

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 = ".";

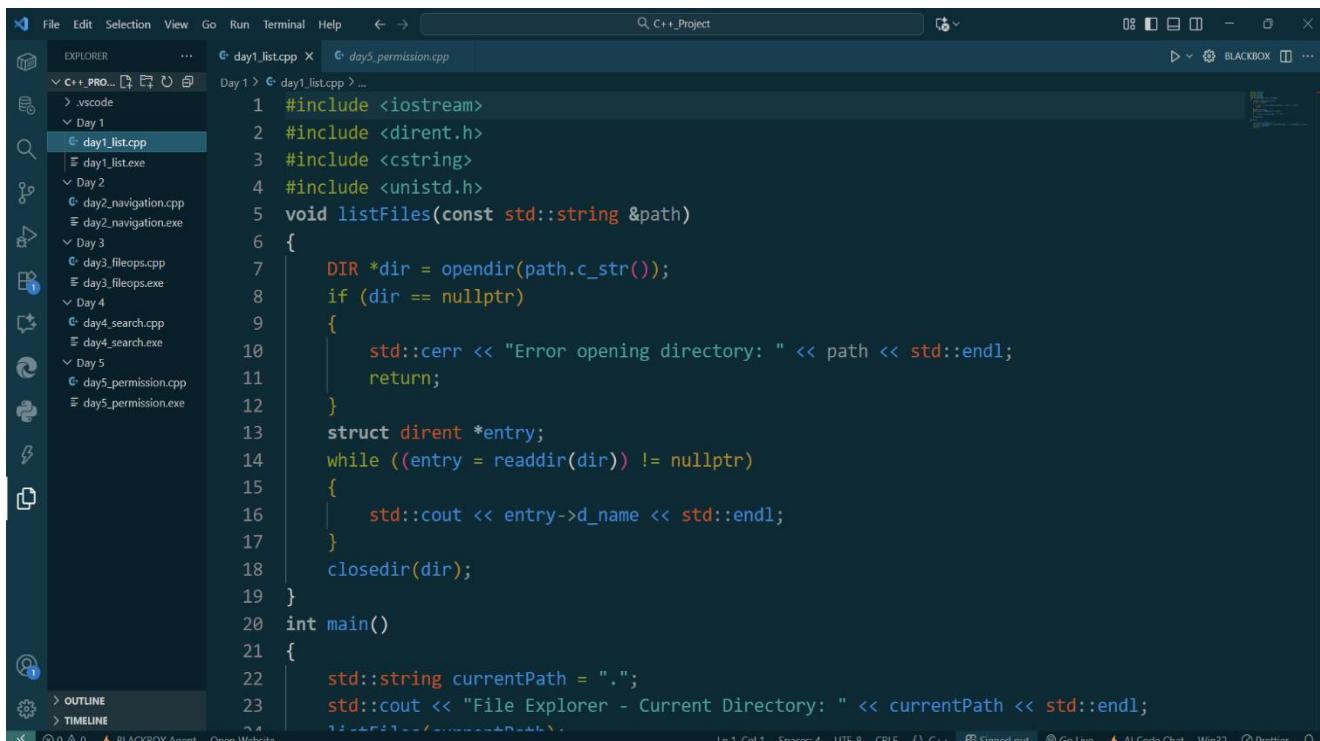
    std::cout << "File Explorer - Current Directory: " << currentPath << std::endl;

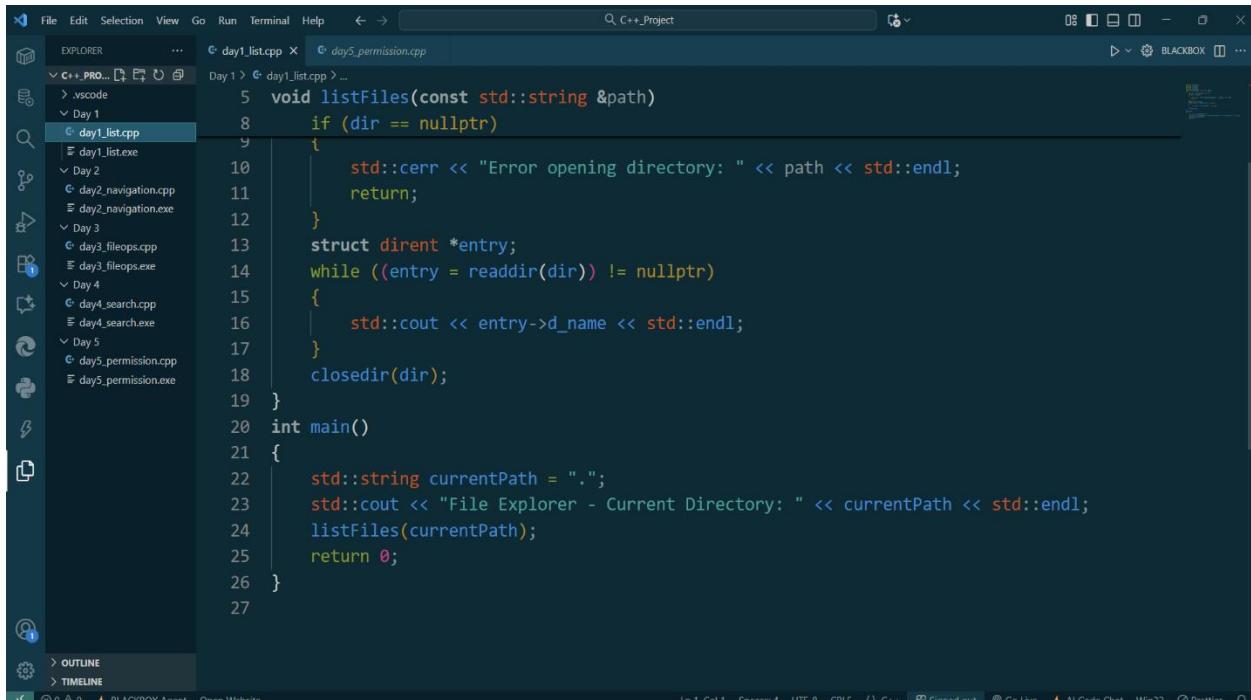
    listFiles(currentPath);

    return 0;
}

```

Screenshots:





```
File Edit Selection View Go Run Terminal Help < → C++_Project Day 1 > day1_list.cpp ... Day 1 > day1_list.cpp > ... 5 void listFiles(const std::string &path) 8 if (dir == nullptr) 9 { 10     std::cerr << "Error opening directory: " << path << std::endl; 11     return; 12 } 13 struct dirent *entry; 14 while ((entry = readdir(dir)) != nullptr) 15 { 16     std::cout << entry->d_name << std::endl; 17 } 18 closedir(dir); 19 } 20 int main() 21 { 22     std::string currentPath = "."; 23     std::cout << "File Explorer - Current Directory: " << currentPath << std::endl; 24     listFiles(currentPath); 25 } 26 } 27 }
```

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;
            }
        }
    }
}
```

```

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

}

return 0;
}

```

Screenshot:

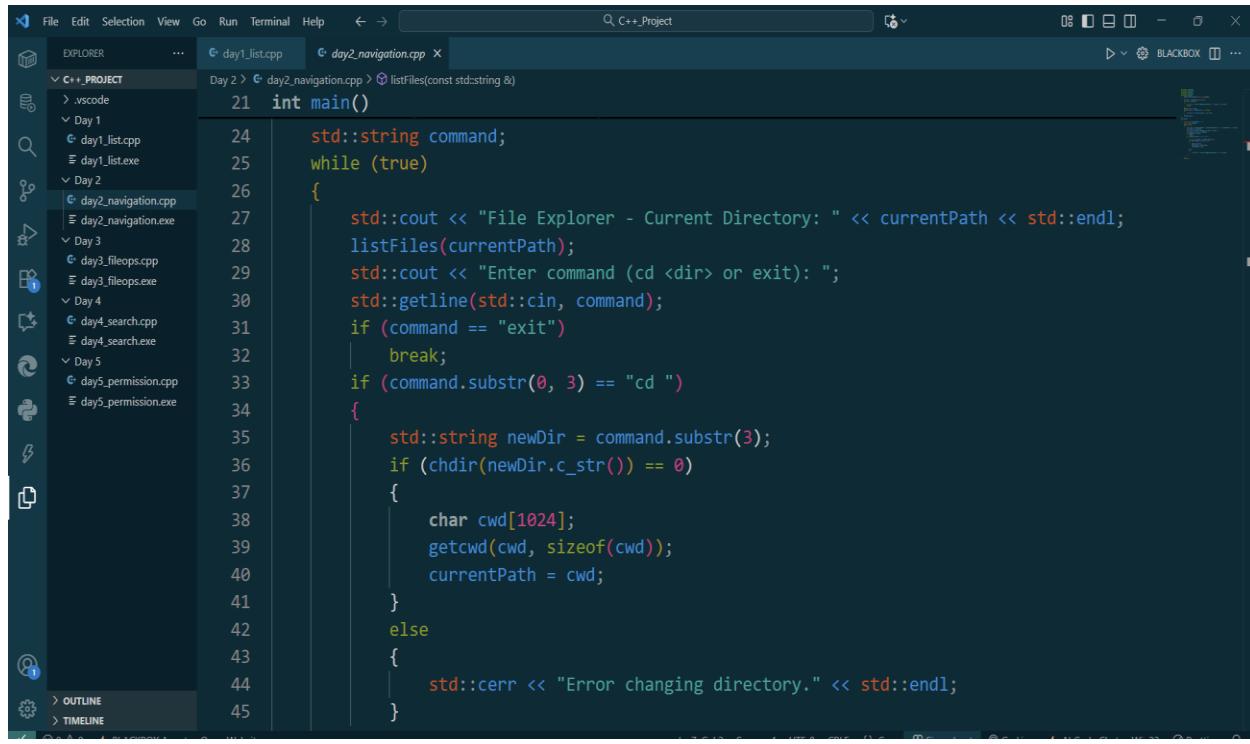
The screenshot shows a C++ development environment with the following details:

- File Explorer:** Shows a project structure under "C++_PROJECT" named "Day 2". It includes files like day1_list.cpp, day2_navigation.cpp, day1_list.exe, day2_navigation.exe, day3_fileops.cpp, day3_fileops.exe, day4_search.cpp, day4_search.exe, day5_permission.cpp, and day5_permission.exe.
- Code Editor:** The active file is "day2_navigation.cpp". The code implements a function to list files in a directory using opendir, readdir, and closedir.
- Status Bar:** Shows the current file is "day2_navigation.cpp", line 7, column 2, with 17 spaces and 8 tabs. It also shows "CRLF" encoding, "Signed out", "Go Live", "AI Code Chat", "Win32", and "Prettier" icons.

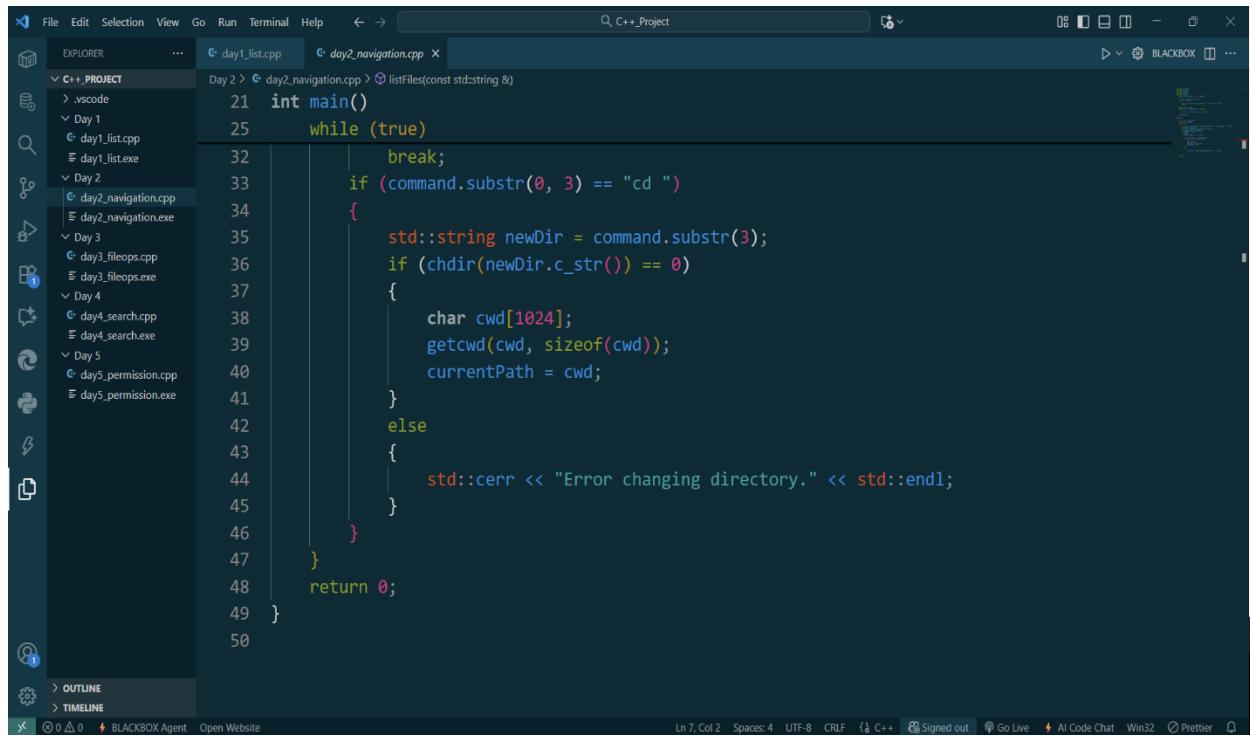
```

1 #include <iostream>
2 #include <dirent.h>
3 #include <cstring>
4 #include <unistd.h>
5 #include <limits>
6 void listFiles(const std::string &path)
7 {
8     DIR *dir = opendir(path.c_str());
9     if (dir == nullptr)
10     {
11         std::cerr << "Error opening directory: " << path << std::endl;
12         return;
13     }
14     struct dirent *entry;
15     while ((entry = readdir(dir)) != nullptr)
16     {
17         std::cout << entry->d_name << std::endl;
18     }
19     closedir(dir);
20 }
21 int main()
22 {
23     std::string currentPath = ".";

```



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

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:

The screenshot shows the Visual Studio Code interface with a dark theme. The left sidebar displays a file tree for a 'C++_Project' folder. The 'EXPLORER' view shows files like 'day1_list.cpp', 'day3_fileops.cpp', and executables 'day1_list.exe', 'day3_fileops.exe', etc. The 'CODE' view shows the source code for 'day3_fileops.cpp'. The code implements two functions: 'listFiles' which lists directory contents using opendir, readdir, and closedir, and 'copyFile' which copies files using ifstream and ofstream. The status bar at the bottom shows various open tabs and settings.

```

1 #include <iostream>
2 #include <dirent.h>
3 #include <unistd.h>
4 #include <cstring>
5 #include <fstream>
6
7 void listFiles(const char *path)
8 {
9     DIR *dir = opendir(path);
10    if (dir)
11    {
12        struct dirent *entry;
13        while ((entry = readdir(dir)))
14        {
15            std::cout << entry->d_name << std::endl;
16        }
17        closedir(dir);
18    }
19 }
20
21 void copyFile(const char *src, const char *dst)
22 {
23     std::ifstream in(src, std::ios::binary);
24
25     std::ofstream out(dst, std::ios::binary);
26
27     if (!in || !out)
28     {
29         std::cerr << "Error opening file" << std::endl;
30         return -1;
31     }
32
33     std::copy(in.begin(), in.end(), out.begin());
34
35     in.close();
36     out.close();
37 }
38
39 int main()
40 {
41     listFiles("Day 3");
42     copyFile("Day 3/day3_fileops.exe", "Day 3/day3_copy.exe");
43
44     return 0;
45 }

```

```
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)
                break;
        }
        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());
        }
    }
}
```

```
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());
}

listFiles(currentPath);
```

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:

A screenshot of the Visual Studio Code interface. The title bar says "C++_Project". The left sidebar shows a project structure under "EXPLORER" with a tree view of files and folders. The main editor area displays the code for "day4_search.cpp". The code implements a search function that lists files in a directory and copies files from one location to another.

```
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 }
```

A screenshot of the Visual Studio Code interface, similar to the first one but with less code visible in the editor. The title bar says "C++_Project". The left sidebar shows the project structure. The main editor area displays the code for "day4_search.cpp", which includes functions for listing files, copying files, and searching for files.

```
8 void listfiles(const char *path)
11     if (dir)
20 }
21
22 void copyFile(const char *src, const char *dst)
23 {
24     std::ifstream in(src, std::ios::binary);
25     std::ofstream out(dst, std::ios::binary);
26     out << in.rdbuf();
27 }
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;
```

The screenshot shows a C++ IDE interface with the following details:

- File Bar:** File, Edit, Selection, View, Go, Run, Terminal, Help.
- Toolbar:** Standard icons for file operations.
- Explorer:** Shows a project structure under "C++ PROJ...".
- Search Bar:** Day 4 > day4.search.cpp > main()
- Code Editor:** Displays the following C++ code:

```
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             closedir(dir);
52         }
53     }
54 }
55
56 }
57
58 int main()
59 {
60     char currentPath[1024] = ".";
61     std::string command;
```

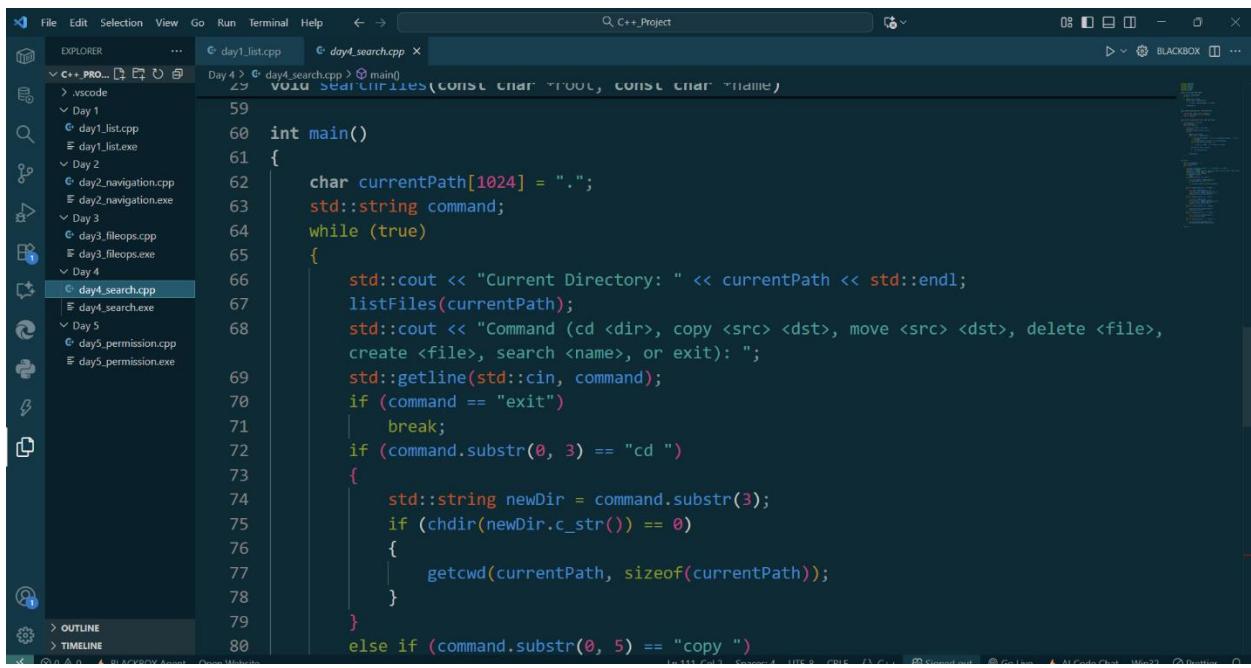
Bottom status bar: Ln 111, Col 2, Spaces: 4, UTF-8, CRLF, Signed out, Go Live, AI Code Chat, Win32, Prettier.

The screenshot shows a C++ IDE interface with the following details:

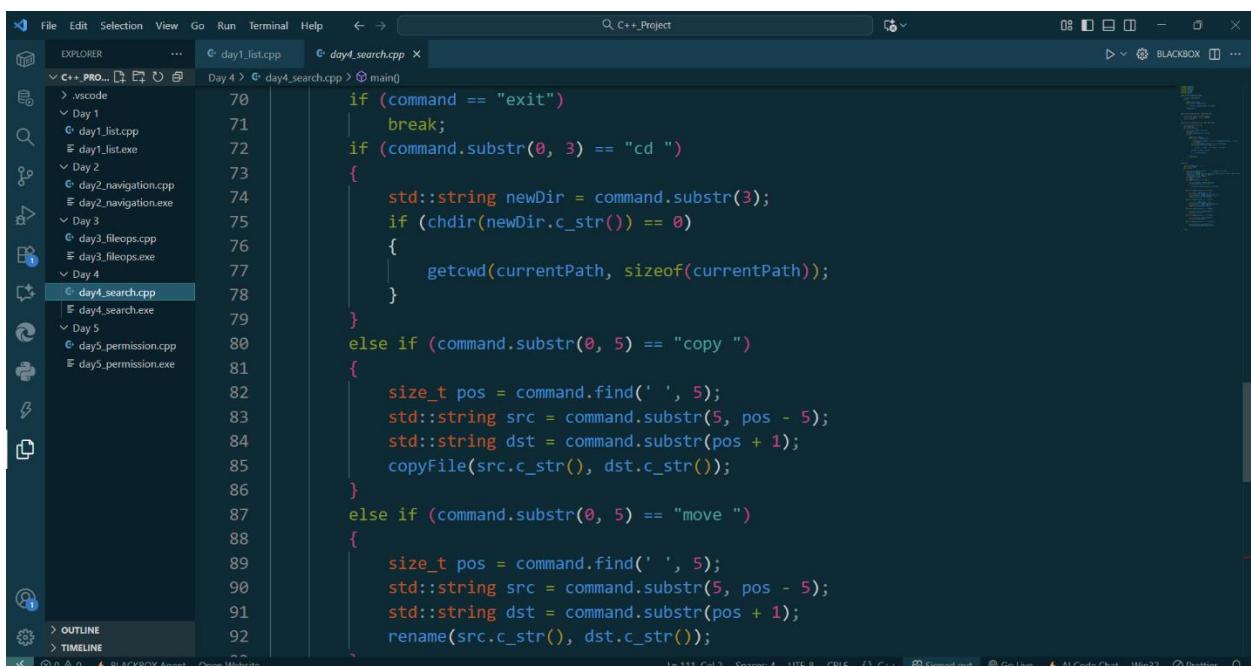
- File Bar:** File, Edit, Selection, View, Go, Run, Terminal, Help.
- Toolbar:** Standard icons for file operations.
- Explorer:** Shows a project structure under "C++ PROJ...".
- Search Bar:** Day 4 > day4.search.cpp > main()
- Code Editor:** Displays the following C++ code:

```
29 void searchFiles(const char *root, const char *name)
33     while (!dirs.empty())
38     if (dir)
41         while ((entry = readdir(dir)))
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             if (entry->d_type == DT_DIR)
51             {
52                 dirs.push(fullPath);
53             }
54         }
55     }
56 }
57
58 int main()
59 {
60     char currentPath[1024] = ".";
61     std::string command;
```

Bottom status bar: Ln 111, Col 2, Spaces: 4, UTF-8, CRLF, Signed out, Go Live, AI Code Chat, Win32, Prettier.



```
Day 4 > C:\Users\Day 4> day4_search.cpp > main()
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 ")
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         }
94     }
95 }
```



```
Day 4 > C:\Users\Day 4> day4_search.cpp > main()
70
71         if (command == "delete")
72             break;
73         if (command.substr(0, 5) == "search")
74         {
75             std::string name = command.substr(5);
76             searchFile(name);
77         }
78     }
79 }
```

```
Day 4 > G day4_search.cpp > main()
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 }
110 return 0;
111 }
```

```
Day 4 > G day4_search.cpp > main()
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 }
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;

    }

}

```

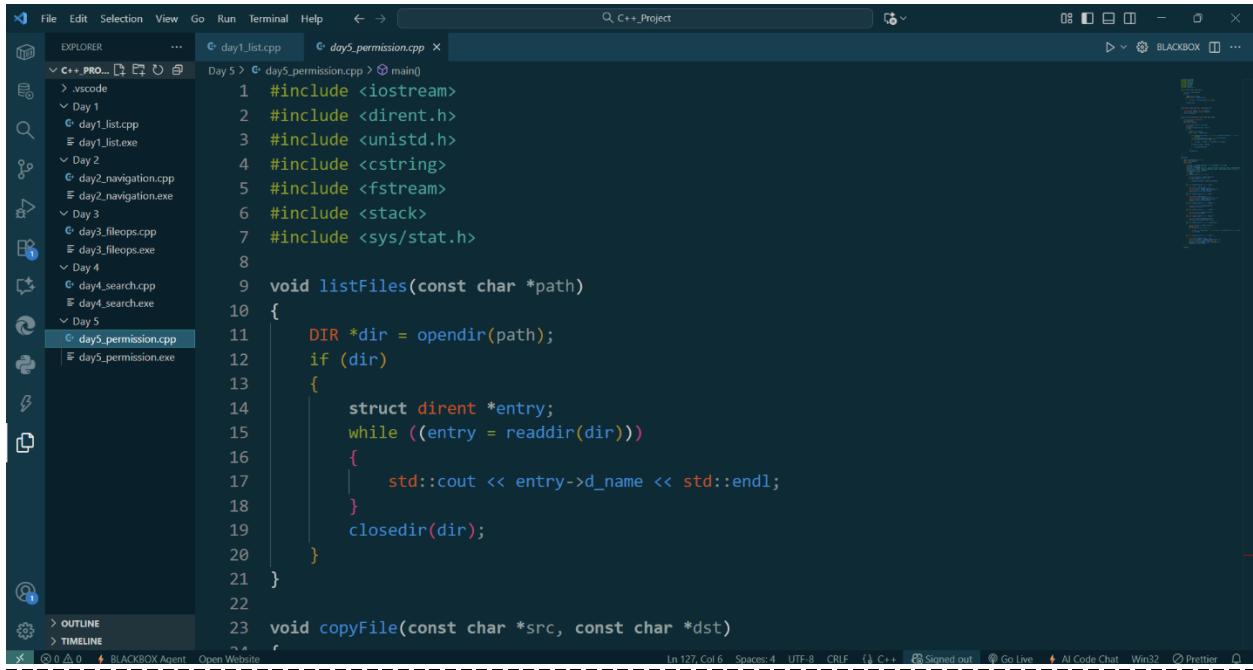
```

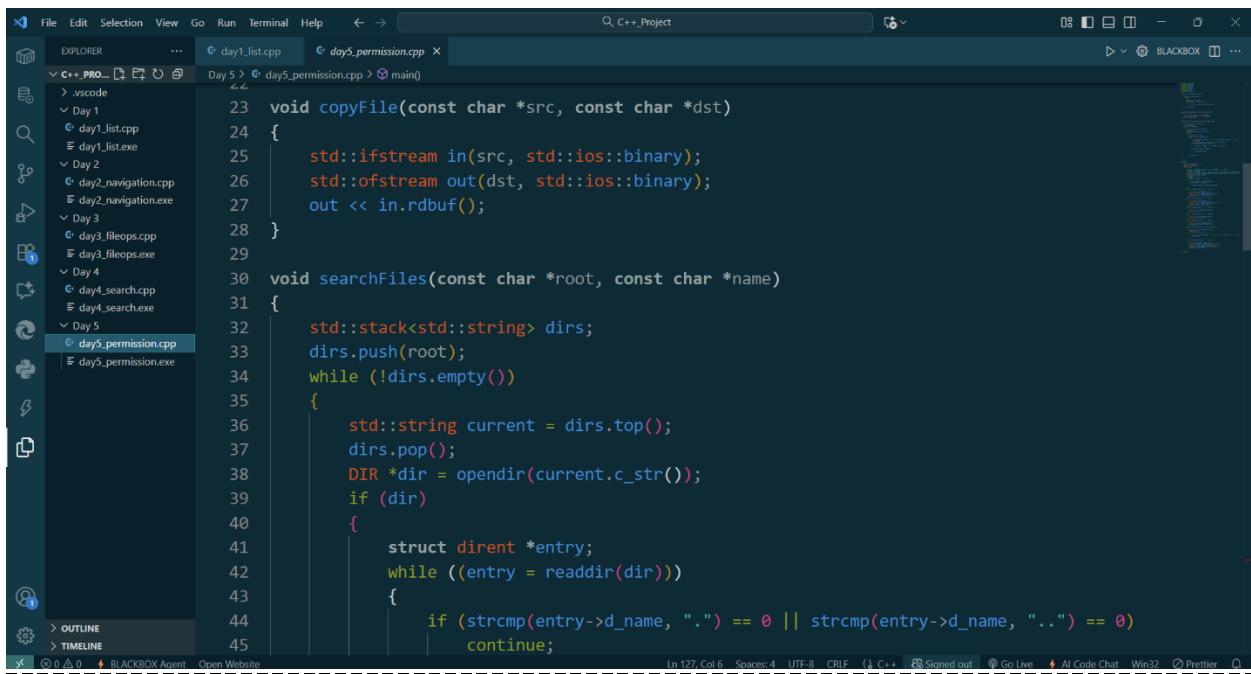
else if (command.substr(0, 6) == "chmod ")
{
    size_t pos = command.find(' ', 6);
    std::string modeStr = command.substr(6, pos - 6);
    std::string file = command.substr(pos + 1);
    int mode = strtol(modeStr.c_str(), nullptr, 8);
    chmod(file.c_str(), mode);
}

return 0;
}

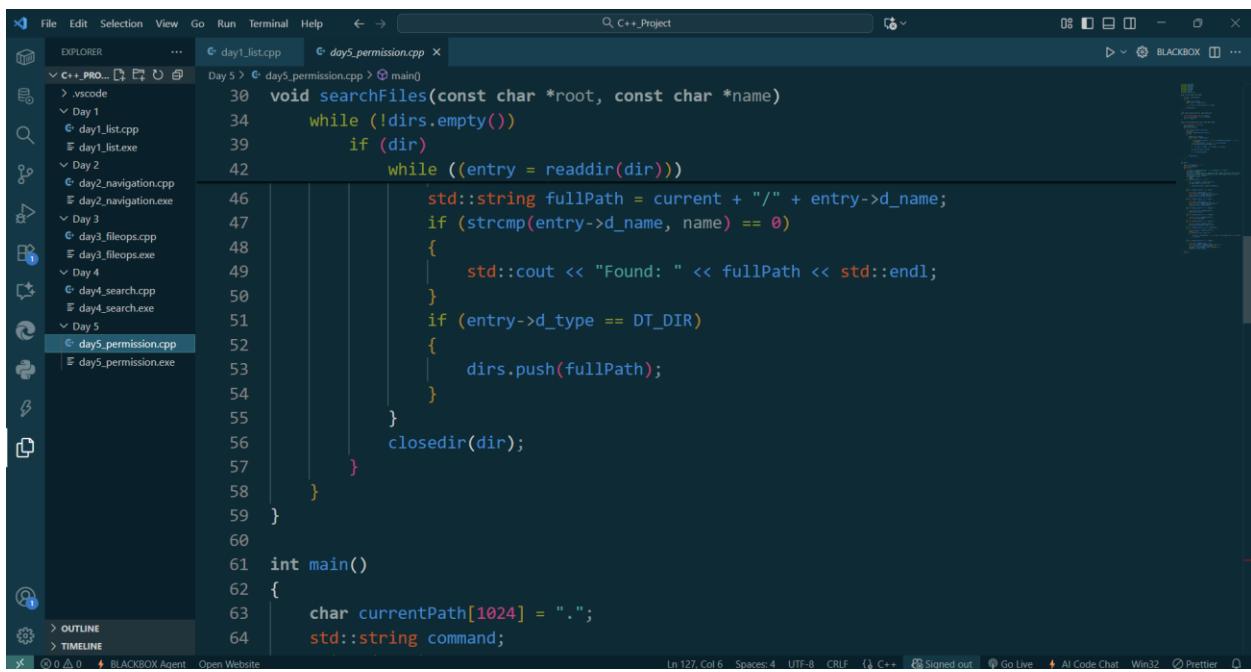
```

Screenshot:





```
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;
```



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

The screenshot shows a C++ development environment with the following details:

- File Menu:** File, Edit, Selection, View, Go, Run, Terminal, Help.
- Toolbar:** Standard window control buttons.
- Search Bar:** C++_Project
- Explorer:** Shows a project structure under "C++ PROJ...":
 - 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
- Code Editor:** The active file is "day5_permission.cpp". The code implements a command-line parser for file operations. It includes functions for searching files, changing permissions, and modifying file modes. The code uses standard C++ libraries like `std::string`, `std::oct`, and `std::endl`.
- Terminal:** Shows command-line history and output.
- Bottom Status Bar:** Includes icons for file operations like Open, Save, and Close, along with "BLACKBOX" and other status indicators.

```
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 else if (command.substr(0, 6) == "chmod ")
120 {
121     size_t pos = command.find(' ', 6);
122     std::string modeStr = command.substr(6, pos - 6);
123     std::string file = command.substr(pos + 1);
124     int mode = strtol(modeStr.c_str(), nullptr, 8);
125     chmod(file.c_str(), mode);
126 }
127 }
128 return 0;
129 }
```