956)