

# **Assignment 1(LinuxOS)**

**Objective:** Develop a console-based file explorer application in C++ that interfaces with the Linux operating system to manage files and directories.

## **Day-wise Tasks:**

**Day 1:** Design the application structure and setup the development environment. Start with basic file operations like listing files in a directory.

### **Code:**

```
#include <iostream>

#include <dirent.h>

#include <cstring>

#include <unistd.h>

void listFiles(const std::string &path)
{
    DIR *dir = opendir(path.c_str());

    if (dir == nullptr)
    {
        std::cerr << "Error opening directory: " << path << std::endl;
        return;
    }

    struct dirent *entry;
    while ((entry = readdir(dir)) != nullptr)
    {
        std::cout << entry->d_name << std::endl;
    }

    closedir(dir);
}
```

```

}

int main()
{
    std::string currentPath = ".";

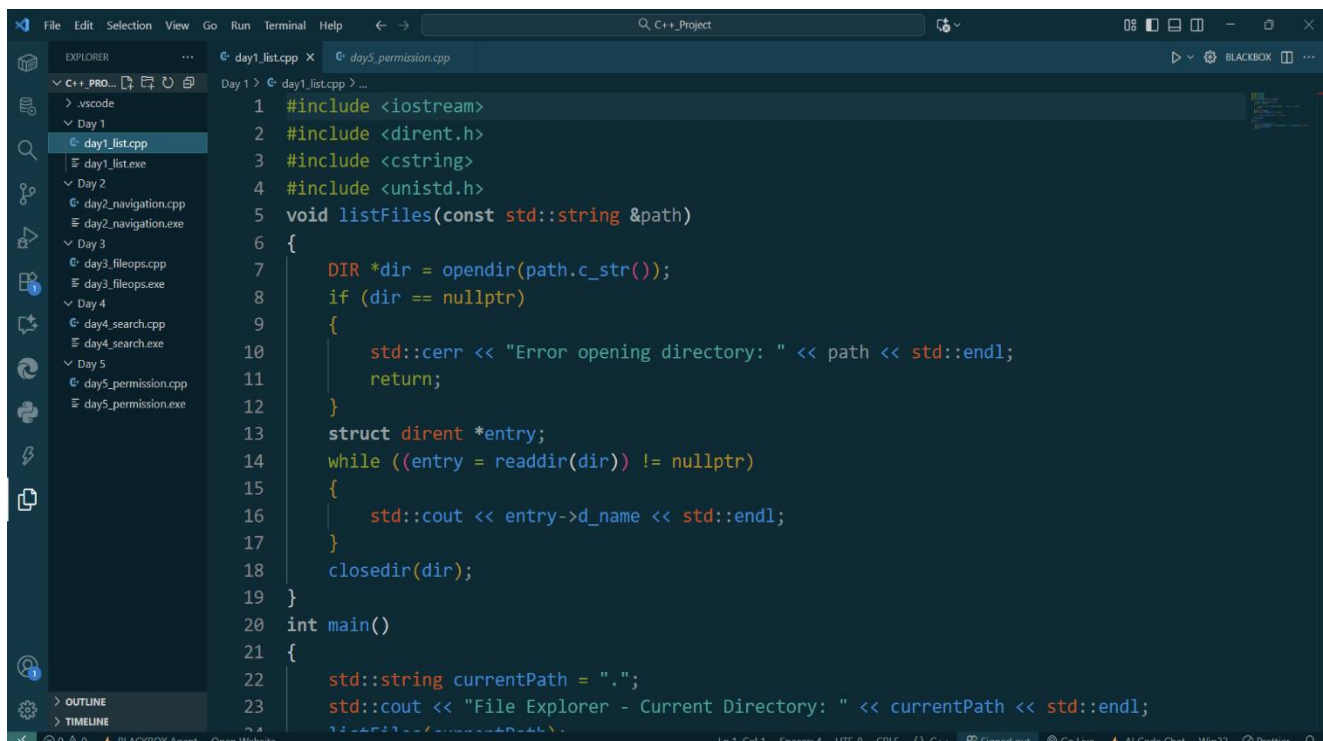
    std::cout << "File Explorer - Current Directory: " << currentPath << std::endl;

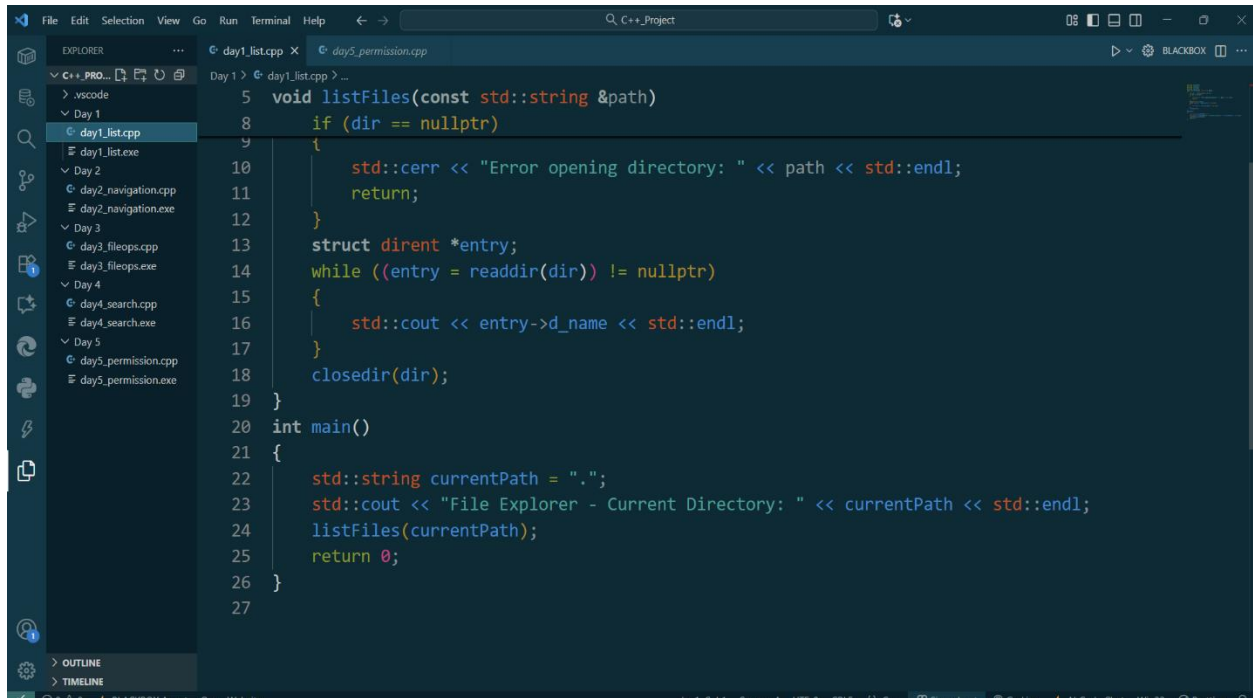
    listFiles(currentPath);

    return 0;
}

```

## Screenshots:





**Day 2:** Implement file and directory navigation features. Enable the user to move through directories.

### **Code:**

```
#include <iostream>

#include <dirent.h>

#include <cstring>

#include <unistd.h>

#include <limits>

void listFiles(const std::string &path)
{
    DIR *dir = opendir(path.c_str());

    if (dir == nullptr)
    {
        std::cerr << "Error opening directory: " << path << std::endl;
        return;
    }
}
```

```

struct dirent *entry;

while ((entry = readdir(dir)) != nullptr)
{
    std::cout << entry->d_name << std::endl;
}

closedir(dir);
}

int main()
{
    std::string currentPath = ".";

    std::string command;

    while (true)
    {
        std::cout << "File Explorer - Current Directory: " << currentPath << std::endl;

        listFiles(currentPath);

        std::cout << "Enter command (cd <dir> or exit): ";

        std::getline(std::cin, command);

        if (command == "exit")
            break;

        if (command.substr(0, 3) == "cd ")
        {
            std::string newDir = command.substr(3);

            if (chdir(newDir.c_str()) == 0)
            {
                char cwd[1024];

                getcwd(cwd, sizeof(cwd));

                currentPath = cwd;
            }
            else
            {

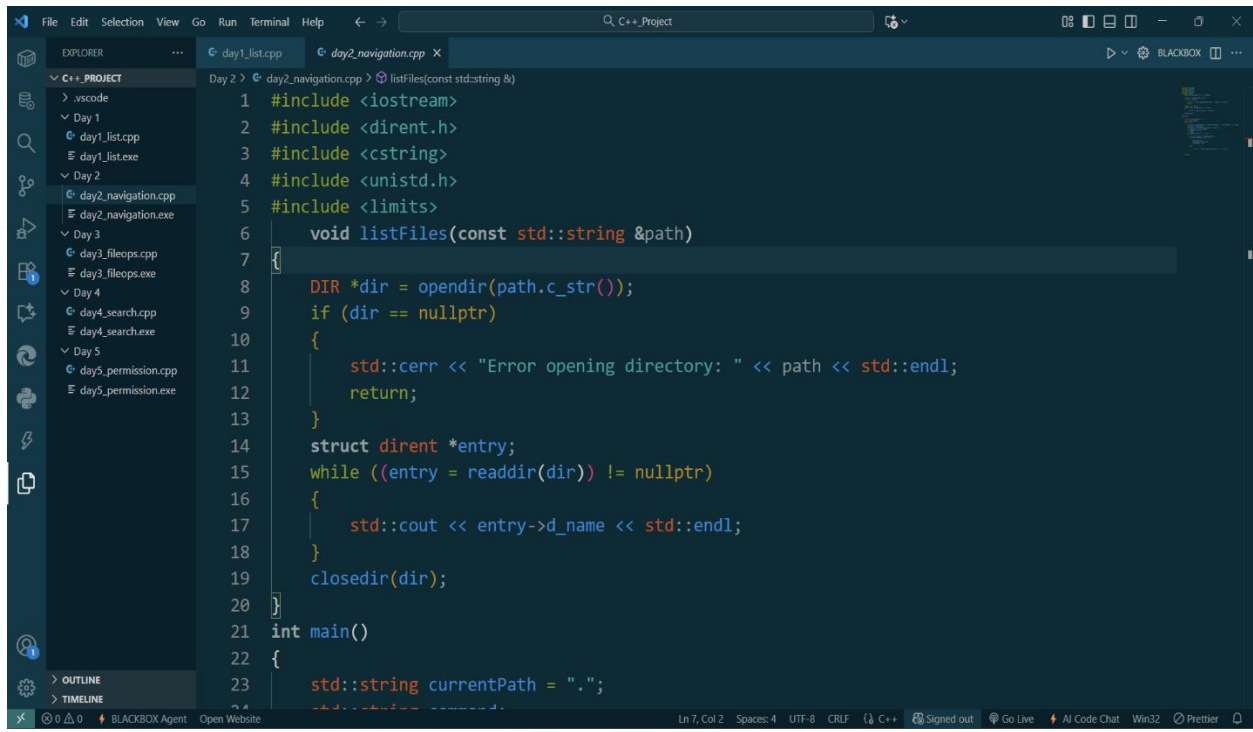
```

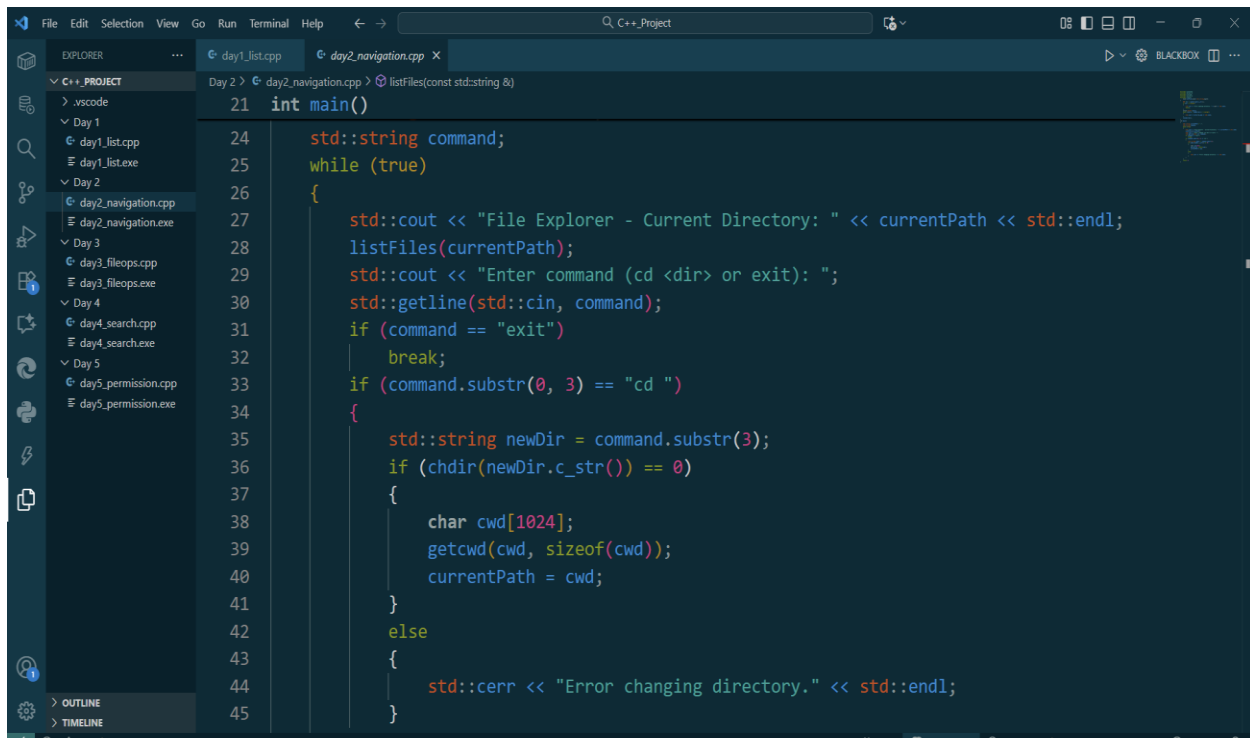
```

        std::cerr << "Error changing directory." << std::endl;
    }
}
}
return 0;
}

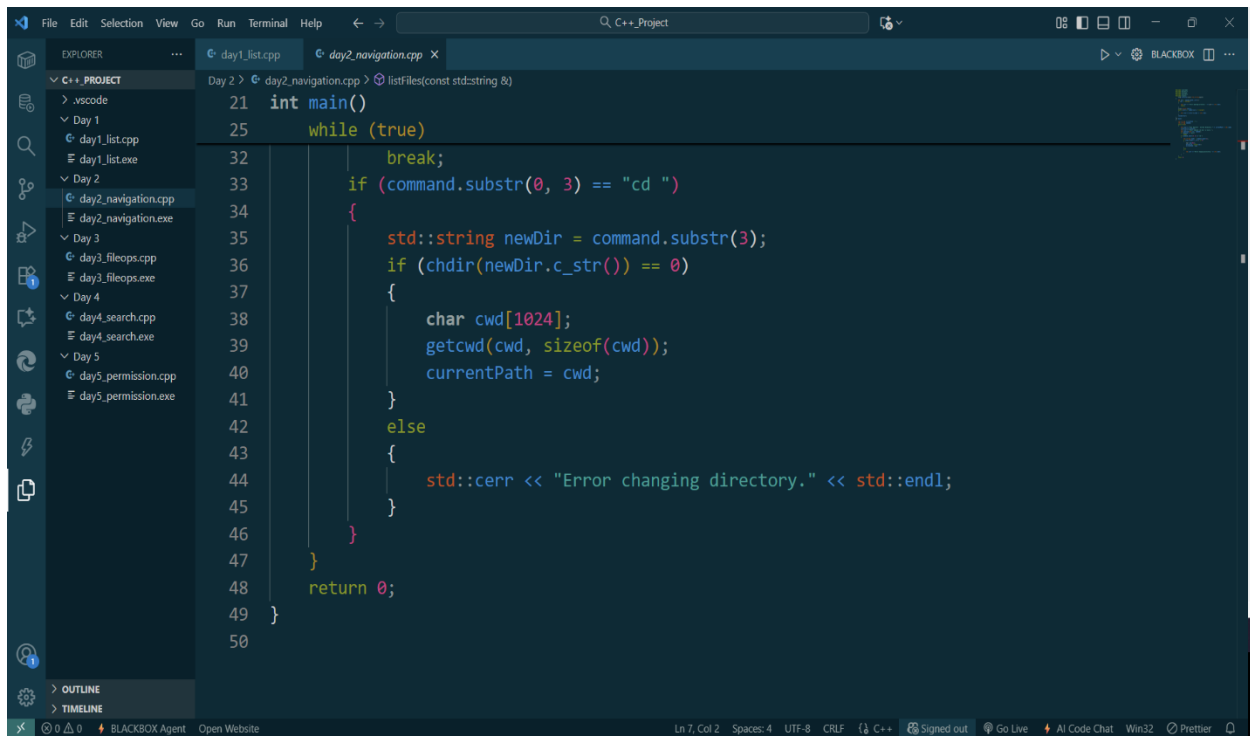
```

## Screenshot:





```
21 int main()
24     std::string command;
25     while (true)
26     {
27         std::cout << "File Explorer - Current Directory: " << currentPath << std::endl;
28         listFiles(currentPath);
29         std::cout << "Enter command (cd <dir> or exit): ";
30         std::getline(std::cin, command);
31         if (command == "exit")
32             break;
33         if (command.substr(0, 3) == "cd ")
34         {
35             std::string newDir = command.substr(3);
36             if (chdir(newDir.c_str()) == 0)
37             {
38                 char cwd[1024];
39                 getcwd(cwd, sizeof(cwd));
40                 currentPath = cwd;
41             }
42             else
43             {
44                 std::cerr << "Error changing directory." << std::endl;
45             }
46         }
47     }
48     return 0;
49 }
```



```
21 int main()
25     while (true)
26     {
27         break;
28     }
29     if (command.substr(0, 3) == "cd ")
30     {
31         std::string newDir = command.substr(3);
32         if (chdir(newDir.c_str()) == 0)
33         {
34             char cwd[1024];
35             getcwd(cwd, sizeof(cwd));
36             currentPath = cwd;
37         }
38         else
39         {
40             std::cerr << "Error changing directory." << std::endl;
41         }
42     }
43 }
44 return 0;
45 }
```

**Day 3:** Add file manipulation capabilities (copy, move, delete, create).

**Code:**

```
#include <iostream>
```

```
#include <dirent.h>
```

```
#include <unistd.h>
```

```
#include <cstring>
```

```
#include <fstream>
```

```
void listFiles(const char *path)
```

```
{
```

```
    DIR *dir = opendir(path);
```

```
    if (dir)
```

```
    {
```

```
        struct dirent *entry;
```

```
        while ((entry = readdir(dir)))
```

```
        {
```

```
            std::cout << entry->d_name << std::endl;
```

```
        }
```

```
        closedir(dir);
```

```
    }
```

```
}
```

```
void copyFile(const char *src, const char *dst)
```

```
{
```

```
    std::ifstream in(src, std::ios::binary);
```

```
    std::ofstream out(dst, std::ios::binary);
```

```
    out << in.rdbuf();
```

```
}
```

```
int main()
```

```

{
    char currentPath[1024] = ".";
    std::string command;
    while (true)
    {
        std::cout << "Current Directory: " << currentPath << std::endl;
        listFiles(currentPath);

        std::cout << "Command (cd <dir>, copy <src> <dst>, move <src> <dst>, delete <file>, create <file>,
or exit): ";
        std::getline(std::cin, command);
        if (command == "exit")
            break;
        if (command.substr(0, 3) == "cd ")
        {
            std::string newDir = command.substr(3);
            if (chdir(newDir.c_str()) == 0)
            {
                getcwd(currentPath, sizeof(currentPath));
            }
        }
        else if (command.substr(0, 5) == "copy ")
        {
            size_t pos = command.find(' ', 5);
            std::string src = command.substr(5, pos - 5);
            std::string dst = command.substr(pos + 1);
            copyFile(src.c_str(), dst.c_str());
        }
        else if (command.substr(0, 5) == "move ")
        {
            size_t pos = command.find(' ', 5);
            std::string src = command.substr(5, pos - 5);

```



```

        std::string dst = command.substr(pos + 1);

        rename(src.c_str(), dst.c_str());
    }

    else if (command.substr(0, 7) == "delete ")
    {
        std::string file = command.substr(7);

        remove(file.c_str());
    }

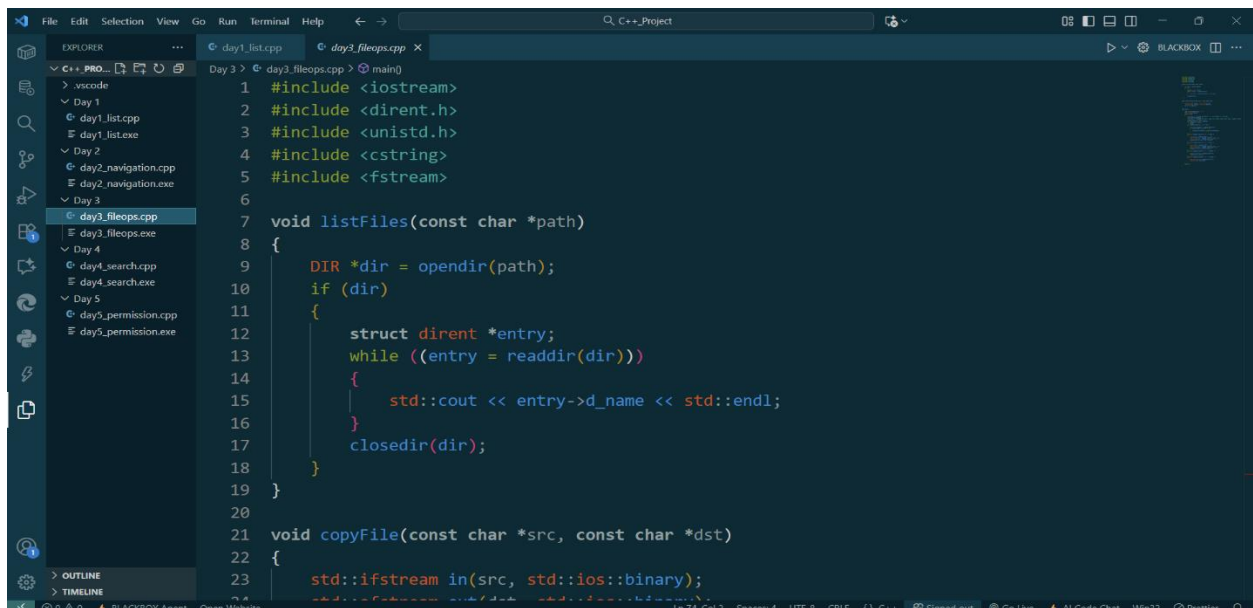
    else if (command.substr(0, 7) == "create ")
    {
        std::string file = command.substr(7);

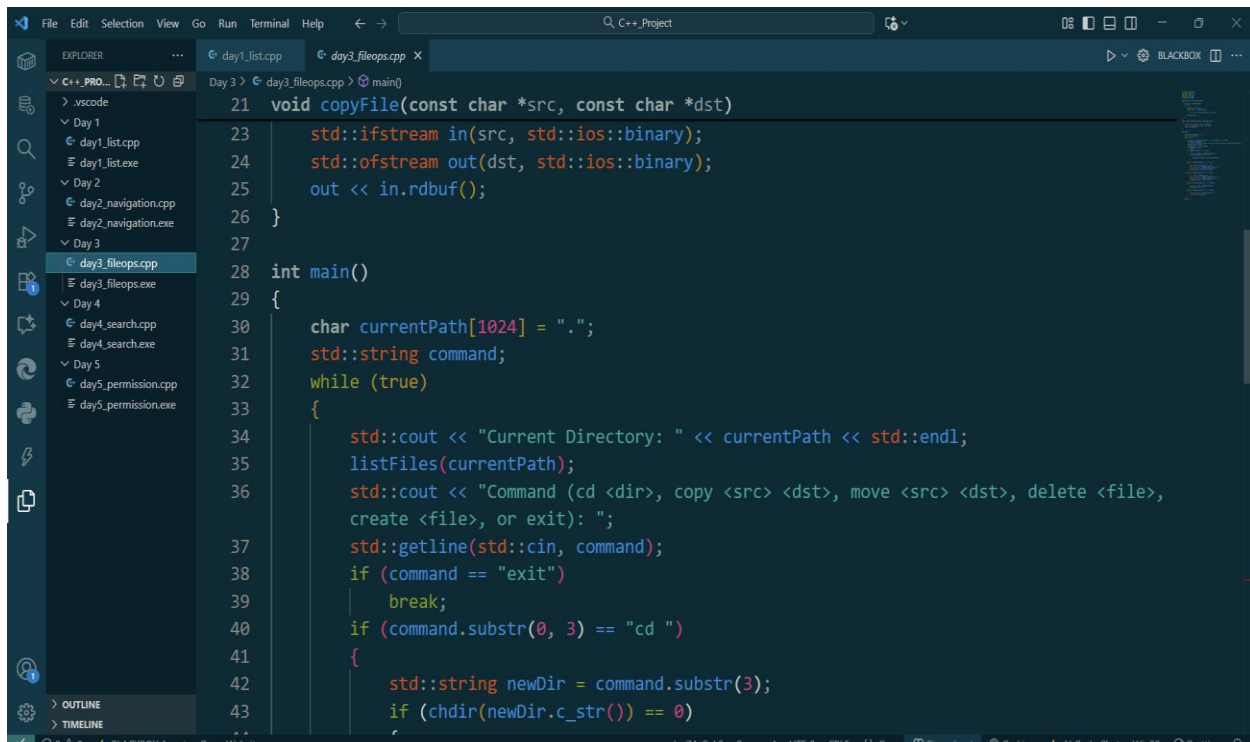
        std::ofstream out(file);
    }
}

return 0;
}

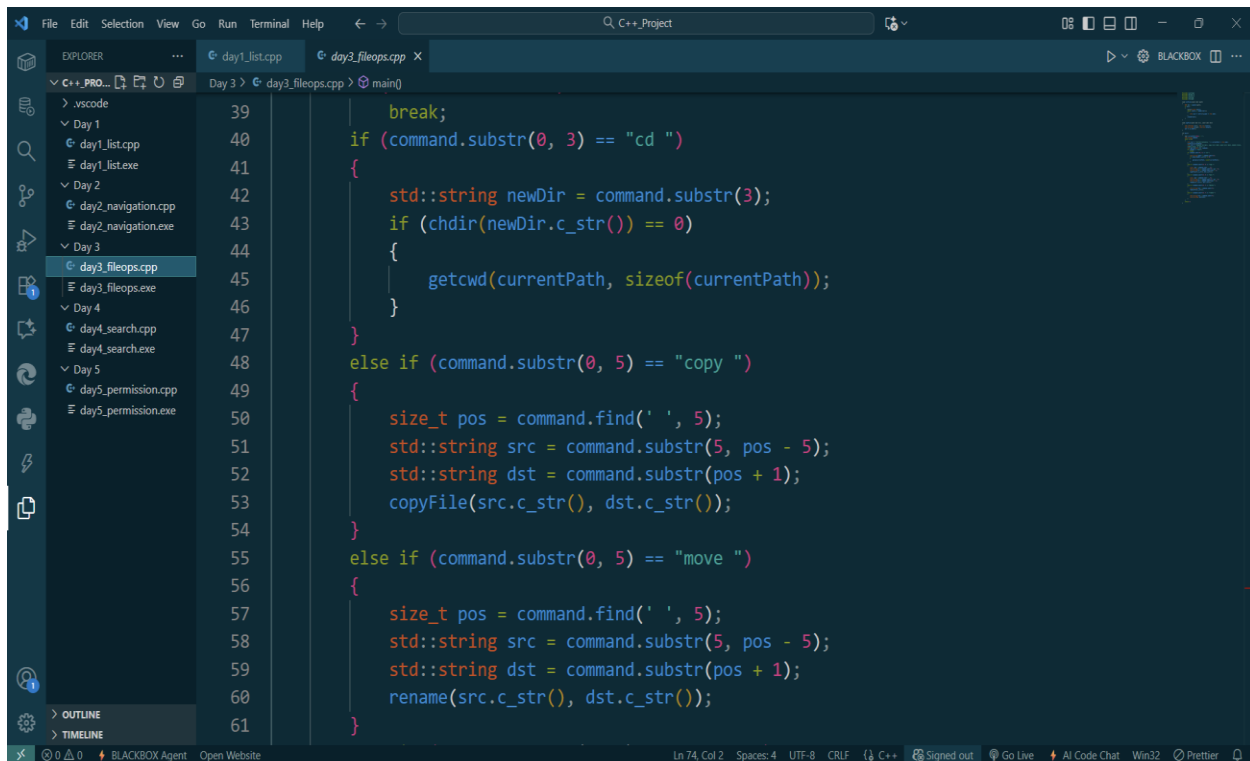
```

## Screenshots:





```
21 void copyFile(const char *src, const char *dst)
22 {
23     std::ifstream in(src, std::ios::binary);
24     std::ofstream out(dst, std::ios::binary);
25     out << in.rdbuf();
26 }
27
28 int main()
29 {
30     char currentPath[1024] = ".";
31     std::string command;
32     while (true)
33     {
34         std::cout << "Current Directory: " << currentPath << std::endl;
35         listFiles(currentPath);
36         std::cout << "Command (cd <dir>, copy <src> <dst>, move <src> <dst>, delete <file>,
37         create <file>, or exit): ";
38         std::getline(std::cin, command);
39         if (command == "exit")
40             break;
41         if (command.substr(0, 3) == "cd ")
42         {
43             std::string newDir = command.substr(3);
44             if (chdir(newDir.c_str()) == 0)
```



```
39             break;
40         if (command.substr(0, 3) == "cd ")
41         {
42             std::string newDir = command.substr(3);
43             if (chdir(newDir.c_str()) == 0)
44             {
45                 getcwd(currentPath, sizeof(currentPath));
46             }
47         }
48         else if (command.substr(0, 5) == "copy ")
49         {
50             size_t pos = command.find(' ', 5);
51             std::string src = command.substr(5, pos - 5);
52             std::string dst = command.substr(pos + 1);
53             copyFile(src.c_str(), dst.c_str());
54         }
55         else if (command.substr(0, 5) == "move ")
56         {
57             size_t pos = command.find(' ', 5);
58             std::string src = command.substr(5, pos - 5);
59             std::string dst = command.substr(pos + 1);
60             rename(src.c_str(), dst.c_str());
61         }
62     }
63 }
```

**Day 4:** Implement file search functionality within the file explorer.

**Code:**

```
#include <iostream>
```

```
#include <dirent.h>
```

```
#include <unistd.h>
```

```
#include <cstring>
```

```
#include <fstream>
```

```
#include <stack>
```

```
void listFiles(const char *path)
```

```
{
    DIR *dir = opendir(path);
    if (dir)
    {
        struct dirent *entry;
        while ((entry = readdir(dir)))
        {
            std::cout << entry->d_name << std::endl;
        }
        closedir(dir);
    }
}
```

```
void copyFile(const char *src, const char *dst)
```

```
{
    std::ifstream in(src, std::ios::binary);
    std::ofstream out(dst, std::ios::binary);
    out << in.rdbuf();
}
```

```

void searchFiles(const char *root, const char *name)
{
    std::stack<std::string> dirs;
    dirs.push(root);
    while (!dirs.empty())
    {
        std::string current = dirs.top();
        dirs.pop();
        DIR *dir = opendir(current.c_str());
        if (dir)
        {
            struct dirent *entry;
            while ((entry = readdir(dir)))
            {
                if (strcmp(entry->d_name, ".") == 0 || strcmp(entry->d_name, "..") == 0)
                    continue;

                std::string fullPath = current + "/" + entry->d_name;
                if (strcmp(entry->d_name, name) == 0)
                {
                    std::cout << "Found: " << fullPath << std::endl;
                }

                if (entry->d_type == DT_DIR)
                {
                    dirs.push(fullPath);
                }
            }
            closedir(dir);
        }
    }
}

```

```

int main()
{
    char currentPath[1024] = ".";

    std::string command;

    while (true)
    {
        std::cout << "Current Directory: " << currentPath << std::endl;

        listFiles(currentPath);

        std::cout << "Command (cd <dir>, copy <src> <dst>, move <src> <dst>, delete <file>, create <file>,  
search <name>, or exit): ";

        std::getline(std::cin, command);

        if (command == "exit")
            break;

        if (command.substr(0, 3) == "cd ")
        {
            std::string newDir = command.substr(3);

            if (chdir(newDir.c_str()) == 0)
            {
                getcwd(currentPath, sizeof(currentPath));
            }
        }

        else if (command.substr(0, 5) == "copy ")
        {
            size_t pos = command.find(' ', 5);

            std::string src = command.substr(5, pos - 5);

            std::string dst = command.substr(pos + 1);

            copyFile(src.c_str(), dst.c_str());
        }

        else if (command.substr(0, 5) == "move ")
        {

```

```

    size_t pos = command.find(' ', 5);

    std::string src = command.substr(5, pos - 5);
    std::string dst = command.substr(pos + 1);
    rename(src.c_str(), dst.c_str());
}

else if (command.substr(0, 7) == "delete ")
{
    std::string file = command.substr(7);
    remove(file.c_str());
}

else if (command.substr(0, 7) == "create ")
{
    std::string file = command.substr(7);
    std::ofstream out(file);
}

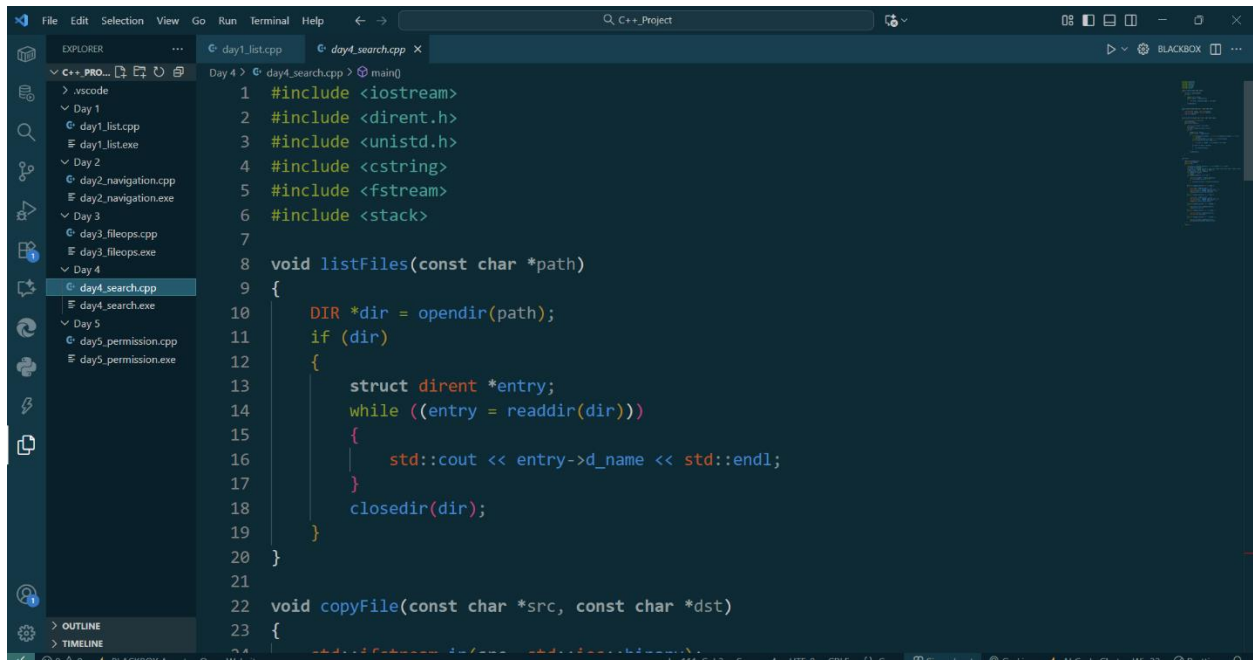
else if (command.substr(0, 7) == "search ")
{
    std::string name = command.substr(7);
    searchFiles(currentPath, name.c_str());
}

}

return 0;
}

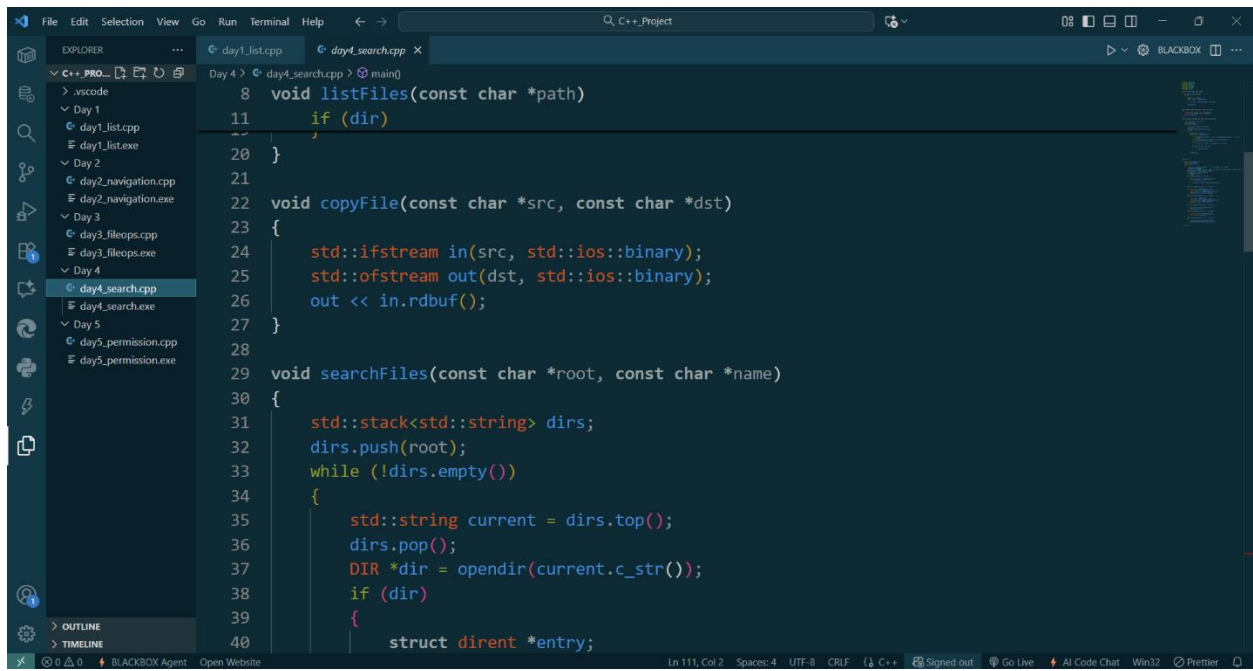
```

## Screenshots:



This screenshot shows the Visual Studio Code editor with the file explorer on the left displaying a project structure for 'C++\_PROJ'. The file explorer shows a hierarchy of folders (Day 1, Day 2, Day 3, Day 4, Day 5) and their respective source files and executables. The main editor window displays the code for 'day4\_search.cpp'. The code includes headers for `<iostream>`, `<dirent.h>`, `<unistd.h>`, `<cstring>`, `<fstream>`, and `<stack>`. It defines two functions: `listFiles` and `copyFile`. The `listFiles` function uses `opendir` to open a directory, `readdir` to iterate through its entries, and `closedir` to close it. The `copyFile` function is currently empty.

```
1 #include <iostream>
2 #include <dirent.h>
3 #include <unistd.h>
4 #include <cstring>
5 #include <fstream>
6 #include <stack>
7
8 void listFiles(const char *path)
9 {
10     DIR *dir = opendir(path);
11     if (dir)
12     {
13         struct dirent *entry;
14         while ((entry = readdir(dir)))
15         {
16             std::cout << entry->d_name << std::endl;
17         }
18         closedir(dir);
19     }
20 }
21
22 void copyFile(const char *src, const char *dst)
23 {
24 }
```



This screenshot shows the Visual Studio Code editor with the file explorer on the left displaying the same project structure as the first screenshot. The main editor window displays the code for 'day4\_search.cpp'. The code includes the same headers as the first screenshot. It defines three functions: `listFiles`, `copyFile`, and `searchFiles`. The `listFiles` function is now empty. The `copyFile` function uses `ifstream` to read from the source file and `ofstream` to write to the destination file. The `searchFiles` function uses a `stack` to store directories, `push` to add the root directory, and `while` to iterate through the stack, using `top` to get the current directory, `pop` to remove it, and `opendir` to open it. It also uses `readdir` to iterate through the entries of the current directory.

```
8 void listFiles(const char *path)
9 {
10 }
11
12 void copyFile(const char *src, const char *dst)
13 {
14     std::ifstream in(src, std::ios::binary);
15     std::ofstream out(dst, std::ios::binary);
16     out << in.rdbuf();
17 }
18
19 void searchFiles(const char *root, const char *name)
20 {
21     std::stack<std::string> dirs;
22     dirs.push(root);
23     while (!dirs.empty())
24     {
25         std::string current = dirs.top();
26         dirs.pop();
27         DIR *dir = opendir(current.c_str());
28         if (dir)
29         {
30             struct dirent *entry;
31             while ((entry = readdir(dir)))
32             {
33                 if (strcmp(entry->d_name, ".") != 0 && strcmp(entry->d_name, "..") != 0)
34                 {
35                     std::string new_dir = current + "/" + entry->d_name;
36                     dirs.push(new_dir.c_str());
37                 }
38             }
39         }
40     }
41 }
```

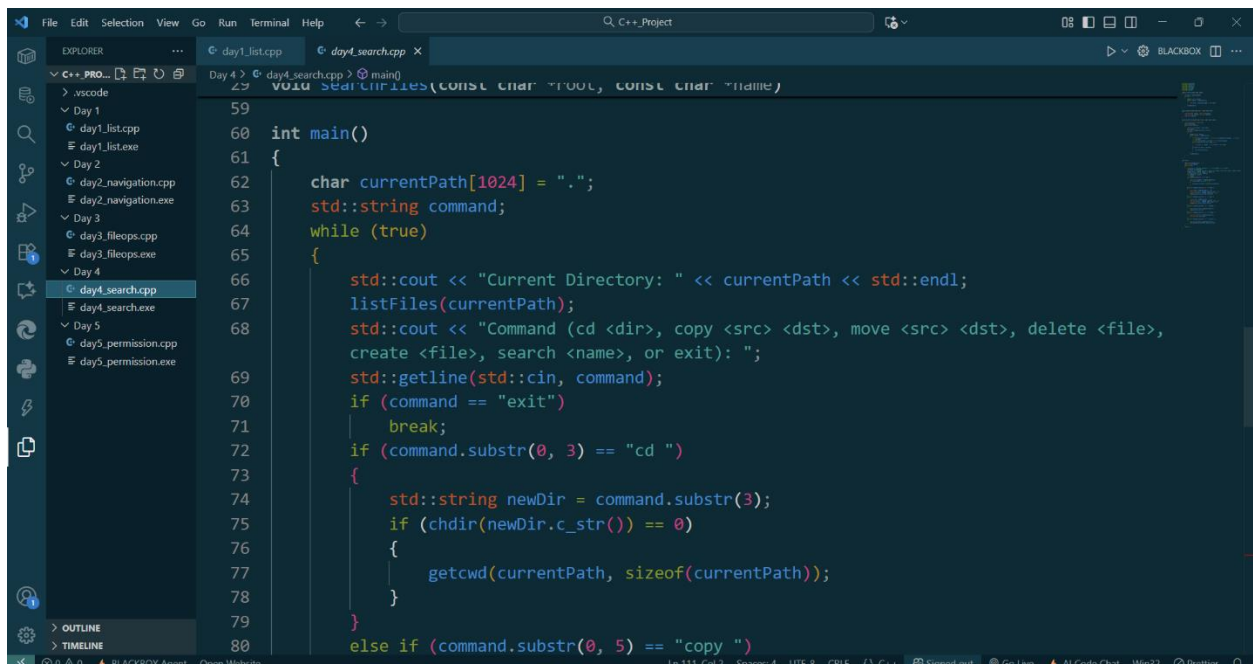
This screenshot shows the VS Code editor with the file explorer on the left displaying a project structure with files from Day 1 to Day 5. The main editor window shows the code for `day4_search.cpp`. The code includes a `copyFile` function and a `searchFiles` function. The `searchFiles` function uses a stack to traverse directories recursively, printing the full path of any directory found that matches the search name.

```
22 void copyFile(const char *src, const char *dst)
28
29 void searchFiles(const char *root, const char *name)
30 {
31     std::stack<std::string> dirs;
32     dirs.push(root);
33     while (!dirs.empty())
34     {
35         std::string current = dirs.top();
36         dirs.pop();
37         DIR *dir = opendir(current.c_str());
38         if (dir)
39         {
40             struct dirent *entry;
41             while ((entry = readdir(dir)))
42             {
43                 if (strcmp(entry->d_name, ".") == 0 || strcmp(entry->d_name, "..") == 0)
44                     continue;
45                 std::string fullPath = current + "/" + entry->d_name;
46                 if (strcmp(entry->d_name, name) == 0)
47                 {
48                     std::cout << "Found: " << fullPath << std::endl;
49                 }
50             }
51             dirs.push(fullPath);
52         }
53     }
54     closedir(dir);
55 }
```

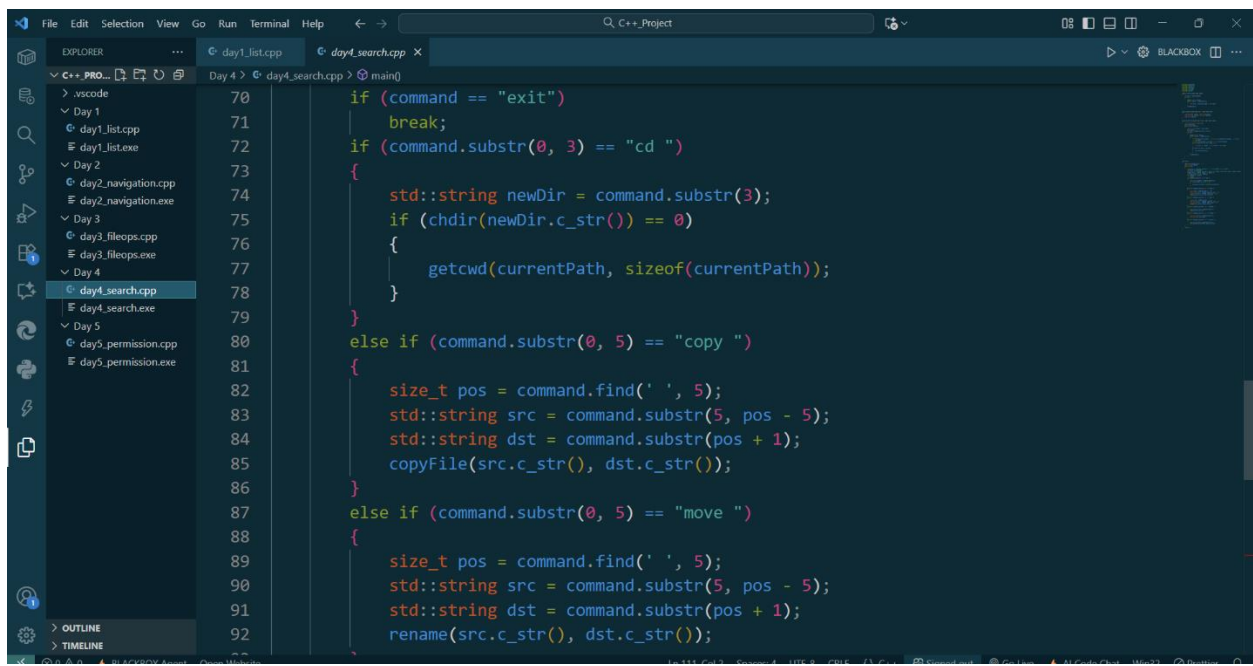
This screenshot shows the continuation of the `day4_search.cpp` file, specifically the `main` function. It initializes a `currentPath` array and calls the `searchFiles` function to search for a directory named `testdir` starting from the current directory.

```
29 void searchFiles(const char *root, const char *name)
33     while (!dirs.empty())
34     {
35         if (dir)
36         {
37             while ((entry = readdir(dir)))
38             {
39                 continue;
40                 std::string fullPath = current + "/" + entry->d_name;
41                 if (strcmp(entry->d_name, name) == 0)
42                 {
43                     std::cout << "Found: " << fullPath << std::endl;
44                 }
45                 if (entry->d_type == DT_DIR)
46                 {
47                     dirs.push(fullPath);
48                 }
49             }
50             closedir(dir);
51         }
52     }
53 }
54
55 int main()
56 {
57     char currentPath[1024] = ".";
58     std::string command;
```





```
File Edit Selection View Go Run Terminal Help C++_Project
EXPLORER C++_PRO... day1_list.cpp day4_search.cpp
> .vscode
  Day 1
    day1_list.cpp
    day1_list.exe
  Day 2
    day2_navigation.cpp
    day2_navigation.exe
  Day 3
    day3_fileops.cpp
    day3_fileops.exe
  Day 4
    day4_search.cpp
    day4_search.exe
  Day 5
    day5_permission.cpp
    day5_permission.exe
  OUTLINE
  TIMELINE
Day 4 > C: day4_search.cpp > main()
29 void listFiles(const char *root, const char *filename)
59
60 int main()
61 {
62     char currentPath[1024] = ".";
63     std::string command;
64     while (true)
65     {
66         std::cout << "Current Directory: " << currentPath << std::endl;
67         listFiles(currentPath);
68         std::cout << "Command (cd <dir>, copy <src> <dst>, move <src> <dst>, delete <file>,
        create <file>, search <name>, or exit): ";
69         std::getline(std::cin, command);
70         if (command == "exit")
71             break;
72         if (command.substr(0, 3) == "cd ")
73         {
74             std::string newDir = command.substr(3);
75             if (chdir(newDir.c_str()) == 0)
76             {
77                 getcwd(currentPath, sizeof(currentPath));
78             }
79         }
80         else if (command.substr(0, 5) == "copy ")
```



```
File Edit Selection View Go Run Terminal Help C++_Project
EXPLORER C++_PRO... day1_list.cpp day4_search.cpp
> .vscode
  Day 1
    day1_list.cpp
    day1_list.exe
  Day 2
    day2_navigation.cpp
    day2_navigation.exe
  Day 3
    day3_fileops.cpp
    day3_fileops.exe
  Day 4
    day4_search.cpp
    day4_search.exe
  Day 5
    day5_permission.cpp
    day5_permission.exe
  OUTLINE
  TIMELINE
Day 4 > C: day4_search.cpp > main()
70         if (command == "exit")
71             break;
72         if (command.substr(0, 3) == "cd ")
73         {
74             std::string newDir = command.substr(3);
75             if (chdir(newDir.c_str()) == 0)
76             {
77                 getcwd(currentPath, sizeof(currentPath));
78             }
79         }
80         else if (command.substr(0, 5) == "copy ")
81         {
82             size_t pos = command.find(' ', 5);
83             std::string src = command.substr(5, pos - 5);
84             std::string dst = command.substr(pos + 1);
85             copyFile(src.c_str(), dst.c_str());
86         }
87         else if (command.substr(0, 5) == "move ")
88         {
89             size_t pos = command.find(' ', 5);
90             std::string src = command.substr(5, pos - 5);
91             std::string dst = command.substr(pos + 1);
92             rename(src.c_str(), dst.c_str());
93         }
```

This screenshot shows the implementation of the 'copy' command in the `day4_search.cpp` file. The code is part of a `main` function that processes a `command` string. It uses `std::string` and `std::ofstream` for file operations. The 'copy' command is implemented by finding the source file path, copying its contents to the destination file, and then removing the source file.

```
84:         std::string dst = command.substr(pos + 1);
85:         copyFile(src.c_str(), dst.c_str());
86:     }
87:     else if (command.substr(0, 5) == "move ")
88:     {
89:         size_t pos = command.find(' ', 5);
90:         std::string src = command.substr(5, pos - 5);
91:         std::string dst = command.substr(pos + 1);
92:         rename(src.c_str(), dst.c_str());
93:     }
94:     else if (command.substr(0, 7) == "delete ")
95:     {
96:         std::string file = command.substr(7);
97:         remove(file.c_str());
98:     }
99:     else if (command.substr(0, 7) == "create ")
100:    {
101:        std::string file = command.substr(7);
102:        std::ofstream out(file);
103:    }
104:    else if (command.substr(0, 7) == "search ")
105:    {
106:        std::string name = command.substr(7);
107:        searchFiles(currentPath, name.c_str());
108:    }
109: }
```

This screenshot shows the implementation of the 'delete' command in the `day4_search.cpp` file. The code is part of a `main` function that processes a `command` string. It uses `std::string` and `std::ofstream` for file operations. The 'delete' command is implemented by removing the file specified in the command.

```
94:     else if (command.substr(0, 7) == "delete ")
95:     {
96:         std::string file = command.substr(7);
97:         remove(file.c_str());
98:     }
99:     else if (command.substr(0, 7) == "create ")
100:    {
101:        std::string file = command.substr(7);
102:        std::ofstream out(file);
103:    }
104:    else if (command.substr(0, 7) == "search ")
105:    {
106:        std::string name = command.substr(7);
107:        searchFiles(currentPath, name.c_str());
108:    }
109: }
110: return 0;
111: }
```

## **Day 5:** Add file permission management features.

### **Code:**

```
#include <iostream>

#include <dirent.h>

#include <unistd.h>

#include <cstring>

#include <fstream>

#include <stack>

#include <sys/stat.h>
```

```
void listFiles(const char *path)
{
    DIR *dir = opendir(path);
    if (dir)
    {
        struct dirent *entry;
        while ((entry = readdir(dir)))
        {
            std::cout << entry->d_name << std::endl;
        }
        closedir(dir);
    }
}
```

```
void copyFile(const char *src, const char *dst)
{
    std::ifstream in(src, std::ios::binary);
    std::ofstream out(dst, std::ios::binary);
    out << in.rdbuf();
}
```

```

void searchFiles(const char *root, const char *name)
{
    std::stack<std::string> dirs;
    dirs.push(root);
    while (!dirs.empty ())
    {
        std::string current = dirs.top();
        dirs.pop();
        DIR *dir = opendir(current.c_str());
        if (dir)
        {
            struct dirent *entry;
            while ((entry = readdir(dir)))
            {
                if (strcmp(entry->d_name, ".") == 0 || strcmp(entry->d_name, "..") == 0)
                    continue;

                std::string fullPath = current + "/" + entry->d_name;
                if (strcmp(entry->d_name, name) == 0)
                {
                    std::cout << "Found: " << fullPath << std::endl;
                }
                if (entry->d_type == DT_DIR)
                {
                    dirs.push(fullPath);
                }
            }
            closedir(dir);
        }
    }
}

```

```
}
```

```
int main()
```

```
{
```

```
    char currentPath [1024] = ".";
```

```
    std::string command;
```

```
    while (true)
```

```
    {
```

```
        std::cout << "Current Directory: " << currentPath << std::endl;
```

```
        listFiles(currentPath);
```

```
        std::cout << "Command (cd <dir>, copy <src> <dst>, move <src> <dst>, delete <file>, create <file>,  
search <name>, permissions <file>, chmod <mode> <file>, or exit): ";
```

```
        std::getline(std::cin, command);
```

```
        if (command == "exit")
```

```
            break;
```

```
        if (command.substr(0, 3) == "cd ")
```

```
        {
```

```
            std::string newDir = command.substr(3);
```

```
            if (chdir(newDir.c_str()) == 0)
```

```
            {
```

```
                getcwd(currentPath, sizeof(currentPath));
```

```
            }
```

```
        }
```

```
        else if (command.substr(0, 5) == "copy ")
```

```
        {
```

```
            size_t pos = command.find(' ', 5);
```

```
            std::string src = command.substr(5, pos - 5);
```

```
            std::string dst = command.substr(pos + 1);
```

```
            copyFile(src.c_str(), dst.c_str());
```

```
        }
```

```
        else if (command.substr(0, 5) == "move ")
```

```

{
    size_t pos = command.find(' ', 5);
    std::string src = command.substr(5, pos - 5);
    std::string dst = command.substr(pos + 1);
    rename(src.c_str(), dst.c_str());
}
else if (command.substr(0, 7) == "delete ")
{
    std::string file = command.substr(7);
    remove(file.c_str());
}
else if (command.substr(0, 7) == "create ")
{
    std::string file = command.substr(7);
    std::ofstream out(file);
}
else if (command.substr(0, 7) == "search ")
{
    std::string name = command.substr(7);
    searchFiles(currentPath, name.c_str());
}
else if (command.substr(0, 12) == "permissions ")
{
    std::string file = command.substr(12);
    struct stat st;
    if (stat(file.c_str(), &st) == 0)
    {
        std::cout << "Permissions: " << std::oct << (st.st_mode & 0777) << std::dec << std::endl;
    }
}
}

```

```

else if (command.substr(0, 6) == "chmod ")
{
    size_t pos = command.find(' ', 6);

    std::string modeStr = command.substr(6, pos - 6);

    std::string file = command.substr(pos + 1);

    int mode = strtol (modeStr.c_str(), nullptr, 8);

    chmod(file.c_str(), mode);

}

}

return 0;

}

```

## Screenshot:

```

File Edit Selection View Go Run Terminal Help
C++_Project
EXPLORER
C++_PROJ...
  .vscode
  Day 1
    day1_list.cpp
    day1_list.exe
  Day 2
    day2_navigation.cpp
    day2_navigation.exe
  Day 3
    day3_fileops.cpp
    day3_fileops.exe
  Day 4
    day4_search.cpp
    day4_search.exe
  Day 5
    day5_permission.cpp
    day5_permission.exe
OUTLINE
TIMELINE
Ln 127, Col 6

```

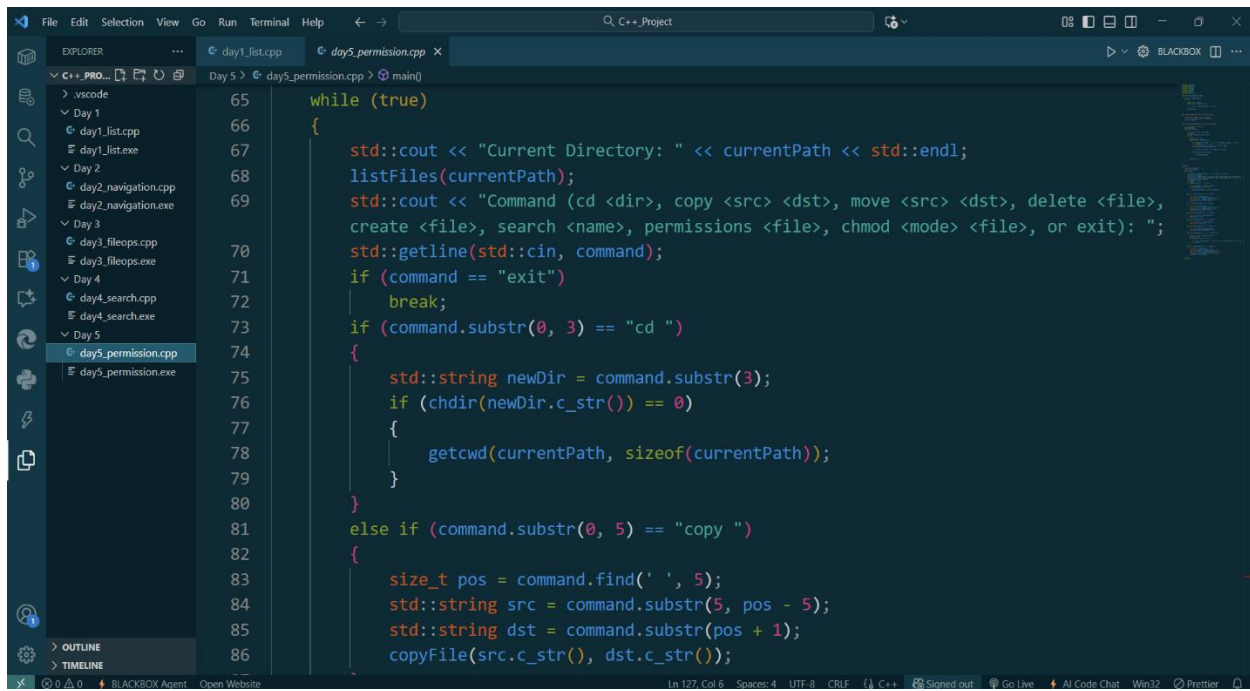
This screenshot shows the VS Code editor with the file explorer on the left displaying a project structure with folders for Day 1 through Day 5. The main editor window shows the code for day5\_permission.cpp. The code includes a copyFile function and a searchFiles function. The copyFile function uses std::ifstream and std::ofstream to copy a file. The searchFiles function uses a stack to traverse a directory tree, printing the full path of each file and directory.

```
23 void copyFile(const char *src, const char *dst)
24 {
25     std::ifstream in(src, std::ios::binary);
26     std::ofstream out(dst, std::ios::binary);
27     out << in.rdbuf();
28 }
29
30 void searchFiles(const char *root, const char *name)
31 {
32     std::stack<std::string> dirs;
33     dirs.push(root);
34     while (!dirs.empty())
35     {
36         std::string current = dirs.top();
37         dirs.pop();
38         DIR *dir = opendir(current.c_str());
39         if (dir)
40         {
41             struct dirent *entry;
42             while ((entry = readdir(dir)))
43             {
44                 if (strcmp(entry->d_name, ".") == 0 || strcmp(entry->d_name, "..") == 0)
45                     continue;
```

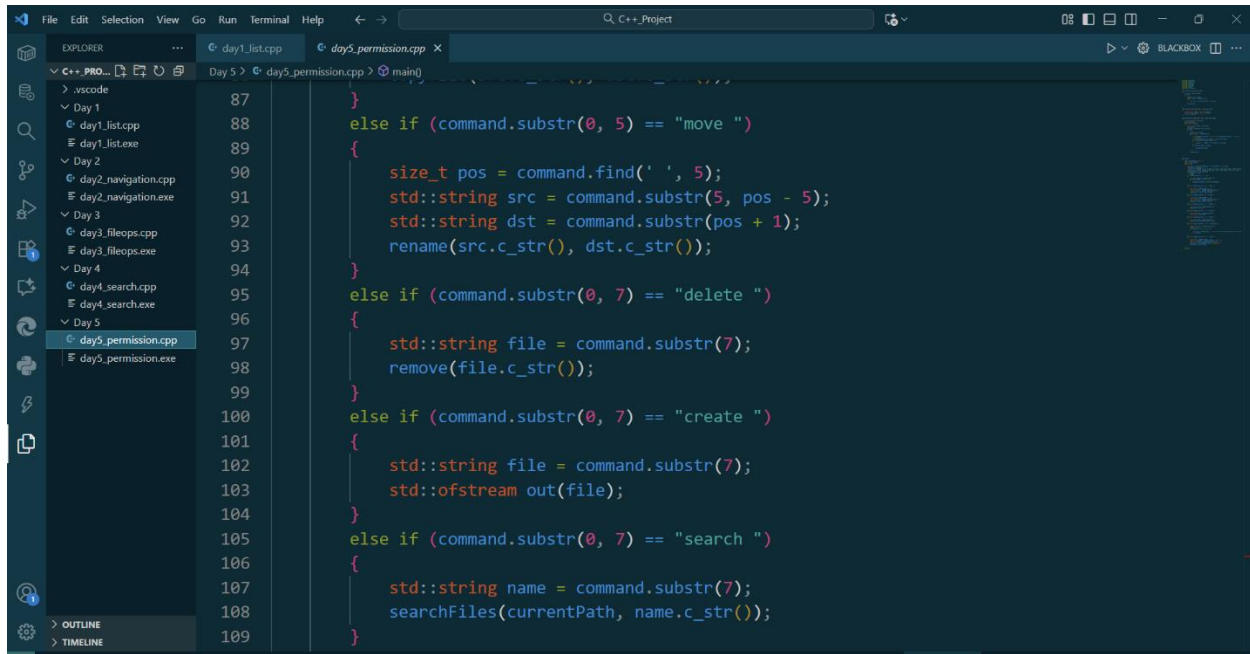
This screenshot shows the continuation of the searchFiles function and the main function in day5\_permission.cpp. The searchFiles function continues with a while loop that iterates over the directory entries, printing the full path of each file and directory. The main function is also shown, which calls searchFiles to search for a file named "command".

```
46         std::string fullPath = current + "/" + entry->d_name;
47         if (strcmp(entry->d_name, name) == 0)
48         {
49             std::cout << "Found: " << fullPath << std::endl;
50         }
51         if (entry->d_type == DT_DIR)
52         {
53             dirs.push(fullPath);
54         }
55     }
56     closedir(dir);
57 }
58 }
59
60
61 int main()
62 {
63     char currentPath[1024] = ".";
64     std::string command;
```





```
65 while (true)
66 {
67     std::cout << "Current Directory: " << currentPath << std::endl;
68     listFiles(currentPath);
69     std::cout << "Command (cd <dir>, copy <src> <dst>, move <src> <dst>, delete <file>,
70     create <file>, search <name>, permissions <file>, chmod <mode> <file>, or exit): ";
71     std::getline(std::cin, command);
72     if (command == "exit")
73         break;
74     if (command.substr(0, 3) == "cd ")
75     {
76         std::string newDir = command.substr(3);
77         if (chdir(newDir.c_str()) == 0)
78         {
79             getcwd(currentPath, sizeof(currentPath));
80         }
81     }
82     else if (command.substr(0, 5) == "copy ")
83     {
84         size_t pos = command.find(' ', 5);
85         std::string src = command.substr(5, pos - 5);
86         std::string dst = command.substr(pos + 1);
87         copyFile(src.c_str(), dst.c_str());
```



```
87 }
88 else if (command.substr(0, 5) == "move ")
89 {
90     size_t pos = command.find(' ', 5);
91     std::string src = command.substr(5, pos - 5);
92     std::string dst = command.substr(pos + 1);
93     rename(src.c_str(), dst.c_str());
94 }
95 else if (command.substr(0, 7) == "delete ")
96 {
97     std::string file = command.substr(7);
98     remove(file.c_str());
99 }
100 else if (command.substr(0, 7) == "create ")
101 {
102     std::string file = command.substr(7);
103     std::ofstream out(file);
104 }
105 else if (command.substr(0, 7) == "search ")
106 {
107     std::string name = command.substr(7);
108     searchFiles(currentPath, name.c_str());
109 }
```

```
108     searchFiles(currentPath, name.c_str());
109 }
110 else if (command.substr(0, 12) == "permissions ")
111 {
112     std::string file = command.substr(12);
113     struct stat st;
114     if (stat(file.c_str(), &st) == 0)
115     {
116         std::cout << "Permissions: " << std::oct << (st.st_mode & 0777) << std::dec
117         << std::endl;
118     }
119 }
120 else if (command.substr(0, 6) == "chmod ")
121 {
122     size_t pos = command.find(' ', 6);
123     std::string modeStr = command.substr(6, pos - 6);
124     std::string file = command.substr(pos + 1);
125     int mode = strtol(modeStr.c_str(), nullptr, 8);
126     chmod(file.c_str(), mode);
127 }
128 return 0;
129 }
```