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:

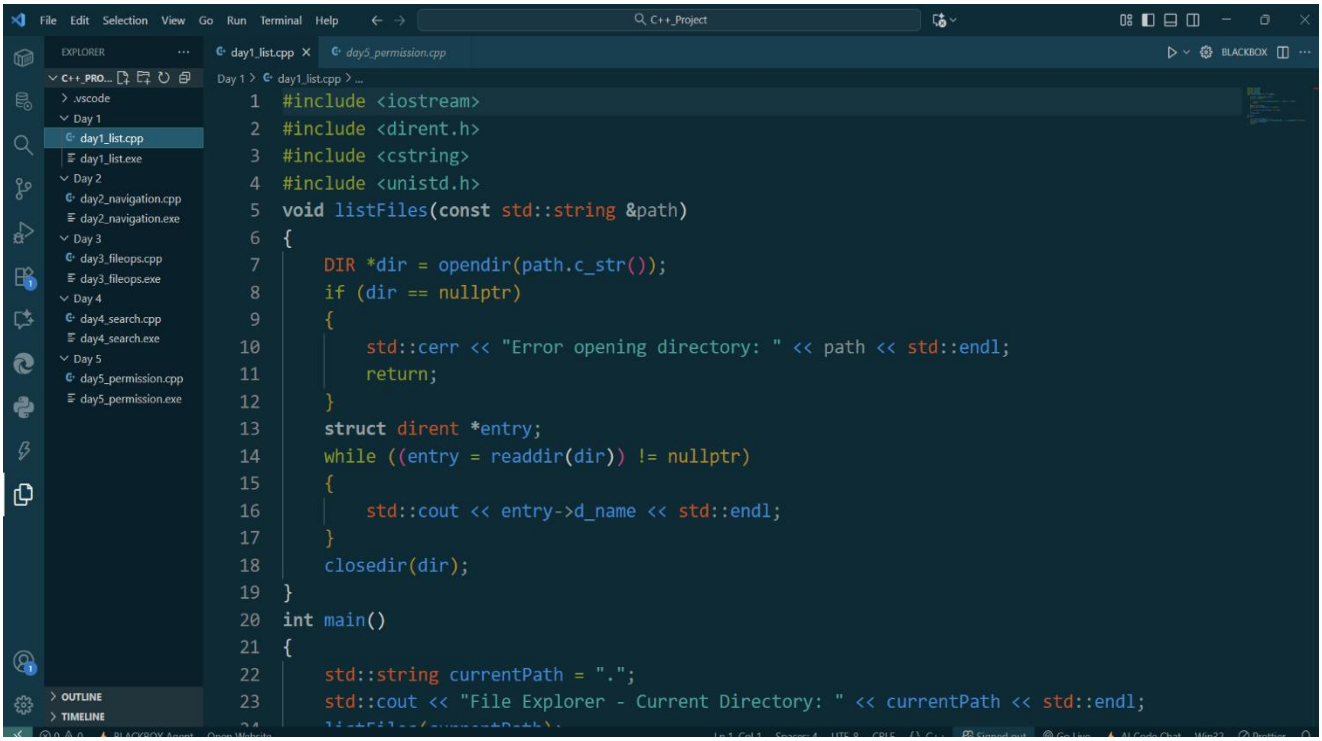
Day 1: Design the application structure and setup the development environment. Start with basic file operations like listing files in a directory.

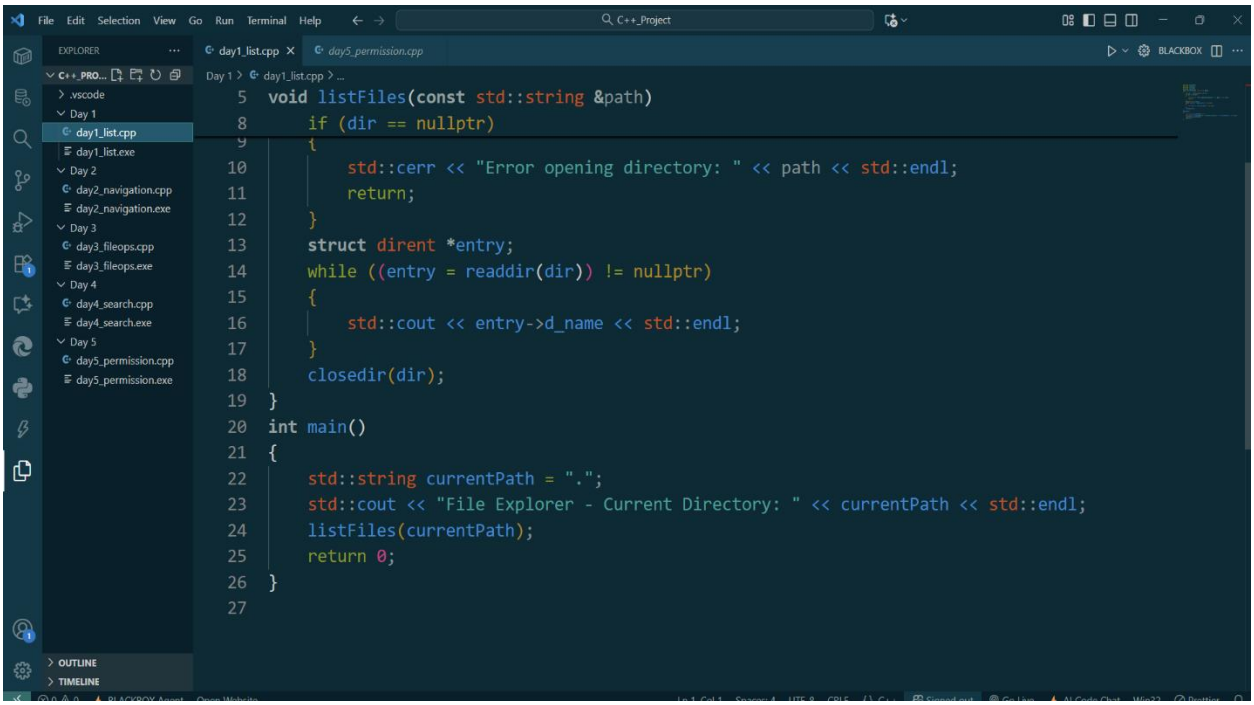
Code:

```
#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.

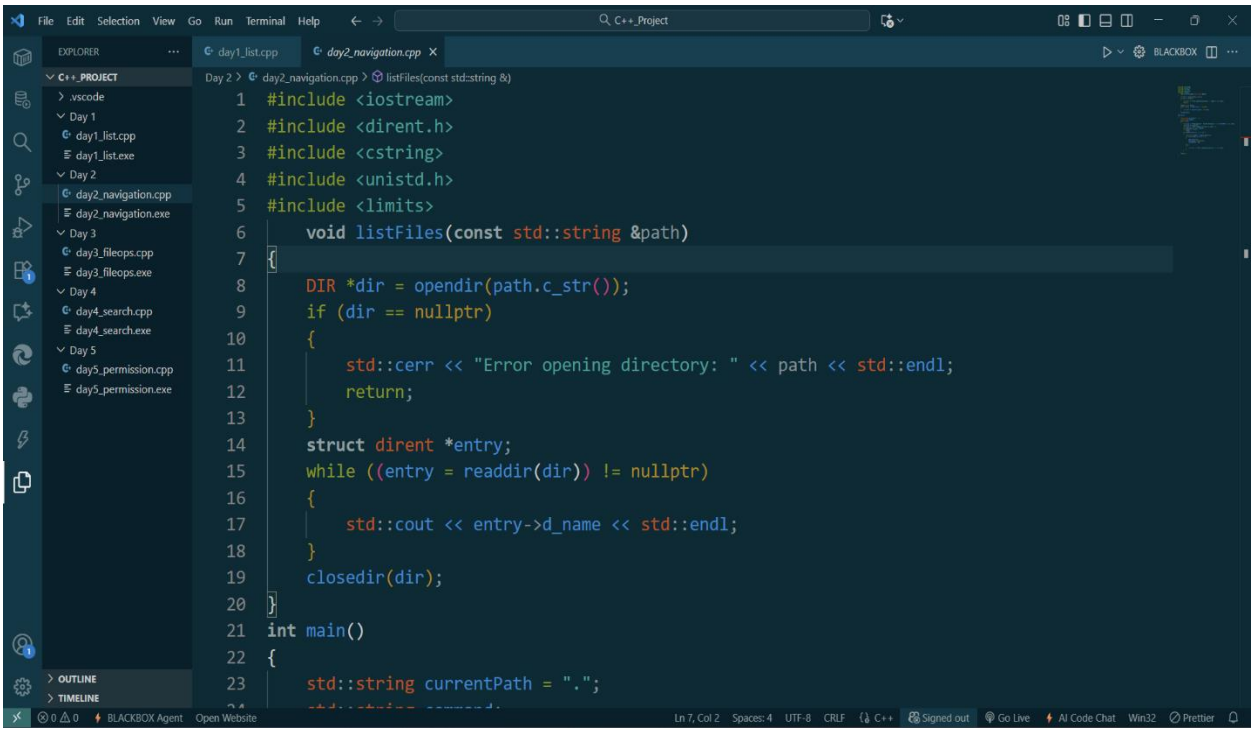
Code:

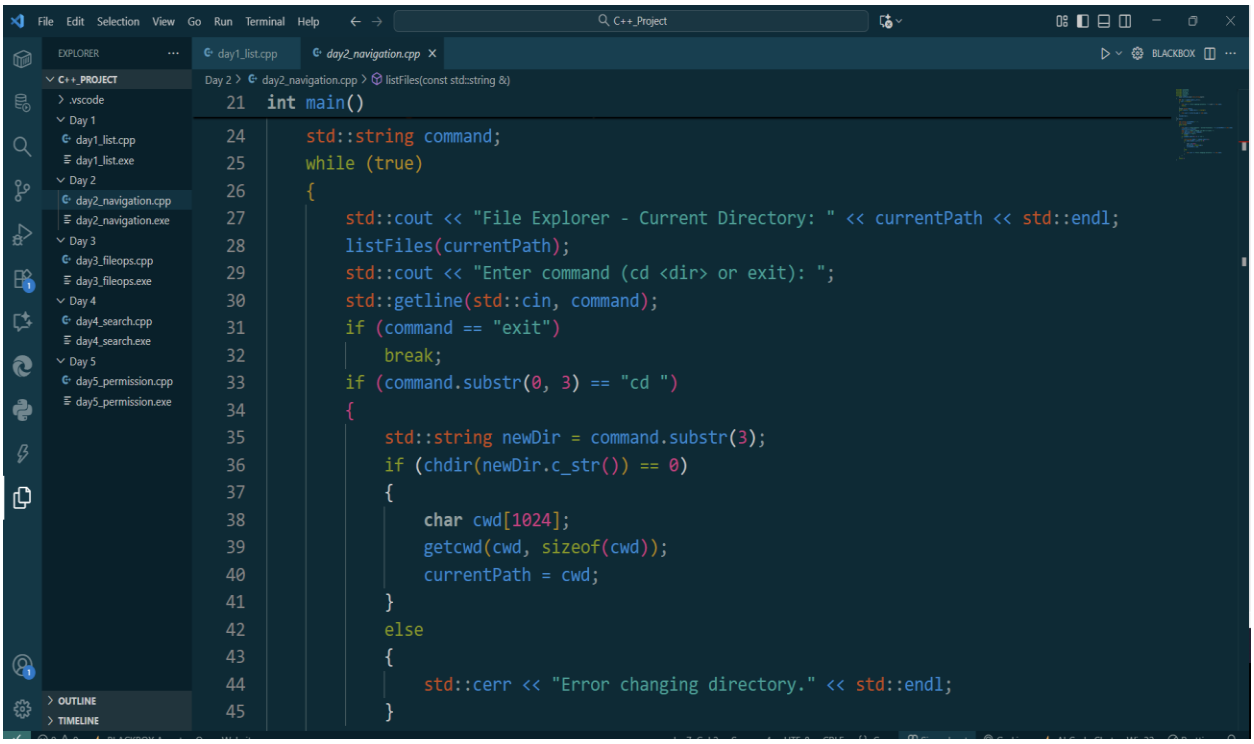
```
#include <iostream>
#include <dirent.h>
#include <cstring>
#include <unistd.h>
#include <limits>
   void listFiles(const std::string &path)
{
   DIR *dir = opendir(path.c_str());
   if (dir == nullptr)
   {
      std::cerr << "Error opening directory: " << path << std::endl;
      return;
   }
```

```
    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
            {
```

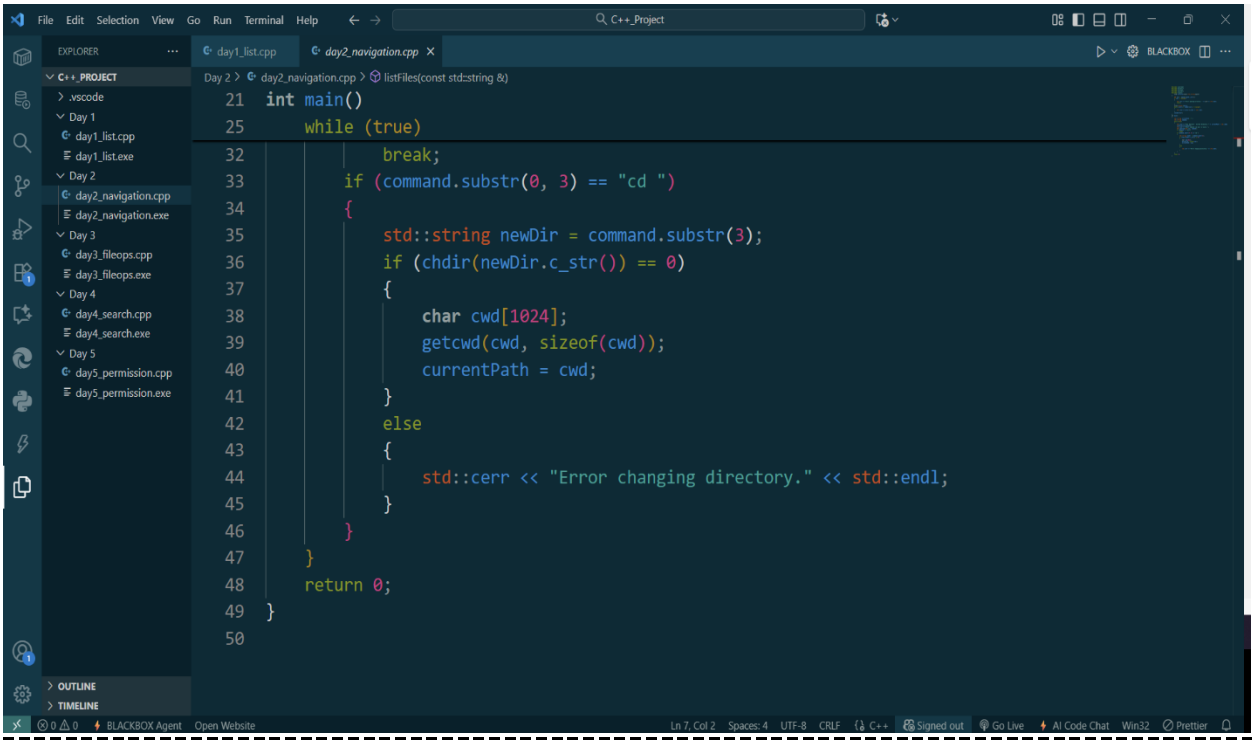
```
            std::cerr << "Error changing directory." << std::endl;
        }
    }
}
return 0;
}
```

Screenshot:





```
int main()
    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;
            }
```



```
int main()
    while (true)
            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;
}
```

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

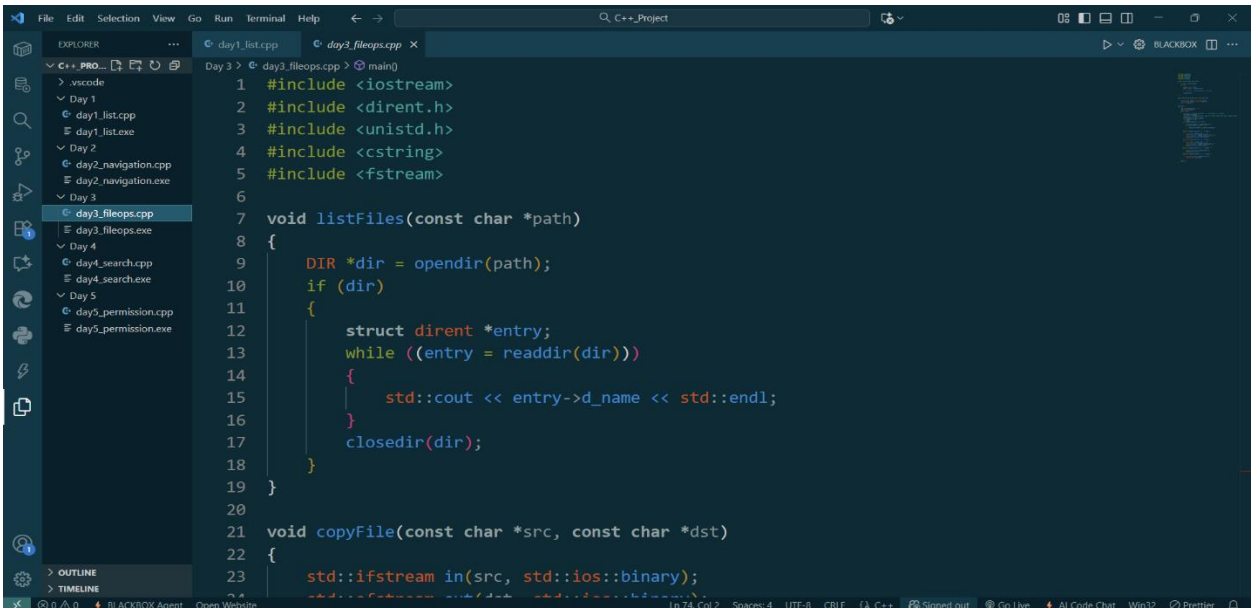
Code:

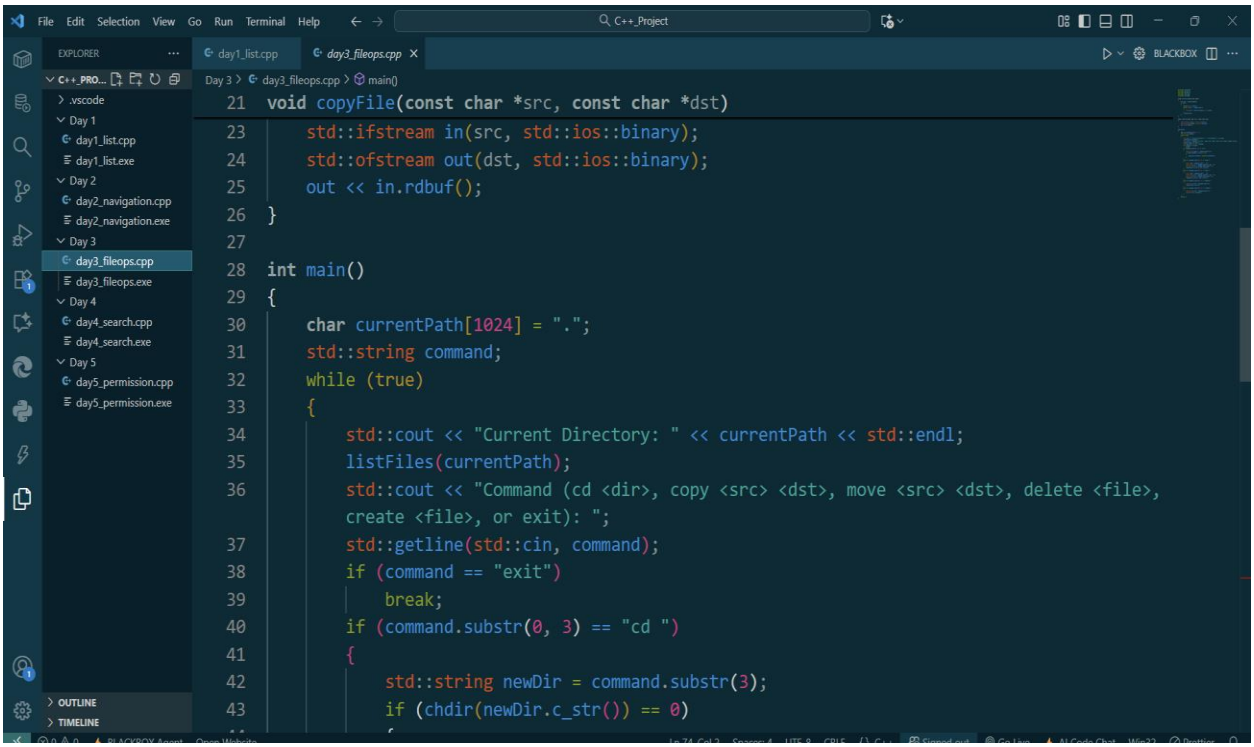
```
#include <iostream>
#include <dirent.h>
#include <unistd.h>
#include <cstring>
#include <fstream>

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()
```

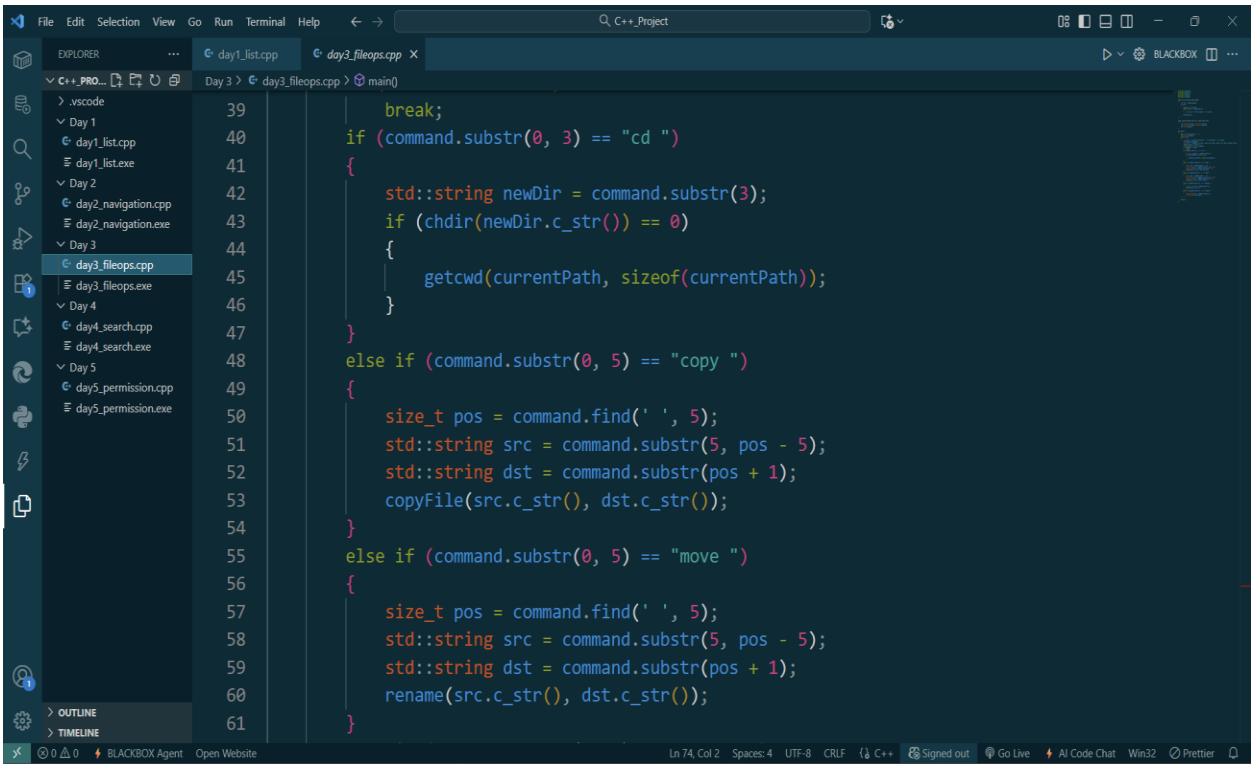
```
{
    char currentPath[1024] = ".";
    std::string command;
    while (true)
    {
        std::cout << "Current Directory: " << currentPath << std::endl;
        listFiles(currentPath);
        std::cout << "Command (cd <dir>, copy <src> <dst>, move <src> <dst>, delete <file>, create <file>,
or 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:




File Edit Selection View Go Run Terminal Help
C++_Project
EXPLORER
day1_list.cpp
day3_fileops.cpp
BLACKBOX
Day 3 > day3_fileops.cpp > main()
C++_PRO...
.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
21 void copyFile(const char *src, const char *dst)
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>,
create <file>, or exit): ";
37 std::getline(std::cin, command);
38 if (command == "exit")
39 break;
40 if (command.substr(0, 3) == "cd ")
41 {
42 std::string newDir = command.substr(3);
43 if (chdir(newDir.c_str()) == 0)


File Edit Selection View Go Run Terminal Help
C++_Project
EXPLORER
day1_list.cpp
day3_fileops.cpp
BLACKBOX
Day 3 > day3_fileops.cpp > main()
C++_PRO...
.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
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 }
BLACKBOX Agent
Open Website
Ln 74, Col 2
Spaces: 4
UTF-8
CRLF
C++
Signed out
Go Live
AI Code Chat
Win32
Prettier

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

Code:

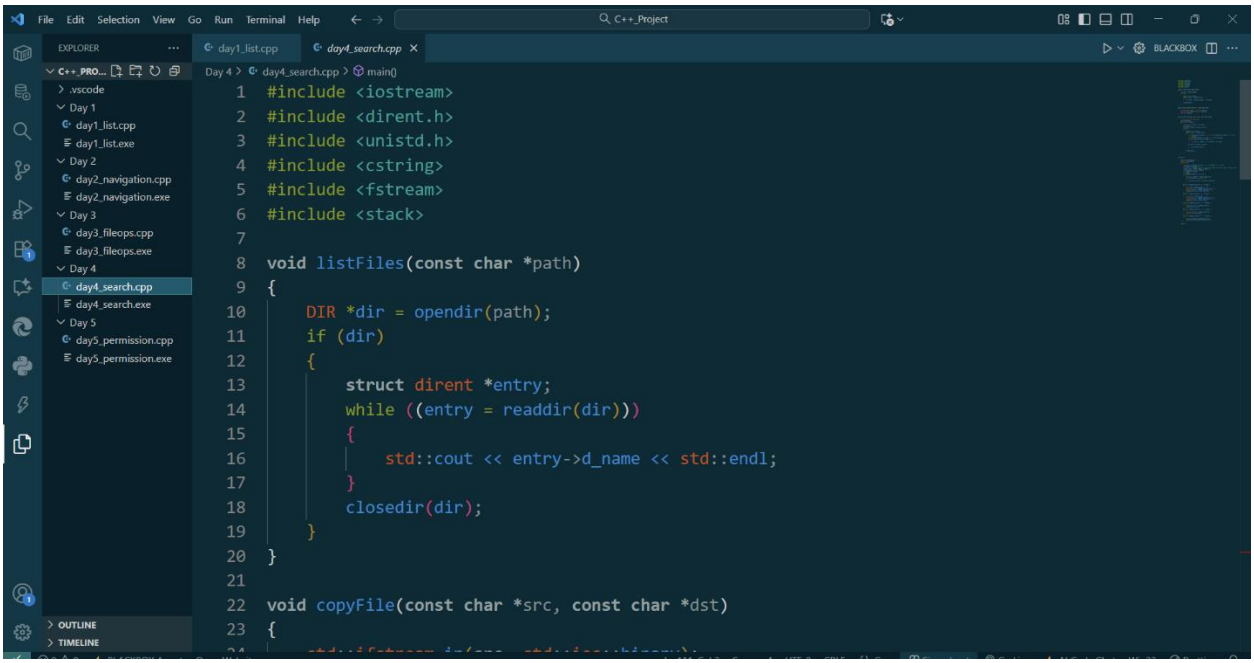
```
#include <iostream>
#include <dirent.h>
#include <unistd.h>
#include <cstring>
#include <fstream>
#include <stack>

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();
}
```

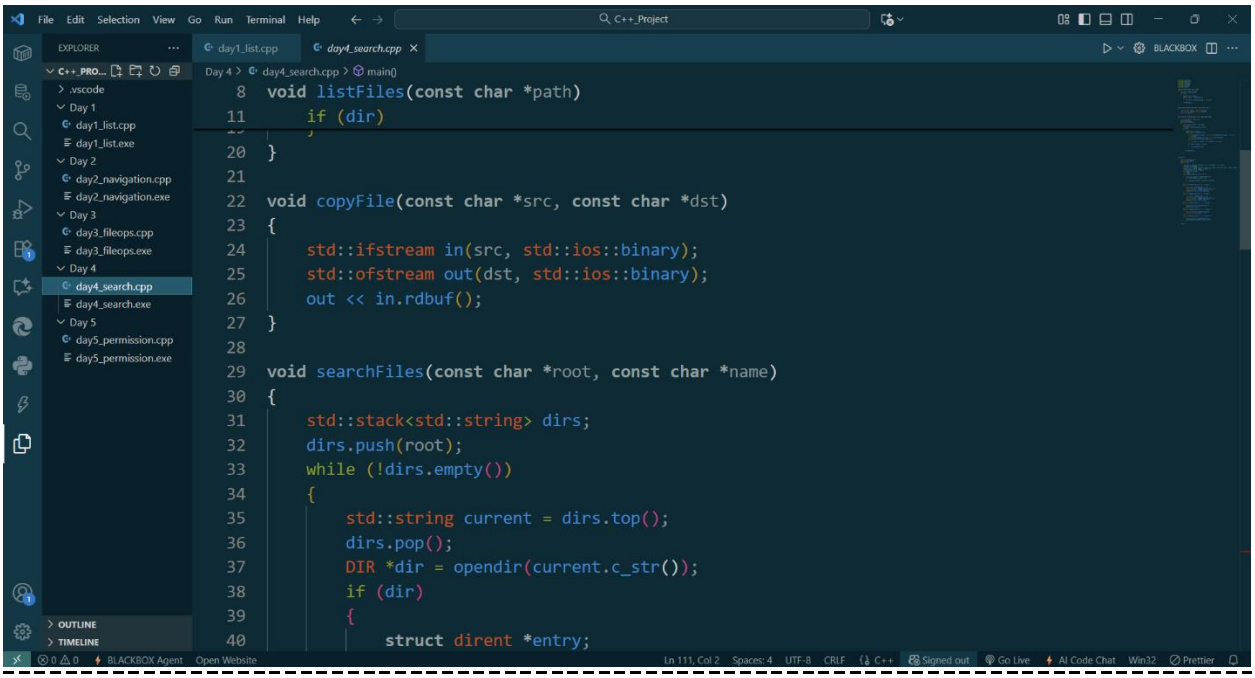
```
void searchFiles(const char *root, const char *name)
{
    std::stack<std::string> dirs;
    dirs.push(root);
    while (!dirs.empty())
    {
        std::string current = dirs.top();
        dirs.pop();
        DIR *dir = opendir(current.c_str());
        if (dir)
        {
            struct dirent *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);
        }
    }
}
```

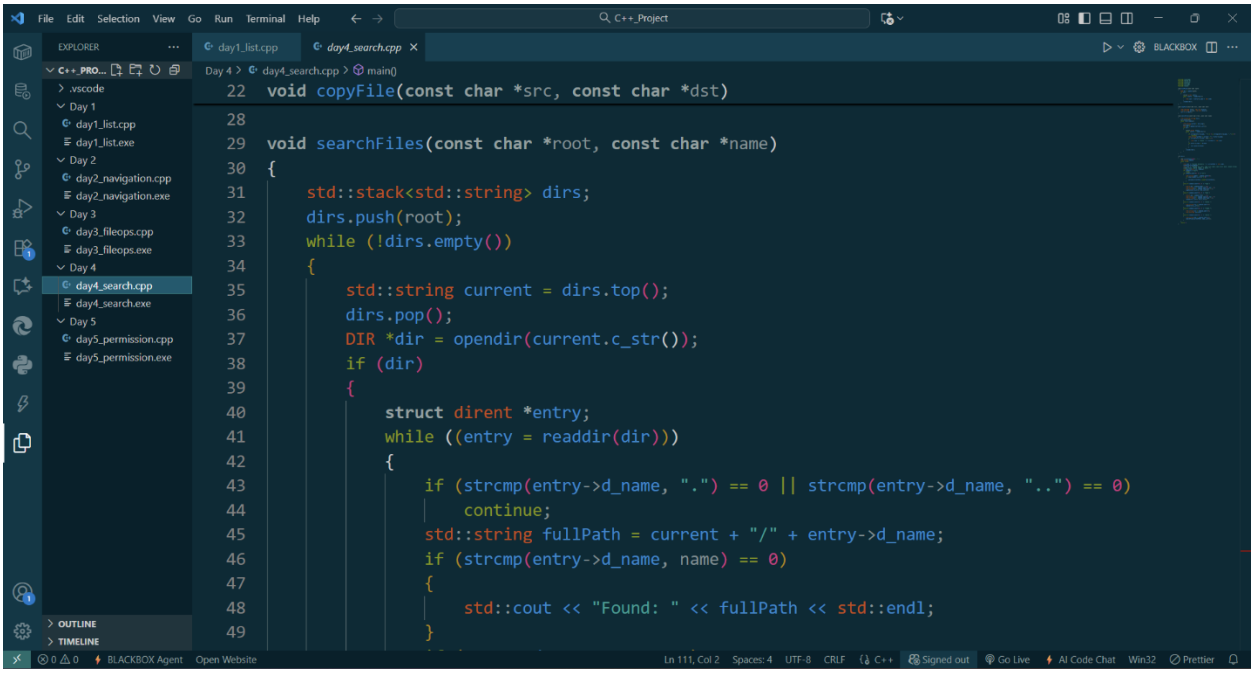
```
int main()
{
    char currentPath[1024] = ".";
    std::string command;
    while (true)
    {
        std::cout << "Current Directory: " << currentPath << std::endl;
        listFiles(currentPath);
        std::cout << "Command (cd <dir>, copy <src> <dst>, move <src> <dst>, delete <file>, create <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;
}
```

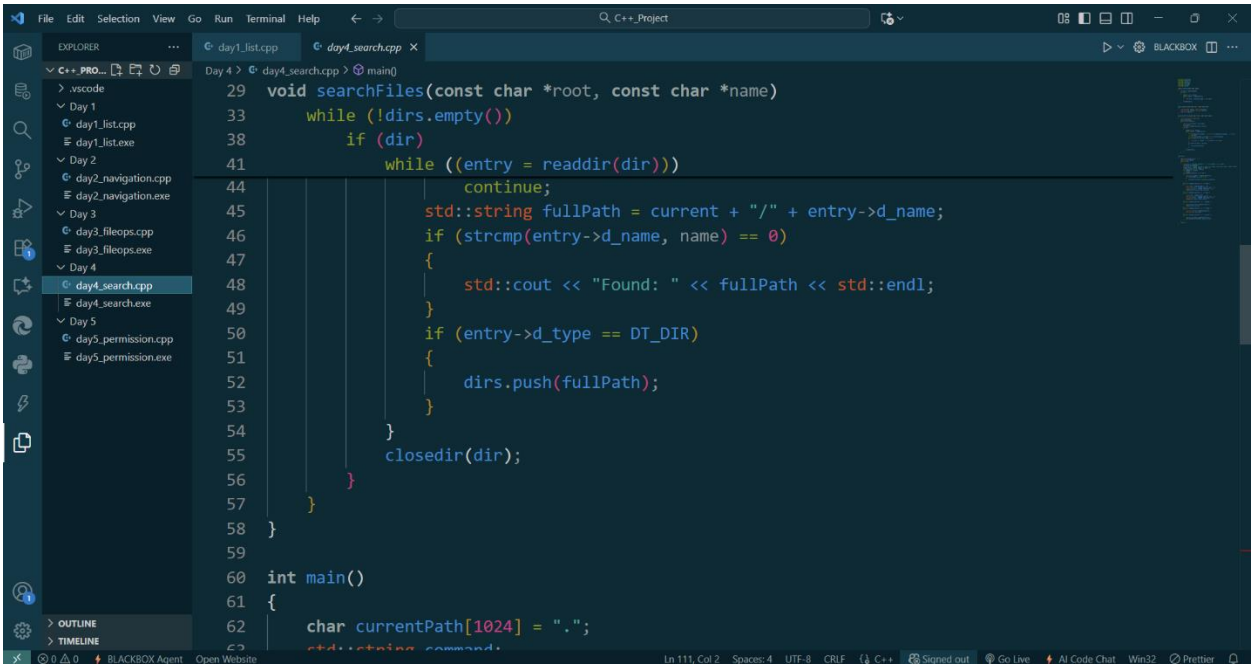
Screenshots:



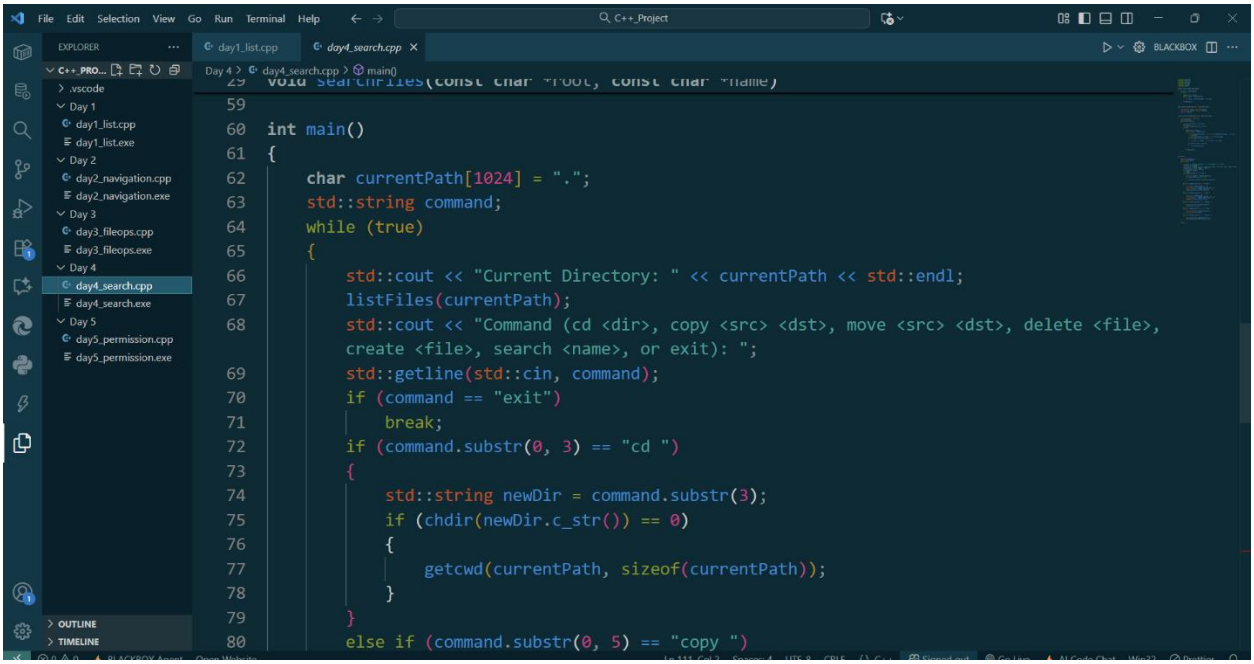




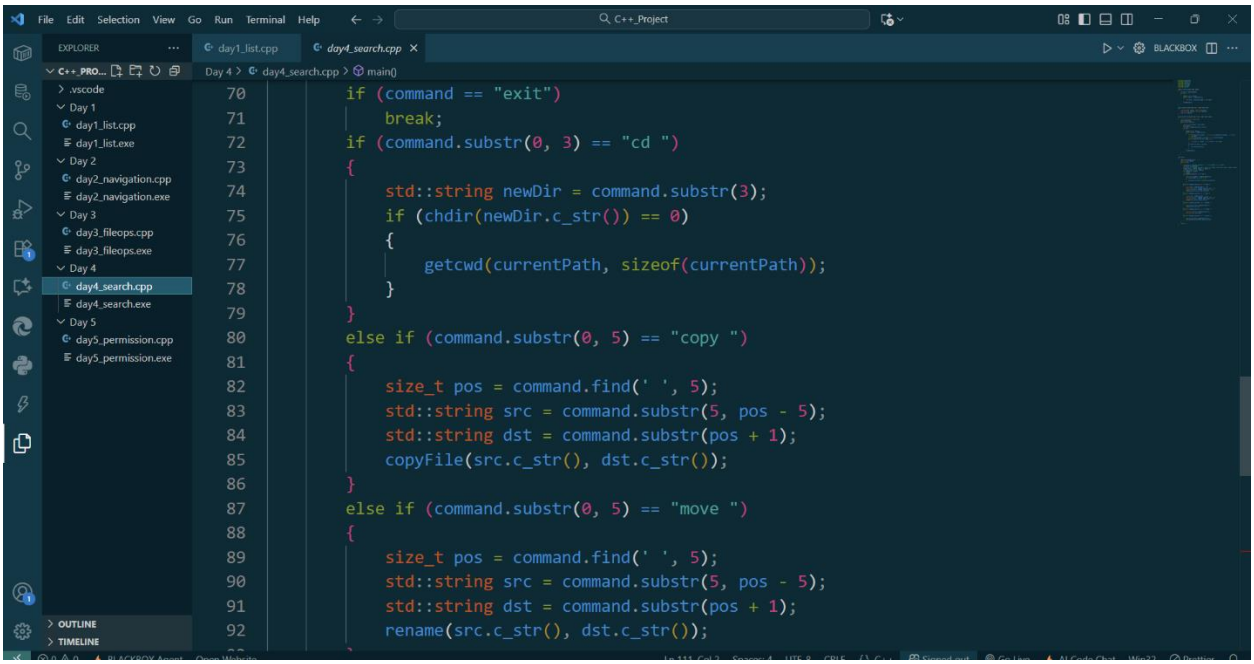
```cpp
void copyFile(const char *src, const char *dst)

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;
                }
```



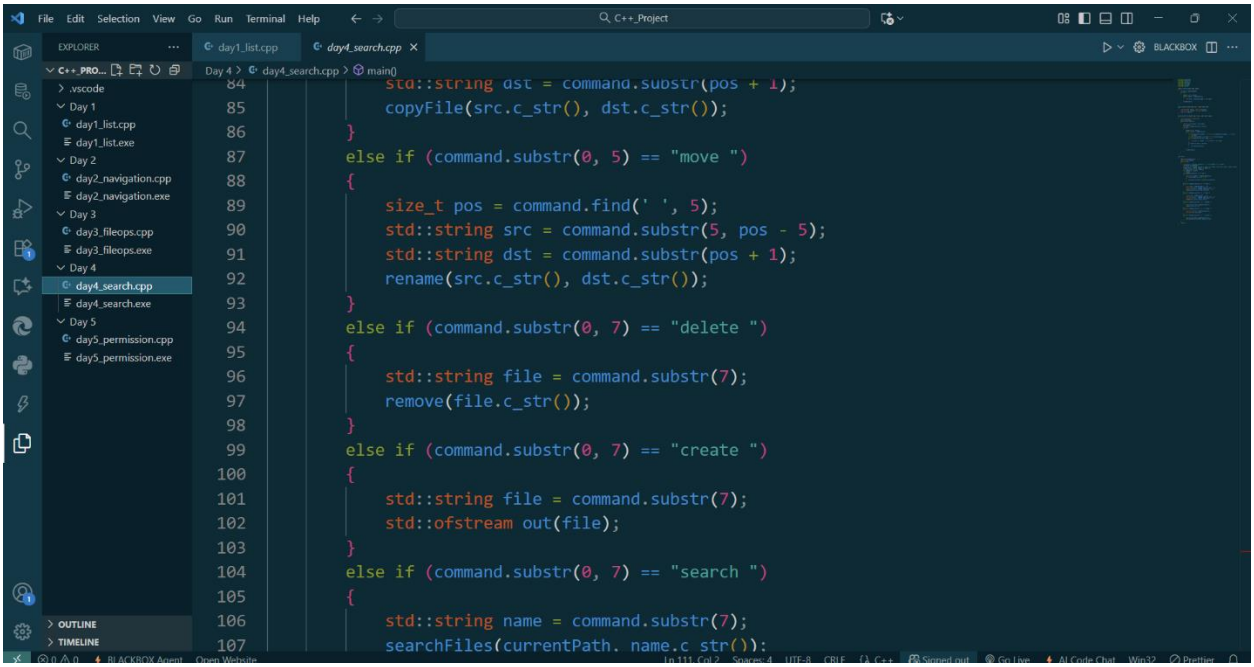
```cpp
void searchFiles(const char *root, const char *name)
    while (!dirs.empty())
        if (dir)
            while ((entry = readdir(dir)))
                continue;
            std::string fullPath = current + "/" + entry->d_name;
            if (strcmp(entry->d_name, name) == 0)
            {
                std::cout << "Found: " << fullPath << std::endl;
            }
            if (entry->d_type == DT_DIR)
            {
                dirs.push(fullPath);
            }
        }
        closedir(dir);
    }
}

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

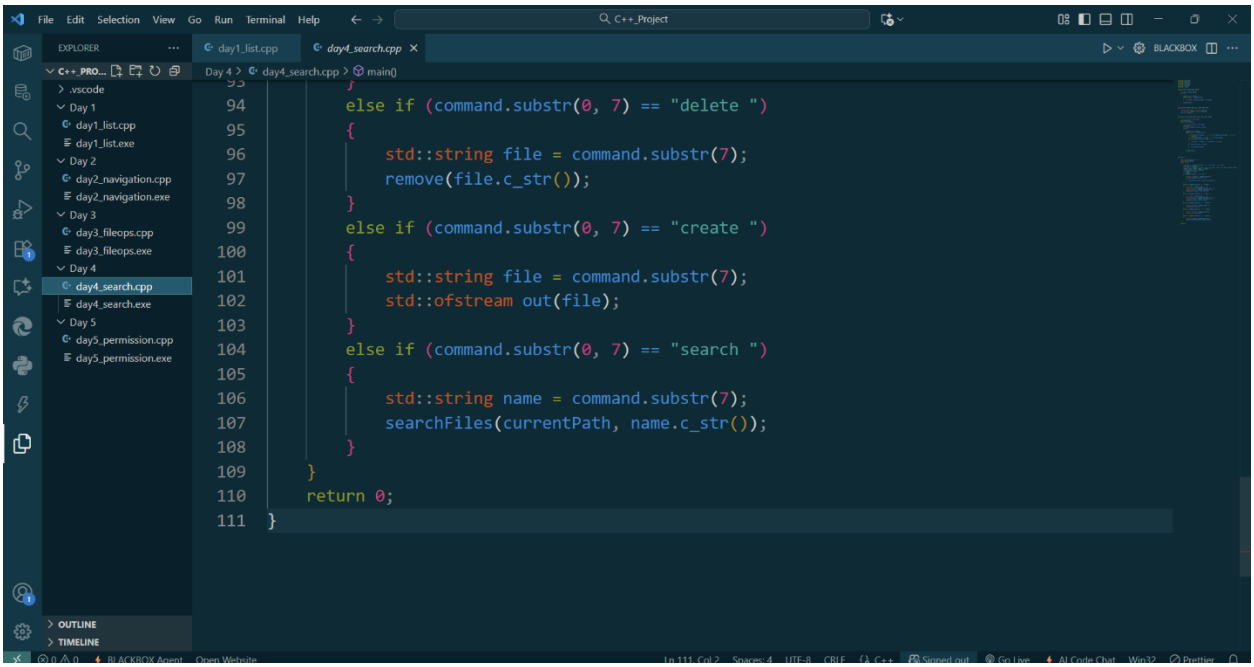


```
59
60  int main()
61  {
62      char currentPath[1024] = ".";
63      std::string command;
64      while (true)
65      {
66          std::cout << "Current Directory: " << currentPath << std::endl;
67          listFiles(currentPath);
68          std::cout << "Command (cd <dir>, copy <src> <dst>, 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 ")
```



```
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());
```


File Edit Selection View Go Run Terminal Help
C++_Project
EXPLORER
day1_list.cpp
day4_search.cpp
BLACKBOX
Day 4 > day4_search.cpp > main()
C++_PRO...
.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
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());


File Edit Selection View Go Run Terminal Help
C++_Project
EXPLORER
day1_list.cpp
day4_search.cpp
BLACKBOX
Day 4 > day4_search.cpp > main()
C++_PRO...
.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
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;
}

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();
}
```

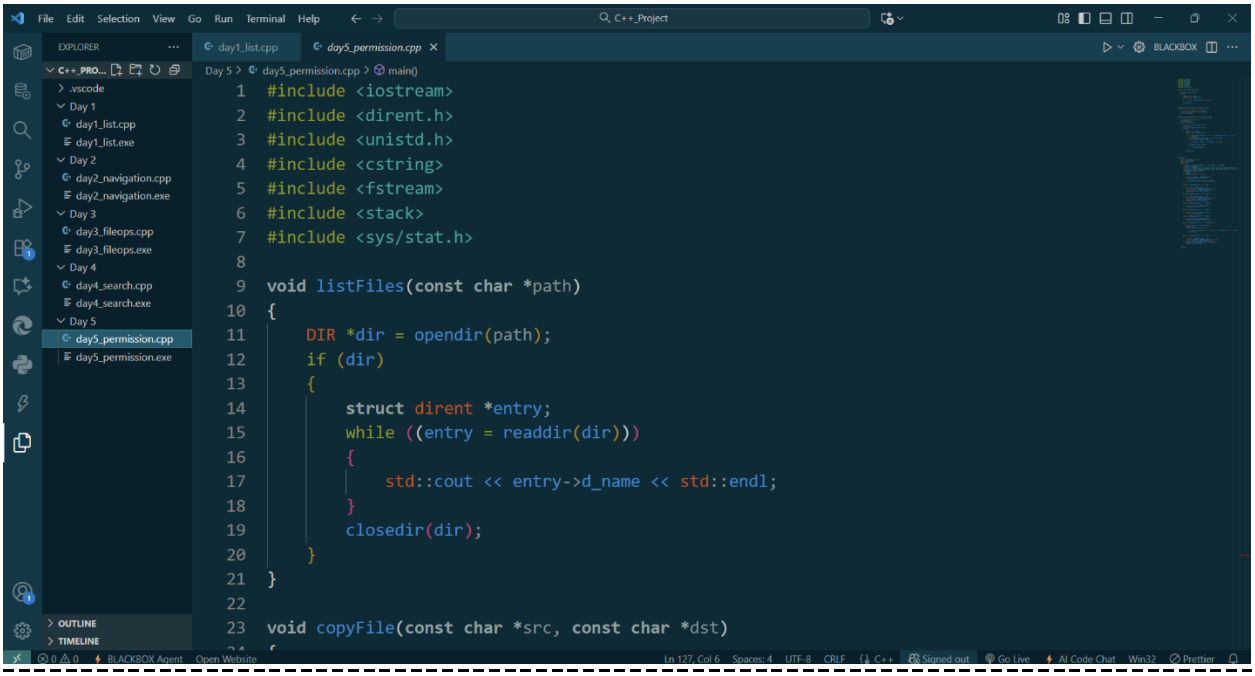
```
void searchFiles(const char *root, const char *name)
{
    std::stack<std::string> dirs;
    dirs. push(root);
    while (!dirs.empty ())
    {
        std::string current = dirs.top();
        dirs.pop();
        DIR *dir = opendir(current.c_str());
        if (dir)
        {
            struct 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);
        }
    }
```

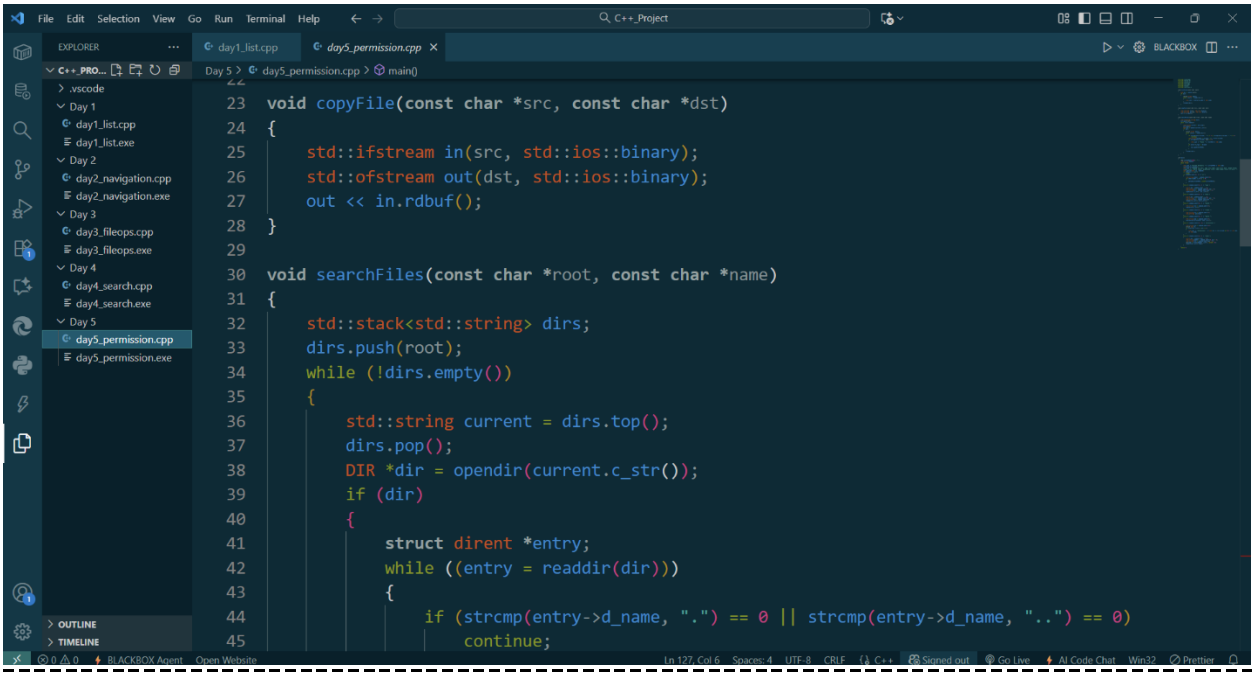
```
}

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;
    }
}
```

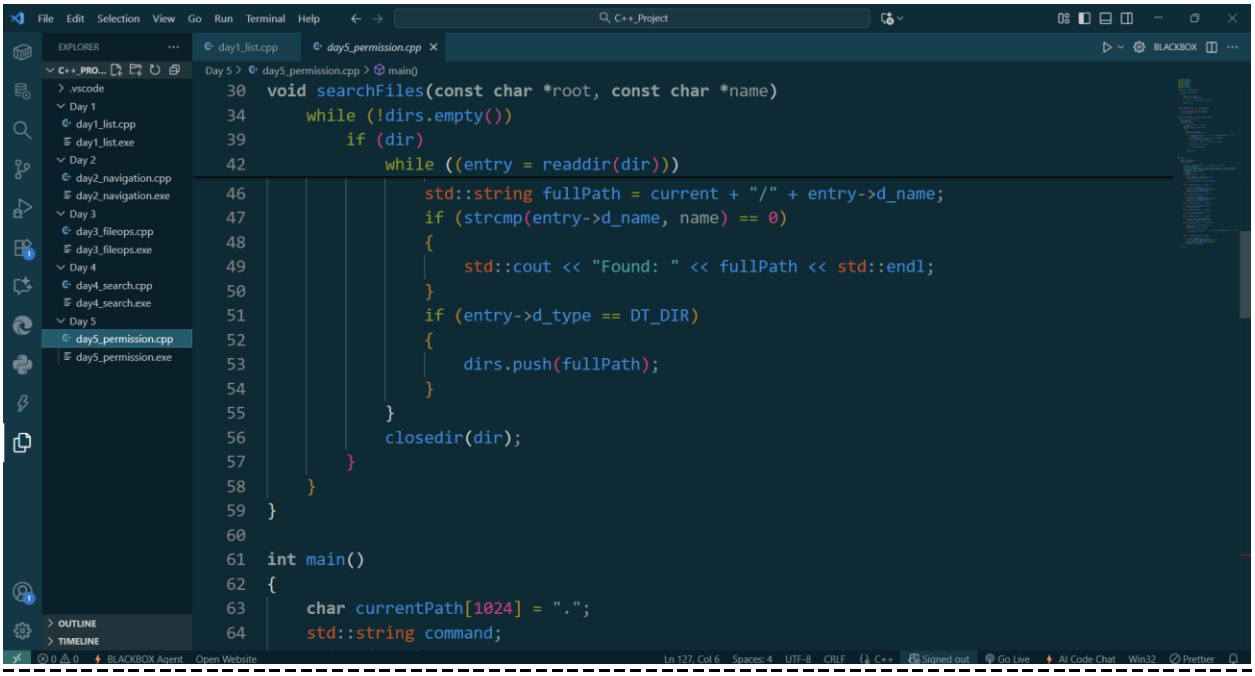
```
        else if (command.substr(0, 6) == "chmod ")
        {
            size_t pos = command.find(' ', 6);
            std::string modeStr = command.substr(6, pos - 6);
            std::string file = command.substr(pos + 1);
            int mode = strtol (modeStr.c_str(), nullptr, 8);
         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

Screenshot:

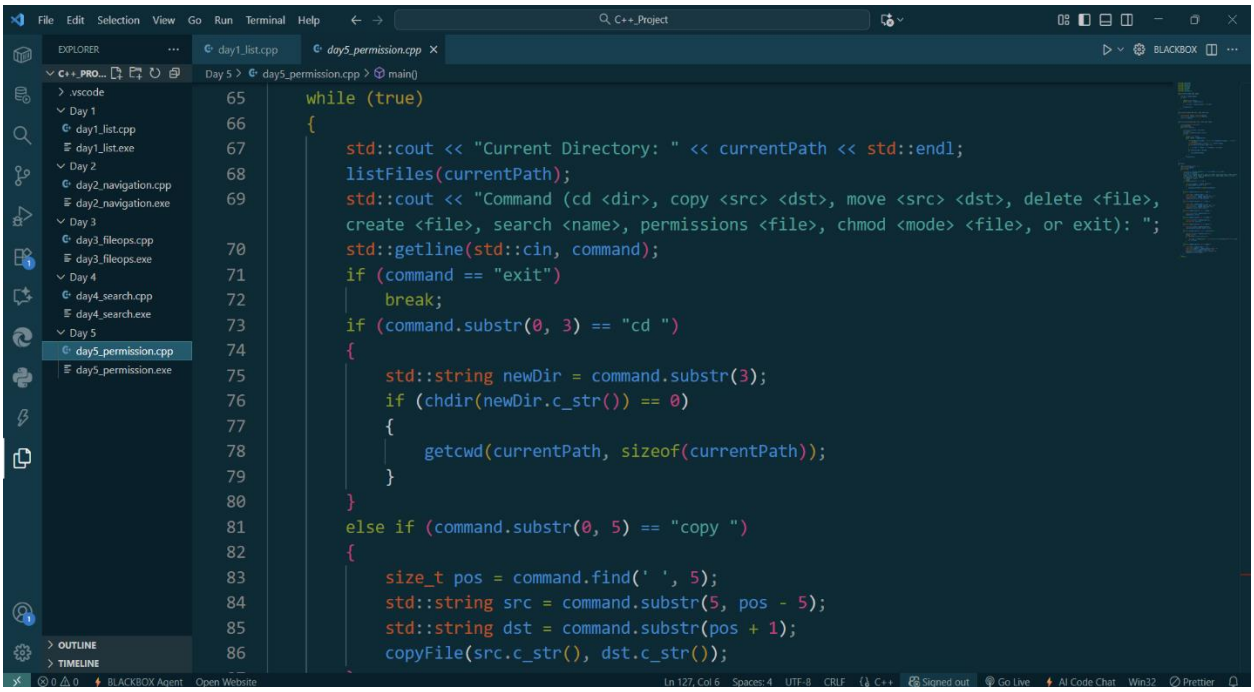




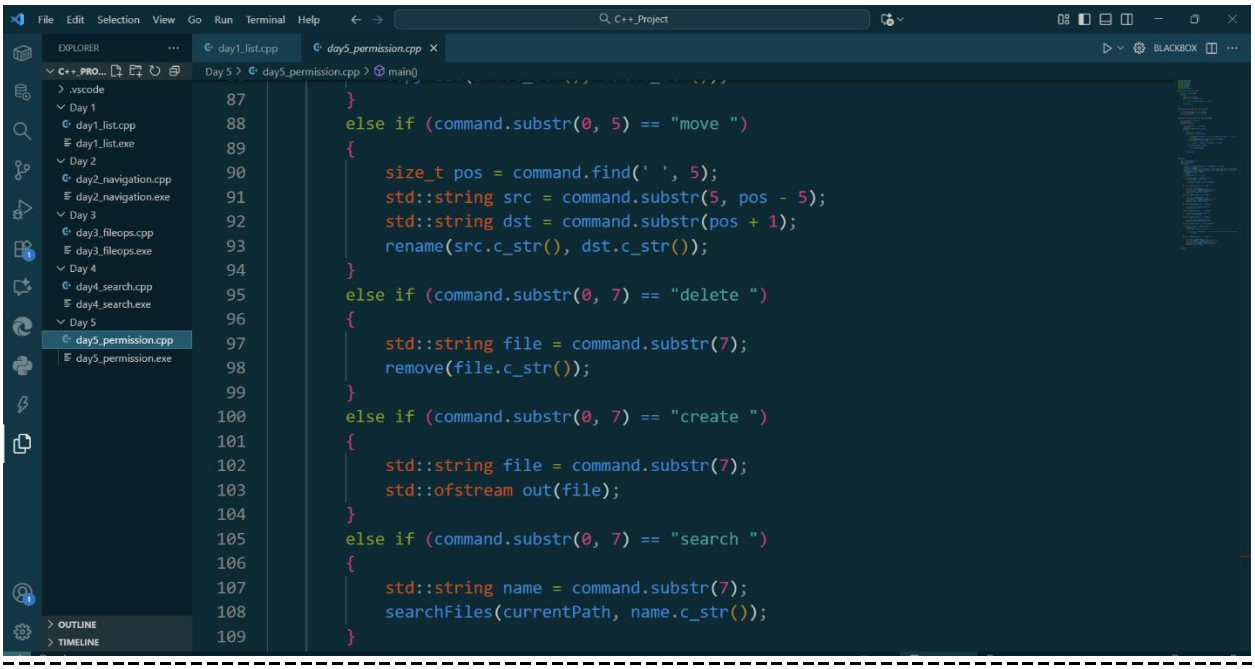
```cpp
void copyFile(const char *src, const char *dst)
{
    std::ifstream in(src, std::ios::binary);
    std::ofstream out(dst, std::ios::binary);
    out << in.rdbuf();
}

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;
```



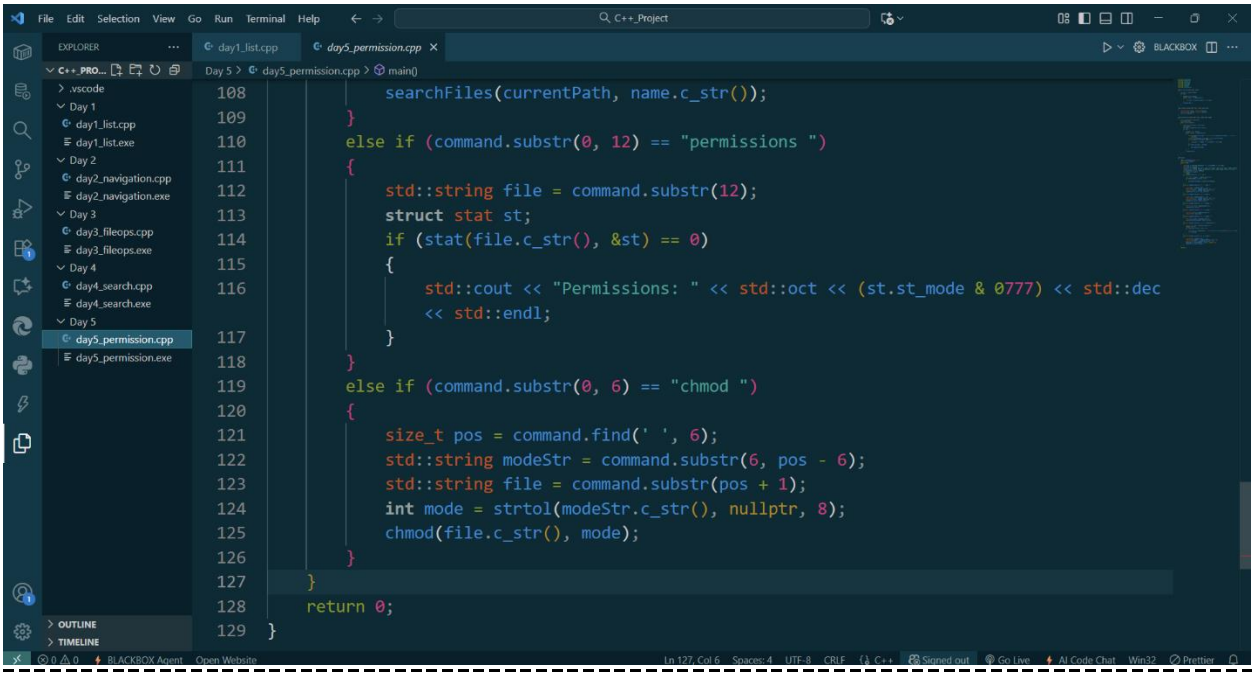
```cpp
void searchFiles(const char *root, const char *name)
    while (!dirs.empty())
        if (dir)
            while ((entry = readdir(dir)))
                std::string fullPath = current + "/" + entry->d_name;
                if (strcmp(entry->d_name, name) == 0)
                {
                    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;
```



```
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>,
            create <file>, search <name>, permissions <file>, chmod <mode> <file>, or exit): ";
70          std::getline(std::cin, command);
71          if (command == "exit")
72              break;
73          if (command.substr(0, 3) == "cd ")
74          {
75              std::string newDir = command.substr(3);
76              if (chdir(newDir.c_str()) == 0)
77              {
78                  getcwd(currentPath, sizeof(currentPath));
79              }
80          }
81          else if (command.substr(0, 5) == "copy ")
82          {
83              size_t pos = command.find(' ', 5);
84              std::string src = command.substr(5, pos - 5);
85              std::string dst = command.substr(pos + 1);
86              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         }
```


File Edit Selection View Go Run Terminal Help
C++_Project
EXPLORER
day1_list.cpp
day5_permission.cpp
BLACKBOX
Day 5 > day5_permission.cpp > main()
C++_PRO...
.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
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;
}
BLACKBOX Agent
Open Website
Ln 127, Col 6
Spaces: 4
UTF-8
CRLF
C++
Signed out
Go Live
AI Code Chat
Win32
Prettier