

Learning C

The following code was able to print my name in the console

Was also able to open and read my file in d:\Documents\Vs Code Projects\Coding Assignments\Group 1\school.c

Was also able to prints out the content in that .c file in the console

```
#include<stdio.h>

int main()
{
    printf("hello Katchiets");

    FILE *file;
    char line[100]; // Assuming lines in the file are no longer than 100 characters.

    // Open the file for reading.
    file = fopen("D:\\Documents\\Vs Code Projects\\Coding Assignments\\Group 1\\school.c", "r");

    if (file == NULL)
    {
        printf("Error opening the file.\n");
        return 1;
    }

    // Read and print each line until the end of the file.
    while (fgets(line, sizeof(line), file) != NULL)
    {
        printf("%s", line);
    }

    // Close the file.
    fclose(file);
}
```

In C strings, backslashes () are used as escape characters to represent special characters. For example, the escape sequence `\n` is used to represent a newline character, `\t` for a tab character, etc. Because backslashes have this special meaning, if you want to include an actual backslash character in a C string, you need to escape it with another backslash.

Here's a brief explanation of how escaping works in C strings:

1. **Single Backslash:** If you want to include a single backslash in a C string, you need to escape it by using another backslash. For example, to represent a string containing a single backslash, you write `"\\\\"`. The first backslash escapes the second one, resulting in a single backslash in the actual string.
2. **Forward Slash:** Forward slashes (/) do not have a special meaning in C strings, so they don't need to be escaped. You can use forward slashes as regular characters in C strings. For example, `"folder/file.txt"` is a valid C string.
3. **Path Separators:** In Windows file paths, backslashes are used as path separators (e.g., `C:\Documents\file.txt`). However, since backslashes are escape characters in C strings, you need to use double backslashes (`\\`) to represent a single backslash in the path. So, in C strings, a Windows file path would look like `"C:\\Documents\\file.txt"`.

Here's an example to illustrate this with a C string containing a Windows file path:

```
#include <stdio.h>

int main()
{
    // A C string representing a Windows file path
    char path[] = "D:\\Documents\\Vs Code Projects\\Coding Assignments\\Group 1\\school.c";

    // Print the path to the console
    printf("File Path: %s\n", path);

    return 0;
}
```

In this example, the **path** variable contains a C string representing a Windows file path. We use double backslashes to escape each backslash, ensuring that the path is correctly represented as a C string. When we print the **path** variable using **printf()**, it will display the correct file path:

File Path: D:\Documents\Vs Code Projects\Coding Assignments\Group 1\school.c

Remember that if you're using forward slashes in your file path, you don't need to escape them. They can be used as regular characters in C strings. However, using double backslashes for path separators is a common practice to ensure portability and consistency across different platforms.

This failed execution

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

void listDirectoriesAndFiles(const char *path);

int main()
{
    printf("Listing directories and files in D:\\Documents:\\n");
    listDirectoriesAndFiles("D:\\Documents");
    return 0;
}

void listDirectoriesAndFiles(const char *path)
{
    DIR *dir;
    struct dirent *entry;
    char folders[1000][FILENAME_MAX]; // Array to store unique folder names
    int numFolders = 0; // Counter for the number of unique folders

    dir = opendir(path);
    if (dir == NULL)
    {
        printf("Error opening directory: %s\\n", path);
        return;
    }

    // Loop through the directory entries and find unique folder names.
    while ((entry = readdir(dir)) != NULL)
    {
        // Skip the ".", and ".." entries and non-directory entries.
        if (entry->d_type != DT_DIR || strcmp(entry->d_name, ".") == 0 || strcmp(entry->d_name, "..") == 0)
            continue;

        int found = 0;
        // Check if the folder name is already in the list of unique folders.
        for (int i = 0; i < numFolders; i++)
        {
            if (strcmp(folders[i], entry->d_name) == 0)
            {
                found = 1;
                break;
            }
        }

        if (!found)
        {
            // If the folder name is not found in the list, add it to the list.
            strcpy(folders[numFolders], entry->d_name);
            numFolders++;
        }
    }

    closedir(dir);

    // Print the summary of unique folder names.
    printf("Folders present in D:\\Documents include (");
    for (int i = 0; i < numFolders; i++)
    {
        printf("%s", folders[i]);
        if (i < numFolders - 1)
            printf(", ");
    }
    printf(")\\n");
}
```

The following code failed execution:

```
#include <stdio.h>
#include <windows.h>

void listDirectoriesInRoot(const wchar_t *path);

int main()
{
    printf("Listing directories in D:\\Documents:\\n");
    listDirectoriesInRoot(L"D:\\Documents"); // Note the L before the string to create a wide-character string
    return 0;
}

void listDirectoriesInRoot(const wchar_t *path)
{
    WIN32_FIND_DATA findFileData; // Use the wide-character version of the structure
    HANDLE hFind = FindFirstFileW(path, &findFileData); // Use the wide-character version of the function
    if (hFind == INVALID_HANDLE_VALUE)
    {
        printf("Error finding directories in: %S\\n", path); // Use %S to print wide-character strings
        return;
    }

    // Loop through the directory entries and find directories.
    do
    {
        if (findFileData.dwFileAttributes & FILE_ATTRIBUTE_DIRECTORY)
        {
            if (wcscmp(findFileData.cFileName, L".") != 0 && wcscmp(findFileData.cFileName, L"..") != 0)
            {
                wprintf(L"Directory: %s\\n", findFileData.cFileName);
            }
        }
    } while (FindNextFileW(hFind, &findFileData) != 0); // Use the wide-character version of the function

    FindClose(hFind);
}
```

This also failed Execution

```
#include <stdio.h>
#include <windows.h>

void listDirectoriesInRoot(const char *path);

int main()
{
    printf("Listing directories in D:\\Documents:\\n");
    listDirectoriesInRoot("D:\\Documents");
    return 0;
}

void listDirectoriesInRoot(const char *path)
{
    WIN32_FIND_DATA findFileData;
    HANDLE hFind = FindFirstFile((LPCWSTR)path, &findFileData);
    if (hFind == INVALID_HANDLE_VALUE)
    {
        printf("Error finding directories in: %s\\n", path);
        return;
    }

    // Loop through the directory entries and find directories.
    do
    {
        if (findFileData.dwFileAttributes & FILE_ATTRIBUTE_DIRECTORY)
        {
            if (wcscmp(findFileData.cFileName, L".") != 0 && wcscmp(findFileData.cFileName, L"..") != 0)
            {
                wprintf(L"Directory: %s\\n", findFileData.cFileName);
            }
        }
    } while (FindNextFile(hFind, &findFileData) != 0);

    FindClose(hFind);
}
```

The following code was able to access my D:/Documents path and return all the directories in the root

```
#include <stdio.h>
#include <windows.h>

void listDirectoriesInRoot(const wchar_t *path);

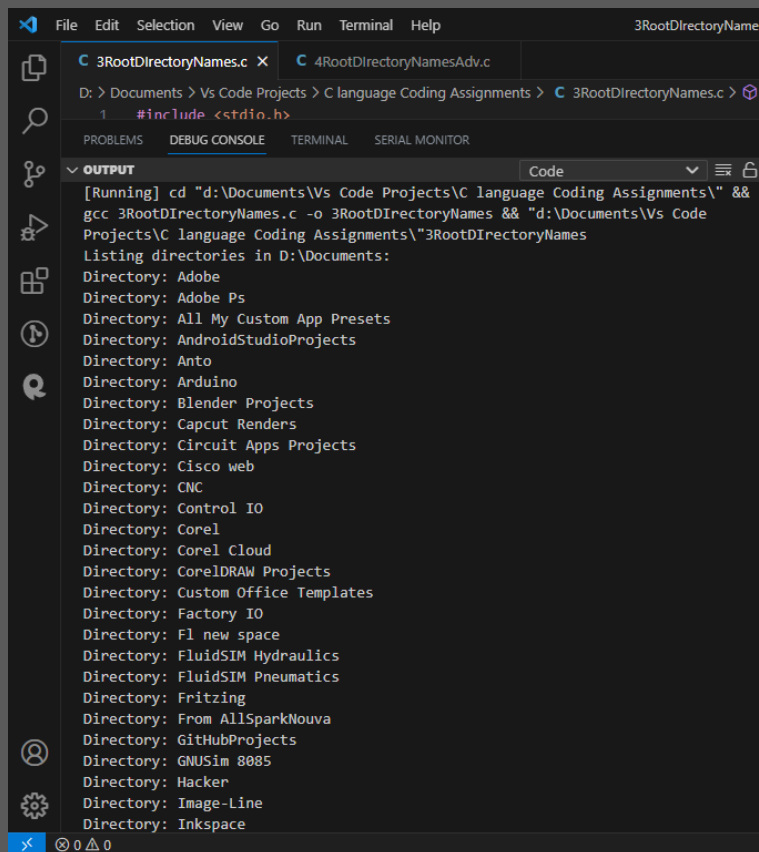
int main()
{
    printf("Listing directories in D:\\Documents:\\n");
    listDirectoriesInRoot(L"D:\\Documents");
    return 0;
}

void listDirectoriesInRoot(const wchar_t *path)
{
    WIN32_FIND_DATAW findFileData;
    HANDLE hFind = FindFirstFileW((wchar_t *)L"D:\\Documents\\*", &findFileData);

    if (hFind == INVALID_HANDLE_VALUE)
    {
        printf("Error finding directories in: %S\\n", path);
        return;
    }

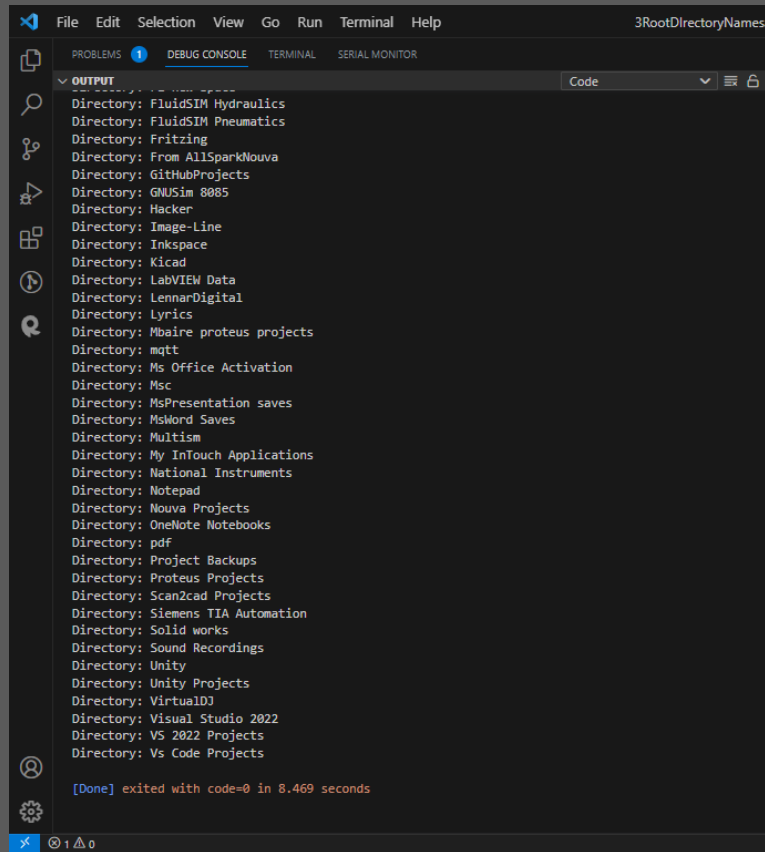
    // Loop through the directory entries and find directories.
    do
    {
        if (findFileData.dwFileAttributes & FILE_ATTRIBUTE_DIRECTORY)
        {
            if (wcscmp(findFileData.cFileName, L".") != 0 && wcscmp(findFileData.cFileName, L"..") != 0)
            {
                wprintf(L"Directory: %s\\n", findFileData.cFileName);
            }
        }
    } while (FindNextFileW(hFind, &findFileData) != 0);

    FindClose(hFind);
}
```



The screenshot shows the VS Code interface with the file explorer on the left and the terminal at the bottom. The terminal output is as follows:

```
[Running] cd "d:\Documents\Vs Code Projects\C language Coding Assignments\" &&
gcc 3RootDirectoryNames.c -o 3RootDirectoryNames && "d:\Documents\Vs Code
Projects\C language Coding Assignments\3RootDirectoryNames
Listing directories in D:\Documents:
Directory: Adobe
Directory: Adobe Ps
Directory: All My Custom App Presets
Directory: AndroidStudioProjects
Directory: Anto
Directory: Arduino
Directory: Blender Projects
Directory: Capcut Renders
Directory: Circuit Apps Projects
Directory: Cisco web
Directory: CNC
Directory: Control IO
Directory: Corel
Directory: Corel Cloud
Directory: CorelDRAW Projects
Directory: Custom Office Templates
Directory: Factory IO
Directory: Fl new space
Directory: FluidSIM Hydraulics
Directory: FluidSIM Pneumatics
Directory: Fritzing
Directory: From AllSparkNouva
Directory: GitHubProjects
Directory: GNUMSim 8085
Directory: Hacker
Directory: Image-Line
Directory: Inkspace
```



The screenshot shows the VS Code interface with the debug console at the bottom. The output is as follows:

```
Directory: FluidSIM Hydraulics
Directory: FluidSIM Pneumatics
Directory: Fritzing
Directory: From AllSparkNouva
Directory: GitHubProjects
Directory: GNUMSim 8085
Directory: Hacker
Directory: Image-Line
Directory: Inkspace
Directory: Kicad
Directory: LabVIEW Data
Directory: LennarDigital
Directory: Lyrics
Directory: Mbaire proteus projects
Directory: mqtt
Directory: Ms Office Activation
Directory: Msc
Directory: MsPresentation saves
Directory: MsWord Saves
Directory: Multism
Directory: My InTouch Applications
Directory: National Instruments
Directory: Notepad
Directory: Nouva Projects
Directory: OneNote Notebooks
Directory: pdf
Directory: Project Backups
Directory: Proteus Projects
Directory: Scan2cad Projects
Directory: Siemens TIA Automation
Directory: Solid works
Directory: Sound Recordings
Directory: Unity
Directory: Unity Projects
Directory: VirtualDJ
Directory: Visual Studio 2022
Directory: VS 2022 Projects
Directory: Vs Code Projects

[Done] exited with code=0 in 8.469 seconds
```

The following code returned me the files and directories present in the root but in alphabetical order:

```
#include <stdio.h>
#include <windows.h>

void listDirectoriesAndFilesInRoot(const wchar_t *path);

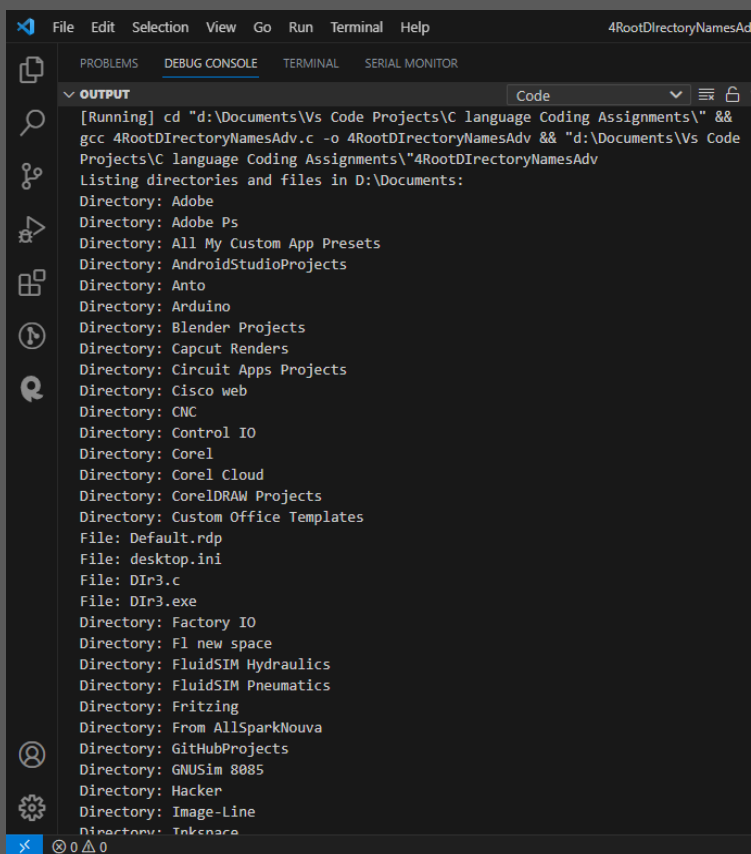
int main()
{
    printf("Listing directories and files in D:\\Documents:\\n");
    listDirectoriesAndFilesInRoot(L"D:\\Documents");
    return 0;
}

void listDirectoriesAndFilesInRoot(const wchar_t *path)
{
    WIN32_FIND_DATAW findFileData;
    HANDLE hFind = FindFirstFileW((wchar_t *)L"D:\\Documents\\*", &findFileData);

    if (hFind == INVALID_HANDLE_VALUE)
    {
        printf("Error finding directories and files in: %S\\n", path);
        return;
    }

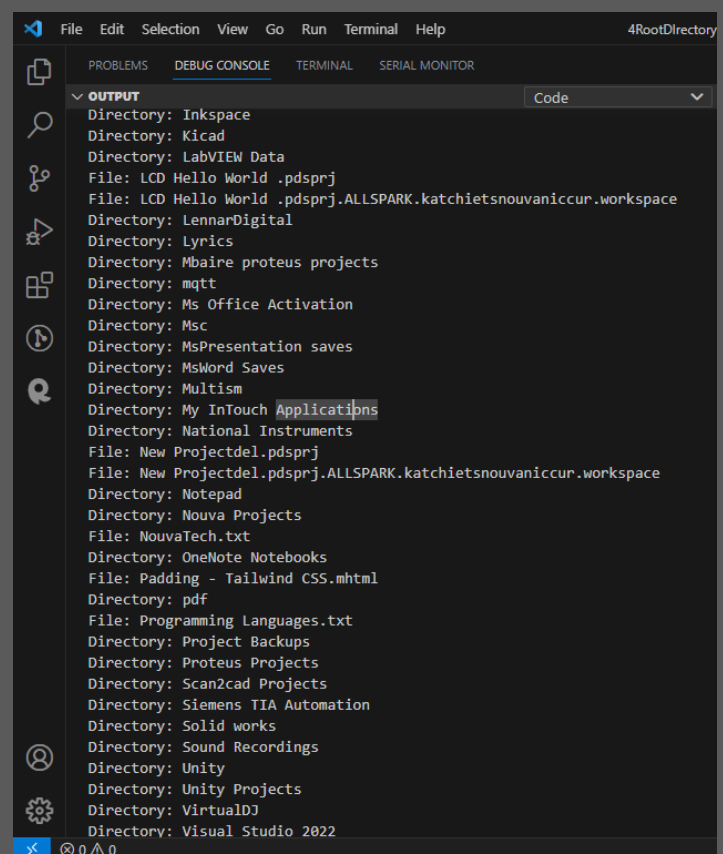
    // Loop through the directory entries and find directories and non-directory files.
    do
    {
        if (findFileData.dwFileAttributes & FILE_ATTRIBUTE_DIRECTORY)
        {
            if (wcscmp(findFileData.cFileName, L".") != 0 && wcscmp(findFileData.cFileName, L"..") != 0)
            {
                wprintf(L"Directory: %s\\n", findFileData.cFileName);
            }
        }
        else
        {
            wprintf(L"File: %s\\n", findFileData.cFileName);
        }
    } while (FindNextFileW(hFind, &findFileData) != 0);

    FindClose(hFind);
}
```



Visual Studio Code interface showing the output of the program. The output window displays the following text:

```
[Running] cd "d:\Documents\Vs Code Projects\C language Coding Assignments\" &&
gcc 4RootDirectoryNamesAdv.c -o 4RootDirectoryNamesAdv && "d:\Documents\Vs Code
Projects\C language Coding Assignments\4RootDirectoryNamesAdv
Listing directories and files in D:\Documents:
Directory: Adobe
Directory: Adobe Ps
Directory: All My Custom App Presets
Directory: AndroidStudioProjects
Directory: Anto
Directory: Arduino
Directory: Blender Projects
Directory: Capcut Renders
Directory: Circuit Apps Projects
Directory: Cisco web
Directory: CNC
Directory: Control IO
Directory: Corel
Directory: Corel Cloud
Directory: CorelDRAW Projects
Directory: Custom Office Templates
File: Default.rdp
File: desktop.ini
File: Dir3.c
File: Dir3.exe
Directory: Factory IO
Directory: Fl new space
Directory: FluidSIM Hydraulics
Directory: FluidSIM Pneumatics
Directory: Fritzing
Directory: From AllSparkNouva
Directory: GitHubProjects
Directory: GNUSim 8085
Directory: Hacker
Directory: Image-Line
Directory: Inkspace
```



Visual Studio Code interface showing the output of the program. The output window displays the following text:

```
Directory: Inkspace
Directory: Kicad
Directory: LabVIEW Data
File: LCD Hello World .pdsprj
File: LCD Hello World .pdsprj.ALLSPARK.katchietsnouvaniccur.workspace
Directory: LennarDigital
Directory: Lyrics
Directory: Mbaire proteus projects
Directory: mqtt
Directory: Ms Office Activation
Directory: Msc
Directory: MsPresentation saves
Directory: MsWord Saves
Directory: Multism
Directory: My InTouch Applications
Directory: National Instruments
File: New Projectdel.pdsprj
File: New Projectdel.pdsprj.ALLSPARK.katchietsnouvaniccur.workspace
Directory: Notepad
Directory: Nouva Projects
File: NouvaTech.txt
Directory: OneNote Notebooks
File: Padding - Tailwind CSS.mhtml
Directory: pdf
File: Programming Languages.txt
Directory: Project Backups
Directory: Proteus Projects
Directory: Scan2cad Projects
Directory: Siemens TIA Automation
Directory: Solid works
Directory: Sound Recordings
Directory: Unity
Directory: Unity Projects
Directory: VirtualDJ
Directory: Visual Studio 2022
```

The following returned in a list form:

```
#include <stdio.h>
#include <windows.h>

void listDirectoriesAndFilesInRoot(const wchar_t *path);

int main()
{
    printf("Listing directories and files in D:\\Documents:\\n");
    listDirectoriesAndFilesInRoot(L"D:\\Documents");
    return 0;
}

void listDirectoriesAndFilesInRoot(const wchar_t *path)
{
    WIN32_FIND_DATAW findFileData;
    HANDLE hFind = FindFirstFileW((wchar_t *)L"D:\\Documents\\*", &findFileData);

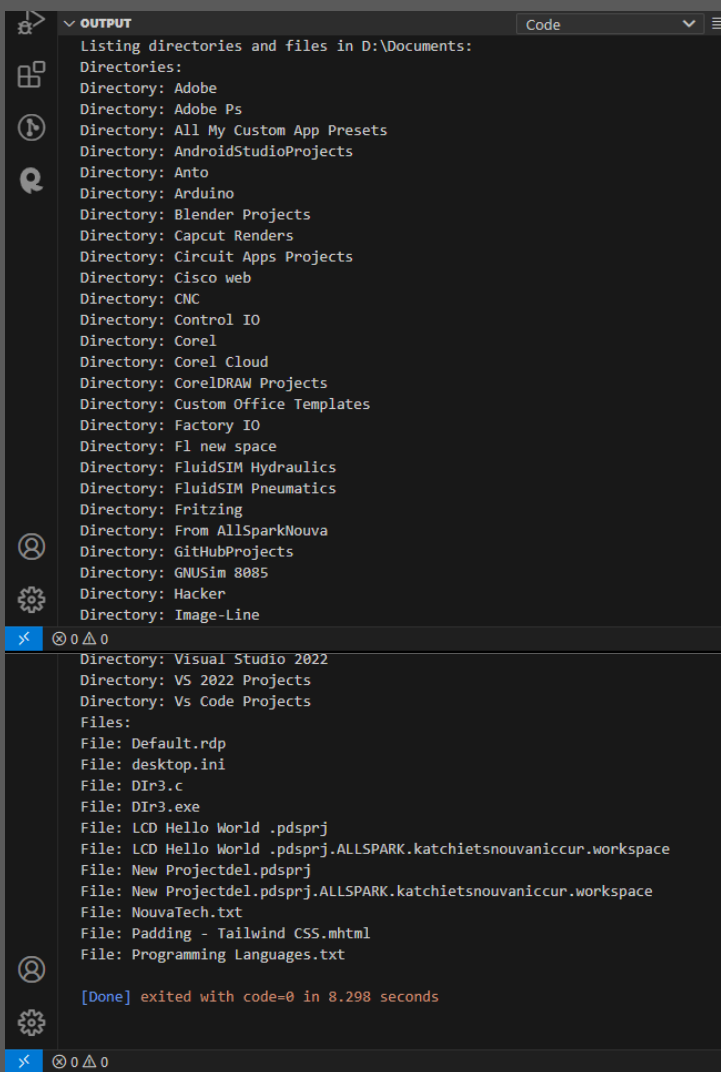
    if (hFind == INVALID_HANDLE_VALUE)
    {
        printf("Error finding directories and files in: %S\\n", path);
        return;
    }

    // Collect directories and non-directory files separately
    wprintf(L"Directories:\\n");
    do
    {
        if (findFileData.dwFileAttributes & FILE_ATTRIBUTE_DIRECTORY)
        {
            if (wcscmp(findFileData.cFileName, L"..") != 0 && wcscmp(findFileData.cFileName, L"..") != 0)
            {
                wprintf(L"Directory: %s\\n", findFileData.cFileName);
            }
        }
    } while (FindNextFileW(hFind, &findFileData) != 0);

    // Close the handle and reopen it to reset the search
    FindClose(hFind);
    hFind = FindFirstFileW((wchar_t *)L"D:\\Documents\\*", &findFileData);

    // Collect non-directory files
    wprintf(L"Files:\\n");
    do
    {
        if (!(findFileData.dwFileAttributes & FILE_ATTRIBUTE_DIRECTORY))
        {
            wprintf(L"File: %s\\n", findFileData.cFileName);
        }
    } while (FindNextFileW(hFind, &findFileData) != 0);

    FindClose(hFind);
}
```

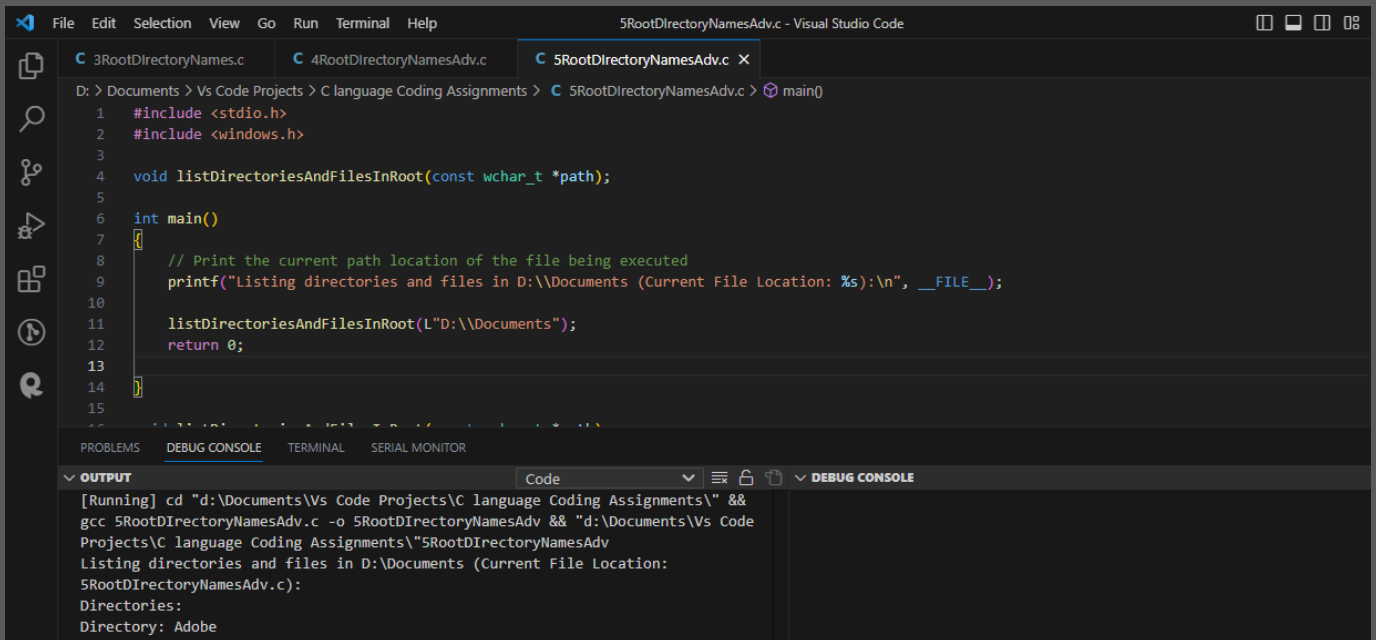


```
Listing directories and files in D:\Documents:
Directories:
Directory: Adobe
Directory: Adobe Ps
Directory: All My Custom App Presets
Directory: AndroidStudioProjects
Directory: Anto
Directory: Arduino
Directory: Blender Projects
Directory: Capcut Renders
Directory: Circuit Apps Projects
Directory: Cisco web
Directory: CNC
Directory: Control IO
Directory: Corel
Directory: Corel Cloud
Directory: CorelDRAW Projects
Directory: Custom Office Templates
Directory: Factory IO
Directory: Fl new space
Directory: FluidSIM Hydraulics
Directory: FluidSIM Pneumatics
Directory: Fritzing
Directory: From AllSparkNouva
Directory: GitHubProjects
Directory: GNUSim 8085
Directory: Hacker
Directory: Image-Line

Directory: Visual Studio 2022
Directory: VS 2022 Projects
Directory: Vs Code Projects
Files:
File: Default.rdp
File: desktop.ini
File: Dir3.c
File: Dir3.exe
File: LCD Hello World .pdsprj
File: LCD Hello World .pdsprj.ALLSPARK.katchietsnouvaniccur.workspace
File: New Projectdel.pdsprj
File: New Projectdel.pdsprj.ALLSPARK.katchietsnouvaniccur.workspace
File: NouvaTech.txt
File: Padding - Tailwind CSS.mhtml
File: Programming Languages.txt

[Done] exited with code=0 in 8.298 seconds
```

Retrieving code path:



```
File Edit Selection View Go Run Terminal Help
5RootDirectoryNamesAdv.c - Visual Studio Code

D: > Documents > Vs Code Projects > C language Coding Assignments > 5RootDirectoryNamesAdv.c > main()
1 #include <stdio.h>
2 #include <windows.h>
3
4 void listDirectoriesAndFilesInRoot(const wchar_t *path);
5
6 int main()
7 {
8     // Print the current path location of the file being executed
9     printf("Listing directories and files in D:\\Documents (Current File Location: %s):\\n", __FILE__);
10
11     listDirectoriesAndFilesInRoot(L"D:\\Documents");
12     return 0;
13 }
14
15

[Running] cd "d:\Documents\Vs Code Projects\C language Coding Assignments\" && gcc 5RootDirectoryNamesAdv.c -o 5RootDirectoryNamesAdv && "d:\Documents\Vs Code Projects\C language Coding Assignments\5RootDirectoryNamesAdv
Listing directories and files in D:\Documents (Current File Location: 5RootDirectoryNamesAdv.c):
Directories:
Directory: Adobe
```

Retrieving code path advanced v1:

FileEditSelectionViewGoRunTerminalHelp

5RootDirectoryNamesAdv.c - Visual Studio Code

C 5RootDirectoryNamesAdv.c X

D: > Documents > Vs Code Projects > C language Coding > C 5RootDirectoryNamesAdv.c > main()

```
1 #include <stdio.h>
2 #include <windows.h>
3 #include <string.h>
4 #include <wchar.h>
5
6 void listDirectoriesAndFilesInRoot(const wchar_t *path);
7
8 int main()
9 {
10     // Get the current file path by extracting the folder path
11     wchar_t currentFilePath[FILENAME_MAX];
12     GetModuleFileNameW(NULL, currentFilePath, FILENAME_MAX);
13
14     // Find the last backslash to remove the file name and get the folder path
15     wchar_t *lastBackslash = wcsrchr(currentFilePath, L'\\');
16     if (lastBackslash != NULL)
17     {
18         *lastBackslash = L'\0';
19     }
20
21     wprintf(L"Listing directories and files in D:\\Documents\\n(Current Code File Location: %ls):\\n", currentFilePath);
22
23     listDirectoriesAndFilesInRoot(L"D:\\Documents");
24     return 0;
25 }
```

PROBLEMS

DEBUG CONSOLE

TERMINAL

SERIAL MONITOR

OUTPUT

Code

```
[Running] cd "d:\Documents\Vs Code Projects\C language Coding\" && gcc
5RootDirectoryNamesAdv.c -o 5RootDirectoryNamesAdv && "d:\Documents\Vs Code
Projects\C language Coding\5RootDirectoryNamesAdv
Listing directories and files in D:\Documents
(Current Code File Location: d:\Documents\Vs Code Projects\C language Coding):
Directories:
Directory: Adobe
```

DEBUG CONSOLE

Ln 24, Col 14

Spaces: 4

UTF-8

Retrieving code path advanced v2:

The code was able to find:

- 1) Find the Current Code File Location of the parent directory:
(Current Code File Location: d:\Documents\Vs Code Projects\C language Coding)
- 2) Find the path of the executable file contained:
(Current Code File: d:\Documents\Vs Code Projects\C language Coding\5RootDirectoryNamesAdv.exe)
- 3) Provide a listing of directories and files in D:\Documents, and it 1st lists the directories then files, in alphabetical order e.g.,
 - i. Directories:
Directory: Adobe
Directory: Adobe Ps
Directory: All My Custom App Presets
Directory: Blender Projects
Directory: Unity Projects
 - ii. Files:
File: desktop.ini
File: Dir3.c
File: Dir3.exe
File: LCD Hello World .pdsprj

```
#include <stdio.h>
#include <windows.h>
#include <string.h>
#include <wchar.h>

void listDirectoriesAndFilesInRoot(const wchar_t *path);

int main()
{
    // Get the current executable file path
    wchar_t currentExecutablePath[FILENAME_MAX];
    GetModuleFileNameW(NULL, currentExecutablePath, FILENAME_MAX);

    // Find the last backslash to remove the file name and get the folder path
    wchar_t *lastBackslash = wcsrchr(currentExecutablePath, L'\\');
    if (lastBackslash != NULL)
    {
        *lastBackslash = L'\0';
    }

    // Get the current code file path
    wchar_t currentCodeFilePath[FILENAME_MAX];
    GetModuleFileNameW(NULL, currentCodeFilePath, FILENAME_MAX);
    // Print the current code file location and the current code file
    wprintf(L"Listing directories and files in D:\\Documents\\n(Current Code File Location: %ls)\\n(Current Code File: %ls):\\n",
currentExecutablePath, currentCodeFilePath);

    listDirectoriesAndFilesInRoot(L"D:\\Documents");
    return 0;
}

void listDirectoriesAndFilesInRoot(const wchar_t *path)
{
    WIN32_FIND_DATAW findFileData;
    HANDLE hFind = FindFirstFileW((wchar_t *)L"D:\\Documents\\*", &findFileData);

    if (hFind == INVALID_HANDLE_VALUE)
    {
        printf("Error finding directories and files in: %S\\n", path);
        return;
    }

    // Collect directories and non-directory files separately
    wprintf(L"Directories:\\n");
    do
    {
        if (findFileData.dwFileAttributes & FILE_ATTRIBUTE_DIRECTORY)
        {
            if (wcscmp(findFileData.cFileName, L"..") != 0 && wcscmp(findFileData.cFileName, L"..") != 0)
            {
                wprintf(L"Directory: %s\\n", findFileData.cFileName);
            }
        }
    } while (FindNextFileW(hFind, &findFileData) != 0);
```

```
// Close the handle and reopen it to reset the search
FindClose(hFind);
hFind = FindFirstFileW((wchar_t *)L"D:\\Documents\\*", &findFileData);

// Collect non-directory files
wprintf(L"Files:\n");
do
{
    if (!(findFileData.dwFileAttributes & FILE_ATTRIBUTE_DIRECTORY))
    {
        wprintf(L"File: %s\n", findFileData.cFileName);
    }
} while (FindNextFileW(hFind, &findFileData) != 0);

FindClose(hFind);
}
```

```
File Edit Selection View Go Run Terminal Help 5RootDirectoryName

PROBLEMS DEBUG CONSOLE TERMINAL SERIAL MONITOR
▼ OUTPUT Code
[Running] cd "d:\Documents\Vs Code Projects\C language Coding\" && gcc
5RootDirectoryNamesAdv.c -o 5RootDirectoryNamesAdv && "d:\Documents\Vs Code
Projects\C language Coding\5RootDirectoryNamesAdv
Listing directories and files in D:\Documents
(Current Code File Location: d:\Documents\Vs Code Projects\C language Coding)
(Current Code File: d:\Documents\Vs Code Projects\C language
Coding\5RootDirectoryNamesAdv.exe):
Directories:
Directory: Adobe
Directory: Adobe Ps
Directory: All My Custom App Presets
Directory: AndroidStudioProjects
Directory: Anto
Directory: Arduino
Directory: Blender Projects
Directory: Capcut Renders
Directory: Circuit Apps Projects
Directory: Cisco web
Directory: CNC
Directory: Control IO
Directory: Corel
Directory: Corel Cloud
Directory: CorelDRAW Projects
Directory: Custom Office Templates
Directory: Factory IO
Directory: Fl new space
Directory: FluidSIM Hydraulics
Directory: FluidSIM Pneumatics
Directory: Fritzing
Directory: From AllSparkNouva
Directory: GitHubProjects
Directory: GNUsim 8085
Directory: Hacker
Directory: Image-Line
Directory: Tokena
```

```
File Edit Selection View Go Run Terminal Help 5RootDirector

PROBLEMS DEBUG CONSOLE TERMINAL SERIAL MONITOR
▼ OUTPUT Code
Directory: musicism
Directory: My InTouch Applications
Directory: National Instruments
Directory: Notepad
Directory: Nouva Projects
Directory: OneNote Notebooks
Directory: pdf
Directory: Project Backups
Directory: Proteus Projects
Directory: Scan2cad Projects
Directory: Siemens TIA Automation
Directory: Solid works
Directory: Sound Recordings
Directory: Unity
Directory: Unity Projects
Directory: VirtualDJ
Directory: Visual Studio 2022
Directory: VS 2022 Projects
Directory: Vs Code Projects
Files:
File: Default.rdp
File: desktop.ini
File: Dir3.c
File: Dir3.exe
File: LCD Hello World .pdsprj
File: LCD Hello World .pdsprj.ALLSPARK.katchietsnouvaniccur.workspace
File: New Projectdel.pdsprj
File: New Projectdel.pdsprj.ALLSPARK.katchietsnouvaniccur.workspace
File: NouvaTech.txt
File: Padding - Tailwind CSS.mhtml
File: Programming Languages.txt

[Done] exited with code=0 in 8.198 seconds
```

The following code was able to add a bin folder: NB added in the int main() part only . however the file executable path wasn't obtained correctly

```
#include <stdio.h>
#include <windows.h>
#include <string.h>
#include <wchar.h>

void listDirectoriesAndFilesInRoot(const wchar_t *path);

int main()
{
    // Get the current executable file path
    wchar_t currentExecutablePath[FILENAME_MAX];
    GetModuleFileNameW(NULL, currentExecutablePath, FILENAME_MAX);

    // Find the last backslash to remove the file name and get the folder path
    wchar_t *lastBackslash = wcsrchr(currentExecutablePath, L'\\');
    if (lastBackslash != NULL)
    {
        *lastBackslash = L'\0';
    }

    // Create a copy of the current code file path before modifying it
    wchar_t currentCodeFilePath[FILENAME_MAX];
    wcsncpy(currentCodeFilePath, currentExecutablePath);

    // Get the current code file path
    wchar_t currentCodeFileFullPath[FILENAME_MAX];
    GetModuleFileNameW(NULL, currentCodeFileFullPath, FILENAME_MAX);

    // Remove the executable file name from the current code file path
    wchar_t *lastBackslashCodeFile = wcsrchr(currentCodeFileFullPath, L'\\');
    if (lastBackslashCodeFile != NULL)
    {
        *lastBackslashCodeFile = L'\0';
    }

    // Create the "bin" directory in the current code file path
    wcscat(currentCodeFilePath, L"\\bin");
    CreateDirectoryW(currentCodeFilePath, NULL);

    // Get just the file name from the current code file path
    wchar_t *fileName = wcsrchr(currentCodeFileFullPath, L'\\');
    if (fileName != NULL)
    {
        fileName++; // Move past the backslash to the actual file name
    }
    else
    {
        fileName = L"Unknown"; // Fallback in case of any issue with the file name
    }

    // Print the current code file location and the current code file
    wprintf(L"Listing directories and files in D:\\Documents\\n(Current Code File Location: %ls)\\n(Current Code File: %ls)\\n(Current Code Bin File Location: %ls)\\n", currentExecutablePath, currentCodeFileFullPath, currentCodeFilePath);

    listDirectoriesAndFilesInRoot(L"D:\\Documents");
    return 0;
}
```

Listing directories and files in D:\\Documents

(Current Code File Location: d:\\Documents\\Vs Code Projects\\C language Coding)

(Current Code File: d:\\Documents\\Vs Code Projects\\C language Coding)

(Current Code Bin File Location: d:\\Documents\\Vs Code Projects\\C language Coding\\bin):

This was able to return the right output:

```
#include <stdio.h>
#include <windows.h>
#include <string.h>
#include <wchar.h>

void listDirectoriesAndFilesInRoot(const wchar_t *path);

int main()
{
    // Get the current executable file path
    wchar_t currentExecutablePath[FILENAME_MAX];
    GetModuleFileNameW(NULL, currentExecutablePath, FILENAME_MAX);

    // Find the last backslash to remove the file name and get the folder path
    wchar_t *lastBackslash = wcsrchr(currentExecutablePath, L'\\');
    if (lastBackslash != NULL)
    {
        *lastBackslash = L'\0';
    }

    // Get just the file name from the current executable path
    wchar_t *fileName = wcsrchr(currentExecutablePath, L'\\');
    if (fileName != NULL)
    {
        fileName++; // Move past the backslash to the actual file name
    }
    else
    {
        fileName = L"Unknown"; // Fallback in case of any issue with the file name
    }

    // Create a copy of the current code file path before modifying it
    wchar_t currentCodeFilePath[FILENAME_MAX];
    wcsncpy(currentCodeFilePath, currentExecutablePath);

    // Create the "bin" directory in the current code file path
    wcscat(currentCodeFilePath, L"\\bin");
    CreateDirectoryW(currentCodeFilePath, NULL);

    // Get the current code file name with .exe extension
    wchar_t currentCodeFileName[FILENAME_MAX];
    GetModuleFileNameW(NULL, currentCodeFileName, FILENAME_MAX);

    // Print the current code file location and the current code file
    wprintf(L"Listing directories and files in D:\\Documents\\n(Current Code File Location: %ls)\\n(Current Code File: %ls)\\n(Current Code Bin File Location: %ls):\\n", currentExecutablePath, currentCodeFileName, currentCodeFilePath);

    listDirectoriesAndFilesInRoot(L"D:\\Documents");
    return 0;
}

void listDirectoriesAndFilesInRoot(const wchar_t *path)
{
    WIN32_FIND_DATAW findFileData;
    HANDLE hFind = FindFirstFileW((wchar_t *)L"D:\\Documents\\*", &findFileData);

    if (hFind == INVALID_HANDLE_VALUE)
    {
        printf("Error finding directories and files in: %S\\n", path);
        return;
    }

    // Collect directories and non-directory files separately
    wprintf(L"Directories:\\n");
    do
    {
        {
            if (findFileData.dwFileAttributes & FILE_ATTRIBUTE_DIRECTORY)
            {
                {
                    if (wcscmp(findFileData.cFileName, L".") != 0 && wcscmp(findFileData.cFileName, L"..") != 0)
                    {
                        wprintf(L"Directory: %s\\n", findFileData.cFileName);
                    }
                }
            }
        } while (FindNextFileW(hFind, &findFileData) != 0);

    // Close the handle and reopen it to reset the search
```

```

FindClose(hFind);
hFind = FindFirstFileW((wchar_t *)L"D:\\Documents\\*", &findFileData);

// Collect non-directory files
wprintf(L"Files:\n");
do
{
    if (!(findFileData.dwFileAttributes & FILE_ATTRIBUTE_DIRECTORY))
    {
        wprintf(L"File: %s\n", findFileData.cFileName);
    }
} while (FindNextFileW(hFind, &findFileData) != 0);

FindClose(hFind);
}

```

6RootDirectoryNamesAdv_CreateFolder.c - Visual Studio Code

PROBLEMS DEBUG CONSOLE TERMINAL SERIAL MONITOR

Code

DEBUG CONSOLE

Filter (e.g. text, lexclude)

OUTPUT

```

Directory: nullism
Directory: My InTouch Applications
Directory: National Instruments
Directory: Notepad
Directory: Nouva Projects
Directory: OneNote Notebooks
Directory: pdf
Directory: Project Backups
Directory: Proteus Projects
Directory: Scan2cad Projects
Directory: Siemens TIA Automation
Directory: Solid works
Directory: Sound Recordings
Directory: Unity
Directory: Unity Projects
Directory: VirtualDJ
Directory: Visual Studio 2022
Directory: VS 2022 Projects
Directory: Vs Code Projects
Files:
File: Default.rdp
File: desktop.ini
File: Dir3.c
File: Dir3.exe
File: LCD Hello World .pdsprj
File: LCD Hello World .pdsprj.ALLSPARK.katchietsnouvaniccur.workspace
File: New Projectdel.pdsprj
File: New Projectdel.pdsprj.ALLSPARK.katchietsnouvaniccur.workspace
File: NouvaTech.txt
File: Padding - Tailwind CSS.mhtml
File: Programming Languages.txt

[Done] exited with code=0 in 8.999 seconds

```

Ln 91, Col 1 (3075 selected) Spaces: 4 UTF-8 CRLF {} C

6RootDirectoryNamesAdv_CreateFolder.c - Visual Studio Code

PROBLEMS DEBUG CONSOLE TERMINAL SERIAL MONITOR

Code

DEBUG CONSOLE

Filter (e.g. text, lexclude)

OUTPUT

```

[Running] cd "d:\Documents\Vs Code Projects\C language Coding\" && gcc 6RootDirectoryNamesAdv_CreateFolder.c -o
6RootDirectoryNamesAdv_CreateFolder && "d:\Documents\Vs Code Projects\C language Coding\6RootDirectoryNamesAdv_CreateFolder
Listing directories and files in D:\Documents
(Current Code File Location: d:\Documents\Vs Code Projects\C language Coding)
(Current Code File: d:\Documents\Vs Code Projects\C language Coding\6RootDirectoryNamesAdv_CreateFolder.exe)
(Current Code Bin File Location: d:\Documents\Vs Code Projects\C language Coding\bin):
Directories:
Directory: Adobe
Directory: Adobe Ps
Directory: All My Custom App Presets
Directory: AndroidStudioProjects
Directory: Anto
Directory: Arduino
Directory: Blender Projects
Directory: Capcut Renders
Directory: Circuit Apps Projects
Directory: Cisco web
Directory: CNC
Directory: Control IO
Directory: Corel
Directory: Corel Cloud
Directory: CorelDRAW Projects
Directory: Custom Office Templates
Directory: Factory IO
Directory: Fl new space
Directory: FluidSIM Hydraulics
Directory: FluidSIM Pneumatics
Directory: Fritzing
Directory: From AllSparkNouva
Directory: GitHubProjects
Directory: GNUSim 8085
Directory: Hacker
Directory: Image-Line
Directory: Inkspace
Directory: Kicad

```

Ln 91, Col 1 (3075 selected) Spaces: 4 UTF-8 CRLF {} C

****The following code didn't create a bin folder and a 1.txt file inside it. But it ran successfully after multiple tries. It also removed some functionalities**

```
#include <stdio.h>
#include <windows.h>
#include <string.h>

void listDirectoriesAndFilesInRoot(const wchar_t *path);

int main()
{
    wchar_t currentCodeFilePath[FILENAME_MAX];
    GetModuleFileNameW(NULL, currentCodeFilePath, FILENAME_MAX);
    const wchar_t *lastBackslash = wcsrchr(currentCodeFilePath, L'\\');
    currentCodeFilePath[lastBackslash - currentCodeFilePath + 1] = L'0';

    wprintf(L"Listing directories and files in D:\\Documents\\n(Current Code File Location: %ls)\\n", currentCodeFilePath);

    listDirectoriesAndFilesInRoot(L"D:\\Documents");
    return 0;
}

void listDirectoriesAndFilesInRoot(const wchar_t *path)
{
    WIN32_FIND_DATAW findFileData;
    HANDLE hFind = FindFirstFileW((wchar_t *)L"D:\\Documents\\*", &findFileData);

    if (hFind == INVALID_HANDLE_VALUE)
    {
        wprintf(L"Error finding directories and files in: %ls\\n", path);
        return;
    }

    // Collect directories and non-directory files separately
    wprintf(L"Directories:\\n");
    do
    {
        {
            if (findFileData.dwFileAttributes & FILE_ATTRIBUTE_DIRECTORY)
            {
                if (wcscmp(findFileData.cFileName, L".") != 0 && wcscmp(findFileData.cFileName, L"..") != 0)
                {
                    wprintf(L"Directory: %ls\\n", findFileData.cFileName);
                }
            }
        }
    } while (FindNextFileW(hFind, &findFileData) != 0);

    // Close the handle and reopen it to reset the search
    FindClose(hFind);
    hFind = FindFirstFileW((wchar_t *)L"D:\\Documents\\*", &findFileData);

    // Collect non-directory files
    wprintf(L"Files:\\n");
    do
    {
        {
            if (!(findFileData.dwFileAttributes & FILE_ATTRIBUTE_DIRECTORY))
            {
                wprintf(L"File: %ls\\n", findFileData.cFileName);
            }
        }
    } while (FindNextFileW(hFind, &findFileData) != 0);

    FindClose(hFind);

    // Create a new directory called "bin" in the current code file location
    wchar_t binDirectory[FILENAME_MAX];
    _snwprintf(binDirectory, FILENAME_MAX, L"%ls\\bin", path);
    CreateDirectoryW(binDirectory, NULL);

    // Create and log to a text file in the "bin" directory
    wchar_t logFileName[FILENAME_MAX];
    int suffix = 1;
    wchar_t suffixString[10];
    _snwprintf(suffixString, 10, L"%d", suffix);
    _snwprintf(logFileName, FILENAME_MAX, L"%ls\\log_%ls.txt", path, suffixString);

    FILE *logFile = _wfopen(logFileName, L"w");
```

```

if (logFile)
{
    fprintf(logFile, L"This is a sample log entry.\n");
    fclose(logFile);
}
}

```

C Code Request

New chat

Today

Adobe

C Code Request

Yesterday

Arduino AI From Scratch

Electron Chirality Explained

Friend's Workplace Job Applic

Database Exception (HYT00)

Previous 7 Days

Force Transmission Analysis

4000 YHz Hypothetical Frequ

Love Song Chord Progression

Upgrade to Plus

chat.openai.com/c/7703556d-852b-40a4-a9fb-1ea82f6daf49

Web3

Rob

DDF

YT

Lyrics

Cloud

SQL

Socials

C

H

GPT

Other bo

[Running] cd "d:\Documents\Vs Code Projects\C language Coding\" && gcc 7RootDirectoryNamesAdv_CreateFolder_File.c -o 7RootDirectoryNamesAdv_CreateFolder_File && "d:\Documents\Vs Code Projects\C language Coding\7RootDirectoryNamesAdv_CreateFolder_File

Listing directories and files in D:\Documents

(Current Code File Location: d:\Documents\Vs Code Projects\C language Coding\)

Directories:

Directory: Adobe

Directory: Adobe Ps

Directory: All My Custom App Presets

Directory: AndroidStudioProjects

Directory: Anto

Directory: Arduino

Directory: bin

Directory: Blender Projects

Directory: Capcut Renders

Directory: Circuit Apps Projects

Directory: Cisco web

Directory: CNC

Directory: Control IO

Directory: Corel

Directory: Corel Cloud

Directory: CorelDRAW Projects

Regenerate response

Send a message

Battery status: 98% remaining

Today

Adobe

C Code Request

Yesterday

Arduino AI From Scratch

Electron Chirality Explained

Friend's Workplace Job Applic

Database Exception (HYT00)

Previous 7 Days

it didnt create a new directory bin such that a path D:\Documents\Vs Code Projects\C language Coding\bin exists and second if it had created bin folder, there woud be a text file inside it which the log would have appeared. i created a bin folder and rerun the code but there was no text file in it.

so let us solve the problem stepwise. since i already have created a bin folder such that the path D:\Documents\Vs Code Projects\C language Coding\bin exists, i have created a file inside it such that the path D:\Documents\Vs Code Projects\C language Coding\bin\1.txt exists. so your task is simple.

i need the result in the output to be logged into the 1.txt file

The following code was able to write into D:\Documents\bin\1.txt. the text This is a sample log entry

```
#include <stdio.h>
#include <windows.h>
#include <string.h>

void listDirectoriesAndFilesInRoot(const wchar_t *path);

int main()
{
    wchar_t currentCodeFilePath[FILENAME_MAX];
    GetModuleFileNameW(NULL, currentCodeFilePath, FILENAME_MAX);
    const wchar_t *lastBackslash = wcsrchr(currentCodeFilePath, L '\\');
    currentCodeFilePath[lastBackslash - currentCodeFilePath + 1] = L '\\0';

    wprintf(L"Listing directories and files in D:\\Documents\\n(Current Code File Location: %ls)\\n", currentCodeFilePath);

    listDirectoriesAndFilesInRoot(L"D:\\Documents");
    return 0;
}

void listDirectoriesAndFilesInRoot(const wchar_t *path)
{
    WIN32_FIND_DATAW findFileData;
    HANDLE hFind = FindFirstFileW((wchar_t *)L"D:\\Documents\\*", &findFileData);

    if (hFind == INVALID_HANDLE_VALUE)
    {
        wprintf(L"Error finding directories and files in: %ls\\n", path);
        return;
    }

    // Collect directories and non-directory files separately
    wprintf(L"Directories:\\n");
    do
    {
        if (findFileData.dwFileAttributes & FILE_ATTRIBUTE_DIRECTORY)
        {
            if (wcscmp(findFileData.cFileName, L".") != 0 && wcscmp(findFileData.cFileName, L"..") != 0)
            {
                wprintf(L"Directory: %ls\\n", findFileData.cFileName);
            }
        }
    } while (FindNextFileW(hFind, &findFileData) != 0);

    // Close the handle and reopen it to reset the search
    FindClose(hFind);
    hFind = FindFirstFileW((wchar_t *)L"D:\\Documents\\*", &findFileData);

    // Collect non-directory files
    wprintf(L"Files:\\n");
    do
    {
        if (!(findFileData.dwFileAttributes & FILE_ATTRIBUTE_DIRECTORY))
        {
            wprintf(L"File: %ls\\n", findFileData.cFileName);
        }
    } while (FindNextFileW(hFind, &findFileData) != 0);

    FindClose(hFind);

    // Create a new directory called "bin" in the current code file location
    wchar_t binDirectory[FILENAME_MAX];
    _snwprintf(binDirectory, FILENAME_MAX, L"%ls\\bin", path);
    CreateDirectoryW(binDirectory, NULL);

    // Create and log to a text file in the "bin" directory
    wchar_t logFileName[FILENAME_MAX];
    _snwprintf(logFileName, FILENAME_MAX, L"%ls\\bin\\1.txt", path);

    FILE *logFile = _wfopen(logFileName, L"w");
    if (logFile)
    {
        fwprintf(logFile, L"This is a sample log entry.\\n");
        fclose(logFile);
        wprintf(L"Successfully logged to %ls.\\n", logFileName);
    }
}
```

```

}
else
{
    wprintf(L"Failed to open or write to %ls.\n", logFileName);
}
}

```

The screenshot shows the Visual Studio Code editor with a C program file named `8RootDirectoryNamesAdv_CreateFolder_File_Logging.c`. The code is as follows:

```

72     fprintf(logFile, L"This is a sample log entry.\n");
73     fclose(logFile);
74     wprintf(L"Successfully logged to %ls.\n",
75            logFileName);
76 }
77 else
78 {
79     wprintf(L"Failed to open or write to %ls.\n",
80            logFileName);
81 }

```

Below the editor, the **DEBUG CONSOLE** is open, showing the output of the program:

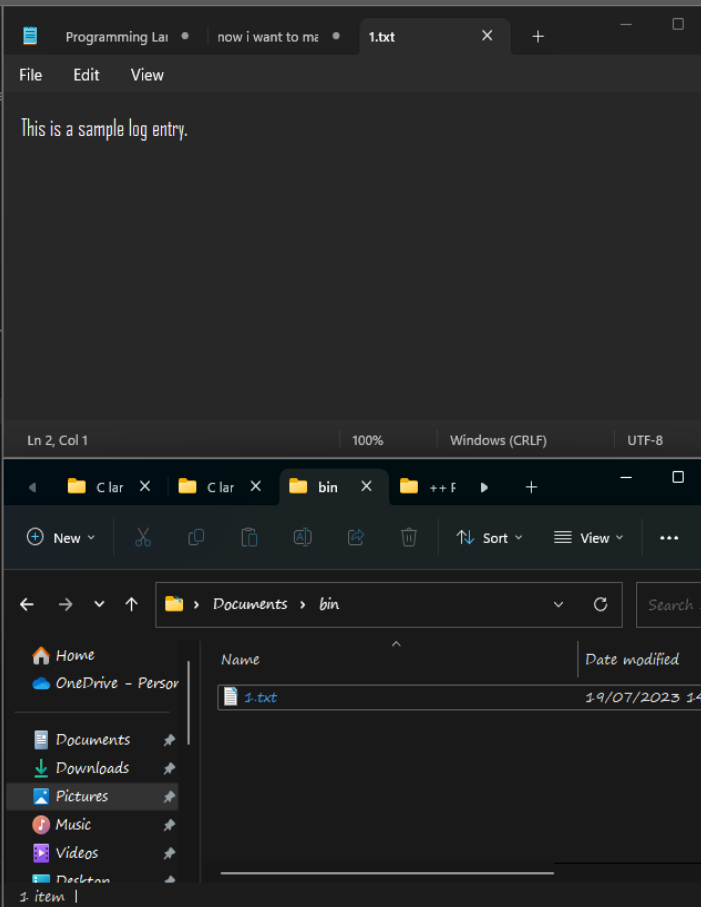
```

File: Default.rdp
File: desktop.ini
File: Dir3.c
File: Dir3.exe
File: LCD Hello World .pdsprj
File: LCD Hello World .pdsprj.ALLSPARK.
katchietsnouvaniccur.workspace
File: log_1.txt
File: log_r.txt
File: New Projectdel.pdsprj
File: New Projectdel.pdsprj.ALLSPARK.
katchietsnouvaniccur.workspace
File: NouvaTech.txt
File: Padding - Tailwind CSS.mhtml
File: Programming Languages.txt
Successfully logged to D:\Documents\bin\1.txt.

[Done] exited with code=0 in 9.906 seconds

```

The status bar at the bottom indicates the file is at line 81, column 1, with 4 spaces, UTF-8 encoding, CRLF line endings, and Win32 architecture.



The code below is a symbol of advancement. It was able to:

1. Create a new folder in the path D:\Documents in case it was non existent
2. Create a new file 1.txt incase it was non existent
3. Log in data into the text file.
4. Incase there was data in the log file it was able to append additional data into the log file.

```
#include <stdio.h>
#include <windows.h>
#include <string.h>

void listDirectoriesAndFilesInRoot(const wchar_t *path);

int main()
{
    wchar_t currentCodeFilePath[FILENAME_MAX];
    GetModuleFileNameW(NULL, currentCodeFilePath, FILENAME_MAX);
    const wchar_t *lastBackslash = wcsrchr(currentCodeFilePath, L'\\');
    currentCodeFilePath[lastBackslash - currentCodeFilePath + 1] = L'\\0';

    wprintf(L"Listing directories and files in D:\\Documents\\n(Current Code File Location: %ls)\\n",
currentCodeFilePath);

    listDirectoriesAndFilesInRoot(L"D:\\Documents");
    return 0;
}

void listDirectoriesAndFilesInRoot(const wchar_t *path)
{
    WIN32_FIND_DATAW findFileData;
    HANDLE hFind = FindFirstFileW((wchar_t *)L"D:\\Documents\\*", &findFileData);

    if (hFind == INVALID_HANDLE_VALUE)
    {
        wprintf(L"Error finding directories and files in: %ls\\n", path);
        return;
    }

    // ... (Code to list directories and files remains unchanged)

    // Create the "bin" directory if it doesn't exist
    wchar_t binDirectory[FILENAME_MAX];
    _snwprintf(binDirectory, FILENAME_MAX, L"%ls\\bin", path);
    CreateDirectoryW(binDirectory, NULL);

    // Create and log to a text file in the "bin" directory
    wchar_t logFileName[FILENAME_MAX];
    _snwprintf(logFileName, FILENAME_MAX, L"%ls\\bin\\1.txt", path);

    FILE *logFile = _wfopen(logFileName, L"a"); // Open in "append" mode instead of "write" mode
    if (logFile)
    {
        fwprintf(logFile, L"This is a sample log entry.\\n");
        fwprintf(logFile, L"Additional log entry 1.\\n");
        fwprintf(logFile, L"Additional log entry 2.\\n");

        fclose(logFile);
        wprintf(L"Successfully logged to %ls.\\n", logFileName);
    }
}
```

```

// Read and print the contents of the log file to the console
wprintf(L"\nLogged Contents:\n");
FILE *readLogFile = _wopen(logFileName, L"r");
if (readLogFile)
{
    wchar_t buffer[512];
    while (fgetws(buffer, 512, readLogFile))
    {
        wprintf(L"%ls", buffer);
    }
    fclose(readLogFile);
}
else
{
    wprintf(L"Failed to read %ls.\n", logFileName);
}
}
else
{
    wprintf(L"Failed to open or write to %ls.\n", logFileName);
}

// ... (The rest of the code remains unchanged)
}

```

The screenshot shows the Visual Studio Code editor with a C++ file named `9RootDirectoryNamesAdv_CreateFolder_File_LoggingAppend.c` open. The file contains the following code:

```

1 #include <stdio.h>
2 #include <windows.h>
3 #include <string.h>
4
5 void listDirectoriesAndFilesInRoot(const wchar_t *path);
6
7 int main()
8 {
9     wchar_t currentCodeFilePath[FILENAME_MAX];

```

The **OUTPUT** window shows the following text:

```

language
Coding\9RootDirectoryNamesAdv_CreateFolder_File_LoggingAppend
Listing directories and files in D:\Documents
(Current Code File Location: d:\Documents\Vs Code Projects\C language Coding\
Successfully logged to D:\Documents\bin\1.txt.

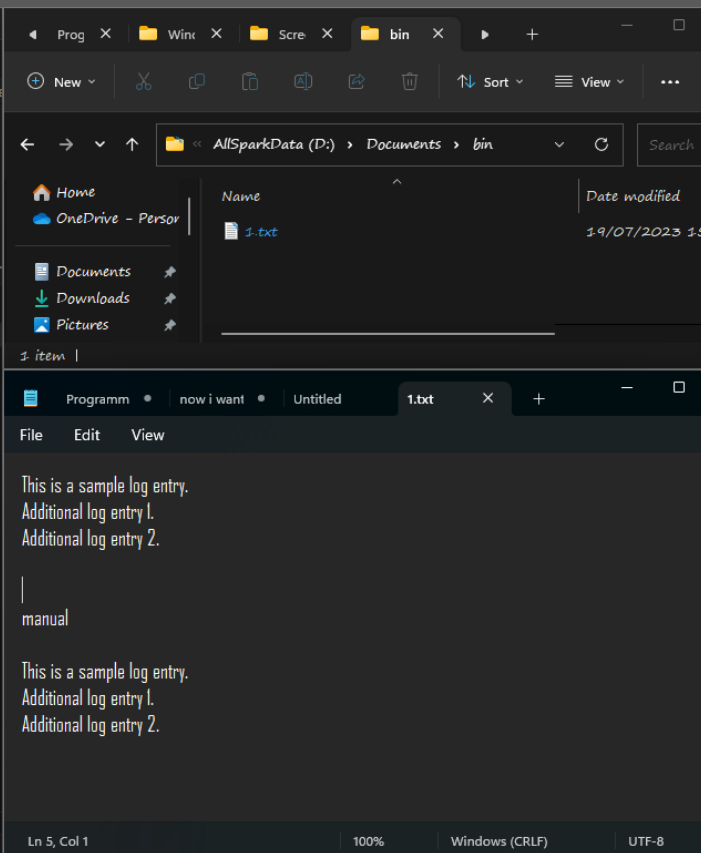
Logged Contents:
This is a sample log entry.
Additional log entry 1.
Additional log entry 2.

manual

This is a sample log entry.
Additional log entry 1.
Additional log entry 2.

[Done] exited with code=0 in 10.279 seconds

```



The code below is a symbol of Super advancement. It was able to:

1. Check if the bin folder exists in the root of the installation folder
2. If not, it will create it before attempting to open the log file (This should prevent the "Failed to open or write to" error from occurring when the folder does not exist.)
3. Create a new file 1.txt in case it was nonexistent.
4. Write the root contents of path D:\Documents into the log file in installation folder\bin\1.txt
5. If the log file already exists, it will append the data to the already existing contents.
6. I successfully copied it to another drive and could do step 1 through to 5 without problem

```

1  #include <stdio.h>
2  #include <windows.h>
3  #include <string.h>
4
5  // Function prototype declaration
6  void listDirectoriesAndFilesInRoot(const wchar_t *path, FILE *logFile);
7
8  int main()
9  {
10     // Variable to store the path of the current code file
11     wchar_t currentCodeFilePath[FILENAME_MAX];
12     // Get the path of the current executable (the code file)
13     GetModuleFileNameW(NULL, currentCodeFilePath, FILENAME_MAX);
14     // Extract the directory path by removing the file name from the path
15     const wchar_t *lastBackslash = wcsrchr(currentCodeFilePath, L'\\');
16     currentCodeFilePath[lastBackslash - currentCodeFilePath + 1] = L'\\0';
17
18     // Print the current directory and the code file's location
19     wprintf(L"Listing directories and files in D:\\Documents\\n(Current Code File Location: %ls)\\n",
currentCodeFilePath);
20
21     // Variable to store the log file name
22     wchar_t logFileName[FILENAME_MAX];
23     // Create the log file name using the current directory path
24     _snwprintf(logFileName, FILENAME_MAX, L"%lsbin\\1.txt", currentCodeFilePath);
25
26     // Check if the 'bin' folder exists, if not, create it
27     wchar_t binFolderPath[FILENAME_MAX];
28     _snwprintf(binFolderPath, FILENAME_MAX, L"%lsbin", currentCodeFilePath);
29     if (!CreateDirectoryW(binFolderPath, NULL))
30     {
31         DWORD error = GetLastError();
32         if (error != ERROR_ALREADY_EXISTS)
33         {
34             // Print an error message if the folder creation fails
35             wprintf(L"Failed to create 'bin' folder: %ls\\n", binFolderPath);
36             return 1; // Return with an error code (1) to indicate failure
37         }
38     }
39
40     // Open the log file in "append" mode (add data to the existing file)
41     FILE *logFile = _wfopen(logFileName, L"a");
42     if (logFile)
43     {
44         // Call the function to list directories and files in the specified path
45         listDirectoriesAndFilesInRoot(L"D:\\Documents", logFile);
46
47         // Close the log file after writing the data
48         fclose(logFile);

```

```

49     // Print a success message with the log file name
50     wprintf(L"Successfully logged to %ls.\n", logFileName);
51
52     // Read and print the contents of the log file to the console
53     wprintf(L"\nLogged Contents:\n");
54     FILE *readLogFile = _wfopen(logFileName, L"r");
55     if (readLogFile)
56     {
57         wchar_t buffer[512];
58         while (fgetws(buffer, 512, readLogFile))
59         {
60             wprintf(L"%ls", buffer);
61         }
62         fclose(readLogFile);
63     }
64     else
65     {
66         // Print an error message if reading the log file fails
67         wprintf(L"Failed to read %ls.\n", logFileName);
68     }
69 }
70 else
71 {
72     // Print an error message if opening or writing to the log file fails
73     wprintf(L"Failed to open or write to %ls.\n", logFileName);
74 }
75
76 return 0; // Return with a success code (0) to indicate successful execution
77 }
78
79 // Function definition to list directories and files in the specified path
80 void listDirectoriesAndFilesInRoot(const wchar_t *path, FILE *logFile)
81 {
82     // Variables for handling file enumeration in the specified path
83     WIN32_FIND_DATAW findFileData;
84     HANDLE hFind = FindFirstFileW((wchar_t *)L"D:\\Documents\\*", &findFileData);
85
86     // Check if file enumeration is successful or not
87     if (hFind == INVALID_HANDLE_VALUE)
88     {
89         // Print an error message if enumeration fails
90         wprintf(L"Error finding directories and files in: %ls\n", path);
91         return; // Return from the function
92     }
93
94     // Collect directories and non-directory files separately
95     wprintf(L"Directories:\n");
96     do
97     {
98         // Check if the current item is a directory
99         if (findFileData.dwFileAttributes & FILE_ATTRIBUTE_DIRECTORY)
100         {
101             // Exclude '.' and '..' directories from the listing
102             if (wcscmp(findFileData.cFileName, L".") != 0 && wcscmp(findFileData.cFileName, L"..")
103 != 0)
104             {
105                 // Print the directory name to the console

```

```

105         wprintf(L"Directory: %ls\n", findFileData.cFileName);
106         // Log the directory name to the log file
107         fwprintf(logFile, L"Directory: %ls\n", findFileData.cFileName);
108     }
109 }
110 } while (FindNextFileW(hFind, &findFileData) != 0);
111
112 // Close the handle and reopen it to reset the search
113 FindClose(hFind);
114 hFind = FindFirstFileW((wchar_t *)L"D:\\Documents\\*", &findFileData);
115
116 // Collect non-directory files
117 wprintf(L"Files:\n");
118 do
119 {
120     // Check if the current item is not a directory (i.e., a file)
121     if (!(findFileData.dwFileAttributes & FILE_ATTRIBUTE_DIRECTORY))
122     {
123         // Print the file name to the console
124         wprintf(L"File: %ls\n", findFileData.cFileName);
125         // Log the file name to the log file
126         fwprintf(logFile, L"File: %ls\n", findFileData.cFileName);
127     }
128 } while (FindNextFileW(hFind, &findFileData) != 0);
129
130 // Close the handle to release resources
131 FindClose(hFind);
132 }
133

```

The screenshot displays the Visual Studio Code interface during the execution of a C program. The editor shows the source code of `10Compilation.c`, which includes a loop to find files and directories in the `D:\Documents` folder. The `OUTPUT` pane at the bottom left shows the program's output, including the file names and the directory names. The right pane shows the list of directories found, which includes `1bin`, `2bin`, `Adobe`, `Adobe Ps`, `All My Custom App Presets`, `AndroidStudioProjects`, `Anto`, `Arduino`, `bin`, `Blender Projects`, `Capcut Renders`, `Circuit Apps Projects`, `Cisco web`, `CNC`, `Control IO`, `Corel`, `Corel Cloud`, `CorelDRAW Projects`, `Custom Office Templates`, `Factory IO`, `FI new space`, `FluidSIM Hydraulics`, `FluidSIM Pneumatics`, `Fritzing`, `From AllSparkNouva`, and `GitHubPniarts`.

```

katchietsnouvaniccur.workspace
File: NouvaTech.txt
File: Padding - Tailwind CSS.mhtml
File: Programming Languages.txt
Successfully logged to d:\Documents\Vs Code
Projects\C language Coding\bin\1.txt.

Logged Contents:
Directory: 1bin
Directory: 2bin
Directory: Adobe
Directory: Adobe Ps
Directory: All My Custom App Presets

```

To change the path you want to look into (e.g., `D:\dukto` instead of `D:\Documents`), you need to modify two parts of the code:

1. In the `main()` function:

```
1 int main()
2 {
3     // ...
4     // Replace the path "D:\\Documents" with "D:\\dukto"
5     listDirectoriesAndFilesInRoot(L"D:\\dukto", logFile);
6     // ...
7 }
8
```

In the `listDirectoriesAndFilesInRoot` function:

```
1 void listDirectoriesAndFilesInRoot(const wchar_t *path, FILE *logFile)
2 {
3     // ...
4     // Replace the path "D:\\Documents\\" with the new path "D:\\dukto\\"
5     HANDLE hFind = FindFirstFileW((wchar_t *)L"D:\\dukto\\", &findFileData);
6     // ...
7 }
```

After making these changes, the program will list directories and files in the specified path (`D:\dukto`) instead of `D:\Documents`.

Remember to ensure that the specified path (`D:\dukto`) exists and contains the files and directories you want to list; otherwise, the program might not produce the expected results.

I wanted to access the default folders. Now suppose that normally in every computer, by default there is a downloads, document, videos, pictures, music and desktop folders. In most operating systems, especially Windows, macOS, and Linux distributions, there are standard default folders commonly found in the user's home directory. These folders are created by the system to organize specific types of files and are commonly known as "special folders" or "user folders."

The standard default folders you mentioned are commonly found in many systems:

1. **Downloads:** This folder is intended to store downloaded files from the internet or other sources.
2. **Documents:** It's used to store user documents and other text-based files.
3. **Videos:** This folder is meant to hold video files.
4. **Pictures:** It's used to store image files and photos.
5. **Music:** This folder is meant to hold audio files, like music tracks.
6. **Desktop:** It's used to store files and shortcuts that are displayed on the user's desktop.

The actual names of these folders might vary slightly depending on the operating system and its language settings. For example, on some systems, "Documents" might be called "My Documents" or "Documents" in the native language.

These folders are intended to help users organize their files conveniently, and most applications typically save and access user-specific files from these standard folders to promote a consistent user experience.

It's important to note that while these default folders are commonly present in most systems, users can still create, rename, or delete folders to suit their preferences or organizational needs. Additionally, the availability and names of these folders can differ in various operating systems and versions.

To access the standard default folders without using hardcoded paths, you can use platform-specific functions that provide the paths to these special folders programmatically. In Windows, you can use the SHGetFolderPath function to get the paths to special folders. On macOS, you can use NSSearchPathForDirectoriesInDomains, and on Linux, you can use XDG Base Directory Specification.

Here's how you can modify the code to access the "Downloads" folder on Windows using SHGetFolderPath

Added the #include <shlobj.h> header to include the necessary declarations for using the SHGetFolderPath function.

Modified the main() function to use SHGetFolderPath to get the path of the "Downloads" folder:

The SHGetFolderPath function retrieves the path of a special folder identified by a CSIDL (Constant Special Item ID List) value. In this case, CSIDL_PERSONAL is used, which represents the "My Documents" folder on Windows, where the "Downloads" folder is often located.

After retrieving the path using SHGetFolderPath, the code appends "\\Downloads" to the path to form the full path to the "Downloads" folder. The rest of the code then uses this dynamically obtained path to list directories and files in the "Downloads" folder.

Using this approach allows the code to access the "Downloads" folder without relying on hardcoded paths, making it more flexible and portable across different systems and users.

```
// ... Your existing function implementation remains unchanged ...  
From code number 57
```

```
1. #include <stdio.h>  
2. #include <windows.h>  
3. #include <shlobj.h> // Include the header for SHGetFolderPath  
4.  
5. void listDirectoriesAndFilesInRoot(const wchar_t *path, FILE *logFile);  
6.  
7. int main()  
8. {  
9.     // Variable to store the path of the Downloads folder  
10.    wchar_t downloadsFolderPath[MAX_PATH];  
11.  
12.    // Get the path of the Downloads folder using SHGetFolderPath  
13.    if (SUCCEEDED(SHGetFolderPathW(NULL, CSIDL_PERSONAL, NULL, 0, downloadsFolderPath)))  
14.    {  
15.        // Append "\\Downloads" to the path to get the full Downloads folder path  
16.        wscat(downloadsFolderPath, L"\\Downloads");  
17.
```

```

18.     // Open the log file in "append" mode (add data to the existing file)
19.     FILE *logFile = _wopen(L"log.txt", L"a");
20.     if (logFile)
21.     {
22.         // Call the function to list directories and files in the specified path
23.         listDirectoriesAndFilesInRoot(downloadsFolderPath, logFile);
24.
25.         fclose(logFile);
26.         wprintf(L"Successfully logged to log.txt.\n");
27.
28.         // Read and print the contents of the log file to the console
29.         wprintf(L"\nLogged Contents:\n");
30.         FILE *readLogFile = _wopen(L"log.txt", L"r");
31.         if (readLogFile)
32.         {
33.             wchar_t buffer[512];
34.             while (fgetws(buffer, 512, readLogFile))
35.             {
36.                 wprintf(L"%ls", buffer);
37.             }
38.             fclose(readLogFile);
39.         }
40.         else
41.         {
42.             wprintf(L"Failed to read log.txt.\n");
43.         }
44.     }
45.     else
46.     {
47.         wprintf(L"Failed to open or write to log.txt.\n");
48.     }
49. }
50. else
51. {
52.     wprintf(L"Failed to get the path of the Downloads folder.\n");
53. }
54. return 0; // Return with a success code (0) to indicate successful execution
55. }
56.
57. // ... Your existing function implementation remains unchanged ...
58.
59. // Function definition to list directories and files in the specified path
60. void listDirectoriesAndFilesInRoot(const wchar_t *path, FILE *logFile)
61. {
62.     // Variables for handling file enumeration in the specified path
63.     WIN32_FIND_DATAW findFileData;
64.     HANDLE hFind = FindFirstFileW((wchar_t *)L"D:\\Downloads\\*", &findFileData);
65.
66.     // Check if file enumeration is successful or not
67.     if (hFind == INVALID_HANDLE_VALUE)
68.     {
69.         // Print an error message if enumeration fails
70.         wprintf(L"Error finding directories and files in: %ls\n", path);
71.         return; // Return from the function
72.     }
73.
74.     // Collect directories and non-directory files separately

```

```

75.     wprintf(L"Directories:\n");
76.     do
77.     {
78.         // Check if the current item is a directory
79.         if (findFileData.dwFileAttributes & FILE_ATTRIBUTE_DIRECTORY)
80.         {
81.             // Exclude '.' and '..' directories from the listing
82.             if (wcscmp(findFileData.cFileName, L"..") != 0 && wcscmp(findFileData.cFileName, L"..")
!= 0)
83.             {
84.                 // Print the directory name to the console
85.                 wprintf(L"Directory: %ls\n", findFileData.cFileName);
86.                 // Log the directory name to the log file
87.                 fwprintf(logFile, L"Directory: %ls\n", findFileData.cFileName);
88.             }
89.         }
90.     } while (FindNextFileW(hFind, &findFileData) != 0);
91.
92.     // Close the handle and reopen it to reset the search
93.     FindClose(hFind);
94.     hFind = FindFirstFileW((wchar_t *)L"D:\\Downloads\\*", &findFileData);
95.
96.     // Collect non-directory files
97.     wprintf(L"Files:\n");
98.     do
99.     {
100.         // Check if the current item is not a directory (i.e., a file)
101.         if (!(findFileData.dwFileAttributes & FILE_ATTRIBUTE_DIRECTORY))
102.         {
103.             // Print the file name to the console
104.             wprintf(L"File: %ls\n", findFileData.cFileName);
105.             // Log the file name to the log file
106.             fwprintf(logFile, L"File: %ls\n", findFileData.cFileName);
107.         }
108.     } while (FindNextFileW(hFind, &findFileData) != 0);
109.
110.     // Close the handle to release resources
111.     FindClose(hFind);
112. }
113.

```

```
14FileSniffer_for_Default_Directories.c - Visual Studio Code
C 14FileSniffer_for_Default_Directories.c X
D:\> Documents > Vs Code Projects > C language Coding > C 14FileSniffer_for_Default_Directories.c > ...
1 #include <stdio.h>
2 #include <windows.h>
3 #include <shlobj.h> // Include the header for SHGetFolderPath
4
5 void listDirectoriesAndFilesInRoot(const wchar_t *path, FILE *logfile);
6
7 int main()
8 {
9     // Variable to store the path of the Downloads folder
10    wchar_t downloadsFolderPath[MAX_PATH];
11
12    // Get the path of the Downloads folder using SHGetFolderPath
13    if (SUCCEEDED(SHGetFolderPath(NULL, CSIDL_PERSONAL, NULL, 0
```

File: Xklusiv Dancers Tun up the Scheme a little @ 2011.mp4
File: XXXTENTACION - Royalty (Official Video) (feat. Ky-Mani Marley, Stefflon Don & Vybz Kartel).mp4
File: XXXTENTACION & Lil Pump - Arms Around You feat. Maluma & Swae Lee [Official Music Video].mp4
File: XXXTENTACION & Lil Pump - Arms Around You feat. Maluma & Swae Lee [Official Music Video]_2.mkv
File: Yeshua.mp4
File: Young Thug - Hoe Tendencias (unreleased).mp4
File: Young Thug - Hoe Tendencias (unreleased)_2.mp4
File: Young Thug - Out The Ghetto [432hz].mp4
File: Young Thug Type Beat X Gunna Type Beat 2023 - Shooting Star - Free Type Beat - Rap_Trap Instrumental.mp4
File: YPB Song 2018.mp4
File: [FREE FOR PROFIT] NATTY X MANDO GTAtv DRILL TYPE BEAT - NATTORIUS APOLLO 2023.mp4
File: _Getintopc.com_Adobe_Lightroom_Classic_2023.mp4
File: _Getintopc.com_Adobe_Photoshop_Lightroom_CC_6.8_Portable.mp4
File: _Getintopc.com_Scan2CAD_2022.mp4
File: How to become a Web3 Blockchain Developer- Ft. Peter Holzer.vtt
File: Masicka & Bounty Killer, Vybz Kartel .mp4
File: (via@Dennis Teo) #shorts.mp4

[Done] exited with code=0 in 20.688 seconds

File Edit View

Directory: 3D
Directory: All DBMS and SQL
Directory: And St
Directory: ar projects
Directory: Arduino
Directory: Azure
Directory: bin
Directory: Calender
Directory: Cisco
Directory: Compressed
Directory: Connecting components in Proteus
Directory: DApp
Directory: Documents
Directory: From pc
Directory: G code
Directory: IDM Cache
Directory: Images
Directory: IR and LCD Simulation Proteus
Directory: MicrosoftWindows.Client.CBS_cw5nlh2txyewy!InputApp
Directory: Motor driver
Directory: Movies
Directory: Music
Directory: nF
Directory: Pictures
Directory: Programs
Directory: Rnhnties

Ln 1, Col 1 110% Windows (CRLF) ANSI

The following code is a template of default folders iteration

```
#include <stdio.h>
#include <windows.h>
#include <shlobj.h> // Include the header for SHGetFolderPath

void processFolder(const wchar_t *folderName);

int main()
{
    // Define an array of folder names
    const wchar_t *defaultFolders[] = {
        L"Downloads",
        L"Documents",
        L"Videos",
        L"Pictures",
        L"Music",
        L"Desktop"
    };

    // Iterate through the array and process each folder
    for (int i = 0; i < sizeof(defaultFolders) / sizeof(defaultFolders[0]); i++)
    {
        processFolder(defaultFolders[i]);
    }

    return 0;
}

// Function to process the contents of a folder
void processFolder(const wchar_t *folderName)
{
    // Variable to store the path of the folder
    wchar_t folderPath[MAX_PATH];

    // Get the path of the specified folder using SHGetFolderPath
    if (SUCCEEDED(SHGetFolderPathW(NULL, CSIDL_PERSONAL, NULL, 0, folderPath)))
    {
        // Append the folder name to the path to get the full folder path
        wcscat(folderPath, L"\\");
        wcscat(folderPath, folderName);

        // Perform operations on the folder here...
        // For example, you can list directories and files in the folder, or do any other processing.

        // For demonstration purposes, let's just print the folder path to the console
        wprintf(L"Folder: %ls\n", folderPath);
    }
    else
    {
        wprintf(L"Failed to get the path of the %ls folder.\n", folderName);
    }
}
```

Visual Studio Code interface showing a C program that iterates through default Windows folders. The program is named `15_iterate_through_default_Windows_folders.c` and is located in the directory `D:\Documents\Vs Code Projects\C language Coding`.

```
1 #include <stdio.h>
2 #include <windows.h>
3 #include <shlobj.h> // Include the header for SHGetFolderPath
4
5 void processFolder(const wchar_t *folderName);
6
7 int main()
8 {
9     // Define an array of folder names
10    const wchar_t *defaultFolders[] = {
11        L"Downloads",
12        L"Documents",
13        L"Videos",
14        L"Pictures",
15        L"Music",
16        L"Desktop"
17    };
18 }
```

The program is executed in the terminal, showing the output of the `gcc` compiler and the execution of the `15_iterate_through_default_Windows_folders` program. The output lists the folders: `D:\Documents\Downloads`, `D:\Documents\Documents`, `D:\Documents\Videos`, `D:\Documents\Pictures`, `D:\Documents\Music`, and `D:\Documents\Desktop`.

The status bar at the bottom indicates the current line and column: `Ln 39, Col 40`. The encoding is `UTF-8` and the line ending is `CRLF`. The window title is `Win32`.

Windows File Explorer interface showing the contents of the `AllSparkData (D:)` drive. The drive contains 15 items, including folders and files.

Name	Date modified	Type
\$AV_ASW	12/06/2023 10:...	File f
+GForce	19/07/2023 07:...	File f
avast! sandbox	19/07/2023 09:...	File f
Desktop	19/07/2023 11:...	File f
Documents	19/07/2023 18:...	File f
Downloads	19/07/2023 17:...	File f
Dukto	26/05/2023 21:...	File f
Msc	11/07/2023 12:...	File f
Music	19/07/2023 08:...	File f
Pictures	18/07/2023 09:...	File f
University Days	01/07/2023 11:...	File f
Videos	19/07/2023 17:...	File f
IMG-20230223-WA0003.jpg	31/05/2023 20:...	JPG
Network - Shortcut	25/06/2023 23:...	Shor
NouvaTech	16/04/2023 19:...	File

The status bar at the bottom indicates the current view: `15 items`.

The above code had the following limitations:

The code you provided seems to be a C program that lists directories and files in specific folders on your computer and logs the results to a file. The folders it lists are the default ones like Downloads, Documents, Videos, Pictures, Music, and Desktop. However, there are a few issues in the code that need to be addressed.

Hardcoded Path: The code currently lists the directories and files in the "D:\Documents" folder. To make it more flexible and work on any system, you can replace the hardcoded path with a dynamic path using the SHGetFolderPath function.

Folder Enumeration: The listDirectoriesAndFilesInRoot function currently uses the "D:\Documents" folder directly for enumeration. Instead, you should use the path parameter that's passed to the function.

Incorrect Folder Enumeration: The listDirectoriesAndFilesInRoot function doesn't properly use the provided path for enumeration. It uses a hardcoded path instead. To fix this, you need to replace (wchar_t *)L"D:\\Documents*" with (wchar_t *)path in both places.

Incorrect Log Folder Path: In the processDefaultFolders function, when calling listDirectoriesAndFilesInRoot, it passes the folder names as paths directly. Instead, it should use the dynamic path obtained from SHGetFolderPath for each default folder.

With these modifications, the code should now correctly list the directories and files in the default folders and log the results to the specified log file. Additionally, it uses the SHGetFolderPath function to dynamically obtain the path of the "Documents" folder, making the code more adaptable to different systems

The listDirectoriesAndFilesInRoot function is not correctly using the provided path for enumeration, and it's always using the "D:\Documents" folder for listing, which leads to the incorrect log.

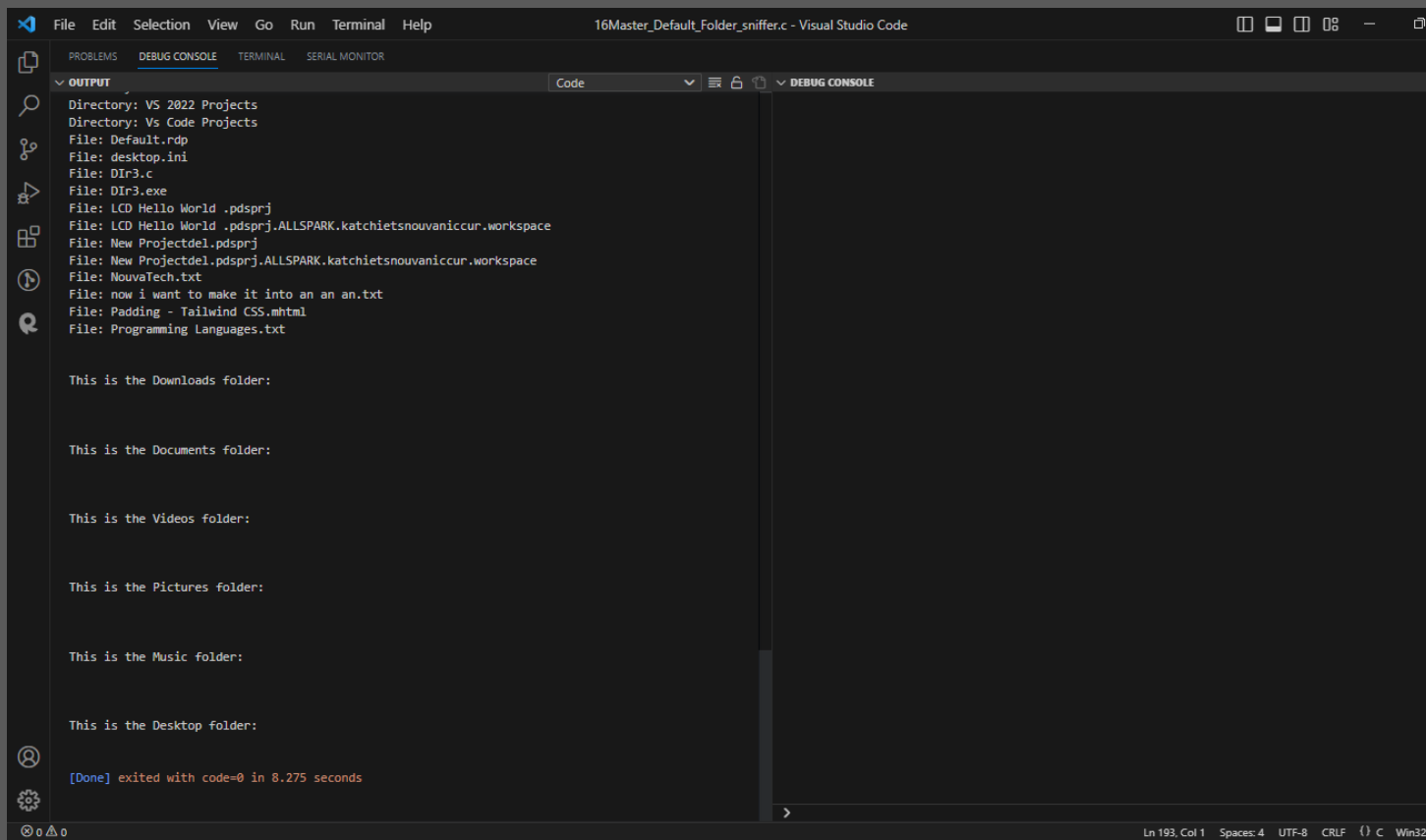
To fix this, we need to replace the hardcoded path (wchar_t *)L"D:\\Documents*" with the path parameter in the listDirectoriesAndFilesInRoot function.

still an issue with the code when it comes to listing the contents of the default folders like Downloads, Documents, Videos, etc. The code is not correctly listing the contents of those folders, and it shows "Error finding directories and files" for each of them.

The problem lies in the processDefaultFolders function, where we are not correctly passing the paths of the default folders to the listDirectoriesAndFilesInRoot function. We need to provide the correct paths to the listDirectoriesAndFilesInRoot function.

To fix this, we need to use the SHGetFolderPath function to obtain the paths of the default folders dynamically

With this modification, the processDefaultFolders function should correctly list the contents of each default folder and log them to the file.



```
File Edit Selection View Go Run Terminal Help
16Master_Default_Folder_sniffer.c - Visual Studio Code

PROBLEMS DEBUG CONSOLE TERMINAL SERIAL MONITOR
▼ OUTPUT Code
Directory: VS 2022 Projects
Directory: Vs Code Projects
File: Default.rdp
File: desktop.ini
File: Dir3.c
File: Dir3.exe
File: LCD Hello World.pdsprj
File: LCD Hello World.pdsprj.ALLSPARK.katchietsnouvaniccur.workspace
File: New Projectdel.pdsprj
File: New Projectdel.pdsprj.ALLSPARK.katchietsnouvaniccur.workspace
File: NouvaTech.txt
File: now i want to make it into an an.txt
File: Padding - Tailwind CSS.mhtml
File: Programming Languages.txt

This is the Downloads folder:

This is the Documents folder:

This is the Videos folder:

This is the Pictures folder:

This is the Music folder:

This is the Desktop folder:

[Done] exited with code=0 in 8.275 seconds
Ln 193, Col 1 Spaces: 4 UTF-8 CRLF {} C Win32
```

t there's still an issue with the code when listing the contents of the default folders. The issue lies in the way we are using the SHGetFolderPath function to obtain the paths of the default folders.

Instead of using SHGetFolderPath for each default folder, we can directly access the default folders using the corresponding CSIDLs (constants). We don't need to use SHGetFolderPath for these default folders because they have predefined CSIDL values.

Let's modify the `processDefaultFolders` function to directly access the default folders using their CSIDL values:

To include the Downloads folder in the output, you can add the CSIDL value for the Downloads folder (`CSIDL_PERSONAL`) to the `defaultFolderCSIDLs` array in the `processDefaultFolders` function.

The following code printed for all the default folders except for the downloads folder:

It displayed for :

1. Desktop folder
2. Documents folder
3. Videos folder
4. Pictures folder
5. Music folder
6. Common Desktop folder

It didn't display correctly for:

1. Downloads folder

```
1. #include <stdio.h>
2. #include <windows.h>
3. #include <shlobj.h> // Include the header for SHGetFolderPath
4.
5. // Function prototypes
6. void listDirectoriesAndFilesInRoot(const wchar_t *path, FILE *logFile);
7. void processDefaultFolders(FILE *logFile);
8.
9. int main()
10. {
11.     // Variable to store the path of the current code file
12.     wchar_t currentCodeFilePath[FILENAME_MAX];
13.     // Get the path of the current executable (the code file)
14.     GetModuleFileNameW(NULL, currentCodeFilePath, FILENAME_MAX);
15.     // Extract the directory path by removing the file name from the path
16.     const wchar_t *lastBackslash = wcsrchr(currentCodeFilePath, L'\\');
17.     currentCodeFilePath[lastBackslash - currentCodeFilePath + 1] = L'\\0';
18.
19.     // Get the path of the 'Documents' folder using SHGetFolderPath
20.     wchar_t documentsFolderPath[FILENAME_MAX];
21.     if (SHGetFolderPathW(NULL, CSIDL_PERSONAL, NULL, 0, documentsFolderPath) != S_OK)
22.     {
23.         // Print an error message if getting the folder path fails
24.         wprintf(L"Failed to get 'Documents' folder path.\n");
25.         return 1; // Return with an error code (1) to indicate failure
26.     }
27.
28.     // Print the current directory and the code file's location
29.     wprintf(L"Listing directories and files in %ls\n(Current Code File Location: %ls)\n",
documentsFolderPath, currentCodeFilePath);
30.
31.     // Variable to store the log file name
32.     wchar_t logFileName[FILENAME_MAX];
33.     // Create the log file name using the current directory path
34.     _snwprintf(logFileName, FILENAME_MAX, L"%lsbin\\1.txt", currentCodeFilePath);
35.
36.     // Check if the 'bin' folder exists, if not, create it
37.     wchar_t binFolderPath[FILENAME_MAX];
38.     _snwprintf(binFolderPath, FILENAME_MAX, L"%lsbin", currentCodeFilePath);
39.     if (!CreateDirectoryW(binFolderPath, NULL))
40.     {
41.         DWORD error = GetLastError();
42.         if (error != ERROR_ALREADY_EXISTS)
43.         {
44.             // Print an error message if the folder creation fails
45.             wprintf(L"Failed to create 'bin' folder: %ls\n", binFolderPath);
46.             return 1; // Return with an error code (1) to indicate failure
47.         }

```

```

48.     }
49.
50.     // Open the log file in "append" mode (add data to the existing file)
51.     FILE *logFile = _w fopen(logFileName, L"a");
52.     if (logFile)
53.     {
54.         // Call the function to list directories and files in the specified path (Documents folder)
55.         listDirectoriesAndFilesInRoot(documentsFolderPath, logFile);
56.
57.         // Call the new function to iterate through the default folders and append data to the log
file
58.         processDefaultFolders(logFile);
59.
60.         // Close the log file after writing the data
61.         fclose(logFile);
62.         // Print a success message with the log file name
63.         wprintf(L"Successfully logged to %ls.\n", logFileName);
64.
65.         // Read and print the contents of the log file to the console
66.         wprintf(L"\nLogged Contents:\n");
67.         FILE *readLogFile = _w fopen(logFileName, L"r");
68.         if (readLogFile)
69.         {
70.             wchar_t buffer[512];
71.             while (fgetws(buffer, 512, readLogFile))
72.             {
73.                 wprintf(L"%ls", buffer);
74.             }
75.             fclose(readLogFile);
76.         }
77.         else
78.         {
79.             // Print an error message if reading the log file fails
80.             wprintf(L"Failed to read %ls.\n", logFileName);
81.         }
82.     }
83.     else
84.     {
85.         // Print an error message if opening or writing to the log file fails
86.         wprintf(L"Failed to open or write to %ls.\n", logFileName);
87.     }
88.
89.     return 0; // Return with a success code (0) to indicate successful execution
90. }
91.
92. // Function definition to list directories and files in the specified path
93. void listDirectoriesAndFilesInRoot(const wchar_t *path, FILE *logFile)
94. {
95.     // Variables for handling file enumeration in the specified path
96.     wchar_t searchPath[MAX_PATH];
97.     WIN32_FIND_DATAW findFileData;
98.
99.     // Create the search pattern for the specified path
100.    _snwprintf(searchPath, MAX_PATH, L"%s\\*", path);
101.
102.    // Find the first file in the specified path
103.    HANDLE hFind = FindFirstFileW(searchPath, &findFileData);

```

```

104.
105.     // Check if file enumeration is successful or not
106.     if (hFind == INVALID_HANDLE_VALUE)
107.     {
108.         // Print an error message if enumeration fails
109.         wprintf(L"Error finding directories and files in: %ls\n", path);
110.         return; // Return from the function
111.     }
112.
113.     // Collect directories and non-directory files separately
114.     wprintf(L"Directories:\n");
115.     do
116.     {
117.         // Check if the current item is a directory
118.         if (findFileData.dwFileAttributes & FILE_ATTRIBUTE_DIRECTORY)
119.         {
120.             // Exclude '.' and '..' directories from the listing
121.             if (wcscmp(findFileData.cFileName, L".") != 0 && wcscmp(findFileData.cFileName,
122. L"..") != 0)
123.             {
124.                 // Print the directory name to the console
125.                 wprintf(L"Directory: %ls\n", findFileData.cFileName);
126.                 // Log the directory name to the log file
127.                 fwprintf(logFile, L"Directory: %ls\n", findFileData.cFileName);
128.             }
129.         } while (FindNextFileW(hFind, &findFileData) != 0);
130.
131.         // Close the handle to release resources
132.         FindClose(hFind);
133.
134.         // Reopen the handle to find non-directory files
135.         hFind = FindFirstFileW(searchPath, &findFileData);
136.
137.         // Collect non-directory files
138.         wprintf(L"Files:\n");
139.         do
140.         {
141.             // Check if the current item is not a directory (i.e., a file)
142.             if (!(findFileData.dwFileAttributes & FILE_ATTRIBUTE_DIRECTORY))
143.             {
144.                 // Print the file name to the console
145.                 wprintf(L"File: %ls\n", findFileData.cFileName);
146.                 // Log the file name to the log file
147.                 fwprintf(logFile, L"File: %ls\n", findFileData.cFileName);
148.             }
149.         } while (FindNextFileW(hFind, &findFileData) != 0);
150.
151.         // Close the handle to release resources
152.         FindClose(hFind);
153.     }
154.
155. void processDefaultFolders(FILE *logFile)
156. {
157.     // Define an array of folder CSIDLs (constants) along with their descriptions
158.     const int defaultFolderCSIDLs[] = {

```

```

159.         CSIDL_DESKTOP, CSIDL_PERSONAL, CSIDL_MYVIDEO, CSIDL_MYPICTURES, CSIDL_MYMUSIC,
        CSIDL_COMMON_DESKTOPDIRECTORY, 0x001A
160.     };
161.
162.     // Define the corresponding folder descriptions
163.     const wchar_t *defaultFolderDescriptions[] = {
164.         L"This is the Desktop folder:",
165.         L"This is the Documents folder:",
166.         L"This is the Videos folder:",
167.         L"This is the Pictures folder:",
168.         L"This is the Music folder:",
169.         L"This is the Common Desktop folder:",
170.         L"This is the Downloads folder:"
171.     };
172.
173.     // Iterate through the array and call listDirectoriesAndFilesInRoot for each folder
174.     for (int i = 0; i < sizeof(defaultFolderCSIDLs) / sizeof(defaultFolderCSIDLs[0]); i++)
175.     {
176.         // Get the path of the default folder using CSIDL
177.         wchar_t folderPath[MAX_PATH];
178.         if (SHGetFolderPathW(NULL, defaultFolderCSIDLs[i], NULL, 0, folderPath) == S_OK)
179.         {
180.             // Add the folder description to the log file
181.             fprintf(logFile, L"\n\n%s\n\n", defaultFolderDescriptions[i]);
182.
183.             // Call listDirectoriesAndFilesInRoot for each default folder
184.             listDirectoriesAndFilesInRoot(folderPath, logFile);
185.         }
186.         else
187.         {
188.             // Print an error message if getting the folder path fails
189.             wprintf(L"Failed to get default folder path for CSIDL %d.\n",
        defaultFolderCSIDLs[i]);
190.         }
191.     }
192. }

```

```
16Master_Default_Folder_sniffer.c - Visual Studio Code
16Master_Default_Folder_sniffer.c
D:\> Documents > Vs Code Projects > C language Coding > 16Master_Default_Folder_sniffer.c > processDefaultFolders(FILE
157 {
158     // Define an array of folder CSIDLs (constants) along with their
    descriptions
159     const int defaultFolderCSIDLs[] = {
160         CSIDL_DESKTOP, CSIDL_PERSONAL, CSIDL_MYVIDEO, CSIDL_MYPICTURES,
        CSIDL_MYMUSIC, CSIDL_COMMON_DESKTOPDIRECTORY, 0x001A
161     };
162
163     // Define the corresponding folder descriptions
164     const wchar_t *defaultFolderDescriptions[] = {
165         L"This is the Desktop folder:",
166         L"This is the Documents folder:",
167         L"This is the Videos folder:",
168         L"This is the Pictures folder:",
169         L"This is the Music folder:",
170         L"This is the Common-Desktop folder:",
171         L"This is the Downloads folder:"
172     };
173
174     // Iterate through the array and call listDirectoriesAndFilesInRoot for
    each folder
175     for (int i = 0; i < sizeof(defaultFolderCSIDLs) / sizeof
        (defaultFolderCSIDLs[0]); i++)
176     {
177
178     }
179 }
180
181 // End of file
182
183 // End of file
184
185 // End of file
186
187 // End of file
188
189 // End of file
190
191 // End of file
192
193 // End of file
194
195 // End of file
196
197 // End of file
198
199 // End of file
200
201 // End of file
202
203 // End of file
204
205 // End of file
206
207 // End of file
208
209 // End of file
210
211 // End of file
212
213 // End of file
214
215 // End of file
216
217 // End of file
218
219 // End of file
220
221 // End of file
222
223 // End of file
224
225 // End of file
226
227 // End of file
228
229 // End of file
230
231 // End of file
232
233 // End of file
234
235 // End of file
236
237 // End of file
238
239 // End of file
240
241 // End of file
242
243 // End of file
244
245 // End of file
246
247 // End of file
248
249 // End of file
250
251 // End of file
252
253 // End of file
254
255 // End of file
256
257 // End of file
258
259 // End of file
260
261 // End of file
262
263 // End of file
264
265 // End of file
266
267 // End of file
268
269 // End of file
270
271 // End of file
272
273 // End of file
274
275 // End of file
276
277 // End of file
278
279 // End of file
280
281 // End of file
282
283 // End of file
284
285 // End of file
286
287 // End of file
288
289 // End of file
290
291 // End of file
292
293 // End of file
294
295 // End of file
296
297 // End of file
298
299 // End of file
300
301 // End of file
302
303 // End of file
304
305 // End of file
306
307 // End of file
308
309 // End of file
310
311 // End of file
312
313 // End of file
314
315 // End of file
316
317 // End of file
318
319 // End of file
320
321 // End of file
322
323 // End of file
324
325 // End of file
326
327 // End of file
328
329 // End of file
330
331 // End of file
332
333 // End of file
334
335 // End of file
336
337 // End of file
338
339 // End of file
340
341 // End of file
342
343 // End of file
344
345 // End of file
346
347 // End of file
348
349 // End of file
350
351 // End of file
352
353 // End of file
354
355 // End of file
356
357 // End of file
358
359 // End of file
360
361 // End of file
362
363 // End of file
364
365 // End of file
366
367 // End of file
368
369 // End of file
370
371 // End of file
372
373 // End of file
374
375 // End of file
376
377 // End of file
378
379 // End of file
380
381 // End of file
382
383 // End of file
384
385 // End of file
386
387 // End of file
388
389 // End of file
390
391 // End of file
392
393 // End of file
394
395 // End of file
396
397 // End of file
398
399 // End of file
400
401 // End of file
402
403 // End of file
404
405 // End of file
406
407 // End of file
408
409 // End of file
410
411 // End of file
412
413 // End of file
414
415 // End of file
416
417 // End of file
418
419 // End of file
420
421 // End of file
422
423 // End of file
424
425 // End of file
426
427 // End of file
428
429 // End of file
430
431 // End of file
432
433 // End of file
434
435 // End of file
436
437 // End of file
438
439 // End of file
440
441 // End of file
442
443 // End of file
444
445 // End of file
446
447 // End of file
448
449 // End of file
450
451 // End of file
452
453 // End of file
454
455 // End of file
456
457 // End of file
458
459 // End of file
460
461 // End of file
462
463 // End of file
464
465 // End of file
466
467 // End of file
468
469 // End of file
470
471 // End of file
472
473 // End of file
474
475 // End of file
476
477 // End of file
478
479 // End of file
480
481 // End of file
482
483 // End of file
484
485 // End of file
486
487 // End of file
488
489 // End of file
490
491 // End of file
492
493 // End of file
494
495 // End of file
496
497 // End of file
498
499 // End of file
500
501 // End of file
502
503 // End of file
504
505 // End of file
506
507 // End of file
508
509 // End of file
510
511 // End of file
512
513 // End of file
514
515 // End of file
516
517 // End of file
518
519 // End of file
520
521 // End of file
522
523 // End of file
524
525 // End of file
526
527 // End of file
528
529 // End of file
530
531 // End of file
532
533 // End of file
534
535 // End of file
536
537 // End of file
538
539 // End of file
540
541 // End of file
542
543 // End of file
544
545 // End of file
546
547 // End of file
548
549 // End of file
550
551 // End of file
552
553 // End of file
554
555 // End of file
556
557 // End of file
558
559 // End of file
560
561 // End of file
562
563 // End of file
564
565 // End of file
566
567 // End of file
568
569 // End of file
570
571 // End of file
572
573 // End of file
574
575 // End of file
576
577 // End of file
578
579 // End of file
580
581 // End of file
582
583 // End of file
584
585 // End of file
586
587 // End of file
588
589 // End of file
590
591 // End of file
592
593 // End of file
594
595 // End of file
596
597 // End of file
598
599 // End of file
600
601 // End of file
602
603 // End of file
604
605 // End of file
606
607 // End of file
608
609 // End of file
610
611 // End of file
612
613 // End of file
614
615 // End of file
616
617 // End of file
618
619 // End of file
620
621 // End of file
622
623 // End of file
624
625 // End of file
626
627 // End of file
628
629 // End of file
630
631 // End of file
632
633 // End of file
634
635 // End of file
636
637 // End of file
638
639 // End of file
640
641 // End of file
642
643 // End of file
644
645 // End of file
646
647 // End of file
648
649 // End of file
650
651 // End of file
652
653 // End of file
654
655 // End of file
656
657 // End of file
658
659 // End of file
660
661 // End of file
662
663 // End of file
664
665 // End of file
666
667 // End of file
668
669 // End of file
670
671 // End of file
672
673 // End of file
674
675 // End of file
676
677 // End of file
678
679 // End of file
680
681 // End of file
682
683 // End of file
684
685 // End of file
686
687 // End of file
688
689 // End of file
690
691 // End of file
692
693 // End of file
694
695 // End of file
696
697 // End of file
698
699 // End of file
700
701 // End of file
702
703 // End of file
704
705 // End of file
706
707 // End of file
708
709 // End of file
710
711 // End of file
712
713 // End of file
714
715 // End of file
716
717 // End of file
718
719 // End of file
720
721 // End of file
722
723 // End of file
724
725 // End of file
726
727 // End of file
728
729 // End of file
730
731 // End of file
732
733 // End of file
734
735 // End of file
736
737 // End of file
738
739 // End of file
740
741 // End of file
742
743 // End of file
744
745 // End of file
746
747 // End of file
748
749 // End of file
750
751 // End of file
752
753 // End of file
754
755 // End of file
756
757 // End of file
758
759 // End of file
760
761 // End of file
762
763 // End of file
764
765 // End of file
766
767 // End of file
768
769 // End of file
770
771 // End of file
772
773 // End of file
774
775 // End of file
776
777 // End of file
778
779 // End of file
780
781 // End of file
782
783 // End of file
784
785 // End of file
786
787 // End of file
788
789 // End of file
790
791 // End of file
792
793 // End of file
794
795 // End of file
796
797 // End of file
798
799 // End of file
800
801 // End of file
802
803 // End of file
804
805 // End of file
806
807 // End of file
808
809 // End of file
810
811 // End of file
812
813 // End of file
814
815 // End of file
816
817 // End of file
818
819 // End of file
820
821 // End of file
822
823 // End of file
824
825 // End of file
826
827 // End of file
828
829 // End of file
830
831 // End of file
832
833 // End of file
834
835 // End of file
836
837 // End of file
838
839 // End of file
840
841 // End of file
842
843 // End of file
844
845 // End of file
846
847 // End of file
848
849 // End of file
850
851 // End of file
852
853 // End of file
854
855 // End of file
856
857 // End of file
858
859 // End of file
860
861 // End of file
862
863 // End of file
864
865 // End of file
866
867 // End of file
868
869 // End of file
870
871 // End of file
872
873 // End of file
874
875 // End of file
876
877 // End of file
878
879 // End of file
880
881 // End of file
882
883 // End of file
884
885 // End of file
886
887 // End of file
888
889 // End of file
890
891 // End of file
892
893 // End of file
894
895 // End of file
896
897 // End of file
898
899 // End of file
900
901 // End of file
902
903 // End of file
904
905 // End of file
906
907 // End of file
908
909 // End of file
910
911 // End of file
912
913 // End of file
914
915 // End of file
916
917 // End of file
918
919 // End of file
920
921 // End of file
922
923 // End of file
924
925 // End of file
926
927 // End of file
928
929 // End of file
930
931 // End of file
932
933 // End of file
934
935 // End of file
936
937 // End of file
938
939 // End of file
940
941 // End of file
942
943 // End of file
944
945 // End of file
946
947 // End of file
948
949 // End of file
950
951 // End of file
952
953 // End of file
954
955 // End of file
956
957 // End of file
958
959 // End of file
960
961 // End of file
962
963 // End of file
964
965 // End of file
966
967 // End of file
968
969 // End of file
970
971 // End of file
972
973 // End of file
974
975 // End of file
976
977 // End of file
978
979 // End of file
980
981 // End of file
982
983 // End of file
984
985 // End of file
986
987 // End of file
988
989 // End of file
990
991 // End of file
992
993 // End of file
994
995 // End of file
996
997 // End of file
998
999 // End of file
1000
1001 // End of file
1002
1003 // End of file
1004
1005 // End of file
1006
1007 // End of file
1008
1009 // End of file
1010
1011 // End of file
1012
1013 // End of file
1014
1015 // End of file
1016
1017 // End of file
1018
1019 // End of file
1020
1021 // End of file
1022
1023 // End of file
1024
1025 // End of file
1026
1027 // End of file
1028
1029 // End of file
1030
1031 // End of file
1032
1033 // End of file
1034
1035 // End of file
1036
1037 // End of file
1038
1039 // End of file
1040
1041 // End of file
1042
1043 // End of file
1044
1045 // End of file
1046
1047 // End of file
1048
1049 // End of file
1050
1051 // End of file
1052
1053 // End of file
1054
1055 // End of file
1056
1057 // End of file
1058
1059 // End of file
1060
1061 // End of file
1062
1063 // End of file
1064
1065 // End of file
1066
1067 // End of file
1068
1069 // End of file
1070
1071 // End of file
1072
1073 // End of file
1074
1075 // End of file
1076
1077 // End of file
1078
1079 // End of file
1080
1081 // End of file
1082
1083 // End of file
1084
1085 // End of file
1086
1087 // End of file
1088
1089 // End of file
1090
1091 // End of file
1092
1093 // End of file
1094
1095 // End of file
1096
1097 // End of file
1098
1099 // End of file
1100
1101 // End of file
1102
1103 // End of file
1104
1105 // End of file
1106
1107 // End of file
1108
1109 // End of file
1110
1111 // End of file
1112
1113 // End of file
1114
1115 // End of file
1116
1117 // End of file
1118
1119 // End of file
1120
1121 // End of file
1122
1123 // End of file
1124
1125 // End of file
1126
1127 // End of file
1128
1129 // End of file
1130
1131 // End of file
1132
1133 // End of file
1134
1135 // End of file
1136
1137 // End of file
1138
1139 // End of file
1140
1141 // End of file
1142
1143 // End of file
1144
1145 // End of file
1146
1147 // End of file
1148
1149 // End of file
1150
1151 // End of file
1152
1153 // End of file
1154
1155 // End of file
1156
1157 // End of file
1158
1159 // End of file
1160
1161 // End of file
1162
1163 // End of file
1164
1165 // End of file
1166
1167 // End of file
1168
1169 // End of file
1170
1171 // End of file
1172
1173 // End of file
1174
1175 // End of file
1176
1177 // End of file
1178
1179 // End of file
1180
1181 // End of file
1182
1183 // End of file
1184
1185 // End of file
1186
1187 // End of file
1188
1189 // End of file
1190
1191 // End of file
1192
1193 // End of file
1194
1195 // End of file
1196
1197 // End of file
1198
1199 // End of file
1200
1201 // End of file
1202
1203 // End of file
1204
1205 // End of file
1206
1207 // End of file
1208
1209 // End of file
1210
1211 // End of file
1212
1213 // End of file
1214
1215 // End of file
1216
1217 // End of file
1218
1219 // End of file
1220
1221 // End of file
1222
1223 // End of file
1224
1225 // End of file
1226
1227 // End of file
1228
1229 // End of file
1230
1231 // End of file
1232
1233 // End of file
1234
1235 // End of file
1236
1237 // End of file
1238
1239 // End of file
1240
1241 // End of file
1242
1243 // End of file
1244
1245 // End of file
1246
1247 // End of file
1248
1249 // End of file
1250
1251 // End of file
1252
1253 // End of file
1254
1255 // End of file
1256
1257 // End of file
1258
1259 // End of file
1260
1261 // End of file
1262
1263 // End of file
1264
1265 // End of file
1266
1267 // End of file
1268
1269 // End of file
1270
1271 // End of file
1272
1273 // End of file
1274
1275 // End of file
1276
1277 // End of file
1278
1279 // End of file
1280
1281 // End of file
1282
1283 // End of file
1284
1285 // End of file
1286
1287 // End of file
1288
1289 // End of file
1290
1291 // End of file
1292
1293 // End of file
1294
1295 // End of file
1296
1297 // End of file
1298
1299 // End of file
1300
1301 // End of file
1302
1303 // End of file
1304
1305 // End of file
1306
1307 // End of file
1308
1309 // End of file
1310
1311 // End of file
1312
1313 // End of file
1314
1315 // End of file
1316
1317 // End of file
1318
1319 // End of file
1320
1321 // End of file
1322
1323 // End of file
1324
1325 // End of file
1326
1327 // End of file
1328
1329 // End of file
1330
1331 // End of file
1332
1333 // End of file
1334
1335 // End of file
1336
1337 // End of file
1338
1339 // End of file
1340
1341 // End of file
1342
1343 // End of file
1344
1345 // End of file
1346
1347 // End of file
1348
1349 // End of file
1350
1351 // End of file
1352
1353 // End of file
1354
1355 // End of file
1356
1357 // End of file
1358
1359 // End of file
1360
1361 // End of file
1362
1363 // End of file
1364
1365 // End of file
1366
1367 // End of file
1368
1369 // End of file
1370
1371 // End of file
1372
1373 // End of file
1374
1375 // End of file
1376
1377 // End of file
1378
1379 // End of file
1380
1381 // End of file
1382
1383 // End of file
1384
1385 // End of file
1386
1387 // End of file
1388
1389 // End of file
1390
1391 // End of file
1392
1393 // End of file
1394
1395 // End of file
1396
1397 // End of file
1398
1399 // End of file
1400
1401 // End of file
1402
1403 // End of file
1404
1405 // End of file
1406
1407 // End of file
1408
1409 // End of file
1410
1411 // End of file
1412
1413 // End of file
1414
1415 // End of file
1416
1417 // End of file
1418
1419 // End of file
1420
1421 // End of file
1422
1423 // End of file
1424
1425 // End of file
1426
1427 // End of file
1428
1429 // End of file
1430
1431 // End of file
1432
1433 // End of file
1434
1435 // End of file
1436
1437 // End of file
1438
1439 // End of file
1440
1441 // End of file
1442
1443 // End of file
1444
1445 // End of file
1446
1447 // End of file
1448
1449 // End of file
1450
1451 // End of file
1452
1453 // End of file
1454
1455 // End of file
1456
1457 // End of file
1458
1459 // End of file
1460
1461 // End of file
1462
1463 // End of file
1464
1465 // End of file
1466
1467 // End of file
1468
1469 // End of file
1470
1471 // End of file
1472
1473 // End of file
1474
1475 // End of file
1476
1477 // End of file
1478
1479 // End of file
1480
1481 // End of file
1482
1483 // End of file
1484
1485 // End of file
1486
1487 // End of file
1488
1489 // End of file
1490
1491 // End of file
1492
1493 // End of file
1494
1495 // End of file
1496
1497 // End of file
1498
1499 // End of file
1500
1501 // End of file
1502
1503 // End of file
1504
1505 // End of file
1506
1507 // End of file
1508
1509 // End of file
1510
1511 // End of file
1512
1513 // End of file
1514
1515 // End of file
1516
1517 // End of file
1518
1519 // End of file
1520
1521 // End of file
1522
1523 // End of file
1524
1525 // End of file
1526
1527 // End of file
1528
1529 // End of file
1530
1531 // End of file
1532
1533 // End of file
1534
1535 // End of file
1536
1537 // End of file
1538
1539 // End of file
1540
1541 // End of file
1542
1543 // End of file
1544
1545 // End of file
1546
1547 // End of file
1548
1549 // End of file
1550
1551 // End of file
1552
1553 // End of file
1554
1555 // End of file
1556
1557 // End of file
1558
1559 // End of file
1560
1561 // End of file
1562
1563 // End of file
1564
1565 // End of file
1566
1567 // End of file
1568
1569 // End of file
1570
1571 // End of file
1572
1573 // End of file
1574
1575 // End of file
1576
1577 // End of file
1578
1579 // End of file
1580
1581 // End of file
1582
1583 // End of file
1584
1585 // End of file
1586
1587 // End of file
1588
1589 // End of file
1590
1591 // End of file
1592
1593 // End of file
1594
1595 // End of file
1596
1597 // End of file
1598
1599 // End of file
1600
1601 // End of file
1602
1603 // End of file
1604
1605 // End of file
1606
1607 // End of file
1608
1609 // End of file
1610
1611 // End of file
1612
1613 // End of file
1614
1615 // End of file
1616
1617 // End of file
1618
1619 // End of file
1620
1621 // End of file
1622
1623 // End of file
1624
1625 // End of file
1626
1627 // End of file
1628
1629 // End of file
1630
1631 // End of file
1632
1633 // End of file
1634
1635 // End of file
1636
1637 // End of file
1638
1639 // End of file
1640
1641 // End of file
1642
1643 // End of file
1644
1645 // End of file
1646
1647 // End of file
1648
1649 // End of file
1650
1651 // End of file
1652
1653 // End of file
1654
1655 // End of file
1656
1657 // End of file
1658
1659 // End of file
1660
1661 // End of file
1662
1663 // End of file
1664
1665 // End of file
1666
1667 // End of file
1668
1669 // End of file
1670
1671 // End of file
1672
1673 // End of file
1674
1675 // End of file
1676
1677 // End of file
1678
1679 // End of file
1680
1681 // End of file
1682
1683 // End of file
1684
1685 // End of file
1686
1687 // End of file
1688
1689 // End of file
1690
1691 // End of file
1692
1693 // End of file
1694
1695 // End of file
1696
1697 // End of file
1698
1699 // End of file
1700
1701 // End of file
1702
1703 // End of file
1704
1705 // End of file
1706
1707 // End of file
1708
1709 // End of file
1710
1711 // End of file
1712
1713 // End of file
1714
1715 // End of file
1716
1717 // End of file
1718
1719 // End of file
1720
1721 // End of file
1722
1723 // End of file
1724
1725 // End of file
1726
1727 // End of file
1728
1729 // End of file
1730
1731 // End of file
1732
1733 // End of file
1734
1735 // End of file
1736
1737 // End of file
1738
1739 // End of file
1740
1741 // End of file
1742
1743 // End of file
1744
1745 // End of file
1746
1747 // End of file
1748
1749 // End of file
1750
1751 // End of file
1752
1753 // End of file
1754
1755 // End of file
1756
1757 // End of file
1758
1759 // End of file
1760
1761 // End of file
1762
1763 // End of file
1764
1765 // End of file
1766
1767 // End of file
1768
1769 // End of file
1770
1771 // End of file
1772
1773 // End of file
1774
1775 // End of file
1776
1777 // End of file
1778
1779 // End of file
1780
1781 // End of file
1782
1783 // End of file
1784
1785 // End of file
1786
1787 // End of file
1788
1789 // End of file
1790
1791 // End of file
1792
1793 // End of file
1794
1795 // End of file
1796
1797 // End of file
1798
1799 // End of file
1800
1801 // End of file
1802
1803 // End of file
1804
1805 // End of file
1806
1807 // End of file
1808
1809 // End of file
1810
1811 // End of file
1812
1813 // End of file
1814
1815 // End of file
1816
1817 // End of file
1818
1819 // End of file
1820
1821 // End of file
1822
1823 // End of file
1824
1825 // End of file
1826
1827 // End of file
1828
1829 // End of file
1830
1831 // End of file
1832
1833 // End of file
1834
1835 // End of file
1836
1837 // End of file
1838
1839 // End of file
1840
1841 // End of file
1842
1843 // End of file
1844
1845 // End of file
1846
1847 // End of file
1848
1849 // End of file
1850
1851 // End of file
1852
1853 // End of file
1854
1855 // End of file
1856
1857 // End of file
1858
1859 // End of file
1860
1861 // End of file
1862
1863 // End of file
1864
1865 // End of file
1866
1867 // End of file
1868
1869 // End of file
1870
1871 // End of file
1872
1873 // End of file
1874
1875 // End of file
1876
1877 // End of file
1878
1879 // End of file
1880
1881 // End of file
1882
1883 // End of file
1884
1885 // End of file
1886
1887 // End of file
1888
1889 // End of file
1890
1891 // End of file
1892
1893 // End of file
1894
1895 // End of file
1896
1897 // End of file
1898
1899 // End of file
1900
1901 // End of file
1902
1903 // End of file
1904
1905 // End of file
1906
1907 // End of file
1908
1909 // End of file
1910
1911 // End of file
1912
1913 // End of file
1914
1915 // End of file
1916
1917 // End of file
1918
1919 // End of file
1920
1921 // End of file
1922
1923 // End of file
1924
1925 // End of file
1926
1927 // End of file
1928
1929 // End of file
1930
1931 // End of file
1932
1933 // End of file
1934
1935 // End of file
1936
1937 // End of file
1938
1939 // End of file
1940
1941 // End of file
1942
1943 // End of file
1944
1945 // End of file
1946
1947 // End of file
1948
1949 // End of file
1950
1951 // End of file
1952
1953 // End of file
1954
1955 // End of file
1956
1957 // End of file
1958
1959 // End of file
1960
1961 // End of file
1962
1963 // End of file
1964
1965 // End of file
1966
1967 // End of file
1968
1969 // End of file
1970
1971 // End of file
1972
1973 // End of file
1974
1975 // End of file
1976
1977 // End of file
1978
1979 // End of file
1980
1981 // End of file
1982
1983 // End of file
1984
1985 // End of file
1986
1987 // End of file
1988
1989 // End of file
1990
1991 // End of file
1992
1993 // End of file
1994
1995 // End of file
1996
1997 // End of file
1998
1999 // End of file
2000
2001 // End of file
2002
2003 // End of file
2004
2005 // End of file
2006
2007 // End of file
2008
2009 //
```

To be able to get and to the contents the download folder, I had to hard Code the path D:\Downloads

```
1. #include <stdio.h>
2. #include <windows.h>
3. #include <string.h>
4.
5. void listDirectoriesAndFilesInRoot(const wchar_t *path, FILE *logFile);
6.
7. int main()
8. {
9.     wchar_t currentCodeFilePath[FILENAME_MAX];
10.    GetModuleFileNameW(NULL, currentCodeFilePath, FILENAME_MAX);
11.    const wchar_t *lastBackslash = wcsrchr(currentCodeFilePath, L'\\');
12.    currentCodeFilePath[lastBackslash - currentCodeFilePath + 1] = L'\\0';
13.
14.    wprintf(L"Listing directories and files in D:\\Documents\\n(Current Code File Location: %ls)\\n",
currentCodeFilePath);
15.
16.    wchar_t logFileName[FILENAME_MAX];
17.    _snwprintf(logFileName, FILENAME_MAX, L"%lsbin\\1.txt", currentCodeFilePath);
18.
19.    // Check if the 'bin' folder exists, if not, create it
20.    wchar_t binFolderPath[FILENAME_MAX];
21.    _snwprintf(binFolderPath, FILENAME_MAX, L"%lsbin", currentCodeFilePath);
22.    if (!CreateDirectoryW(binFolderPath, NULL))
23.    {
24.        DWORD error = GetLastError();
25.        if (error != ERROR_ALREADY_EXISTS)
26.        {
27.            wprintf(L"Failed to create 'bin' folder: %ls\\n", binFolderPath);
28.            return 1;
29.        }
30.    }
31.
32.    FILE *logFile = _wfopen(logFileName, L"a"); // Open in "append" mode instead of "write" mode
33.    if (logFile)
34.    {
35.        listDirectoriesAndFilesInRoot(L"D:\\Downloads", logFile);
36.
37.        fclose(logFile);
38.        wprintf(L"Successfully logged to %ls.\\n", logFileName);
39.
40.        // Read and print the contents of the log file to the console
41.        wprintf(L"\\nLogged Contents:\\n");
42.        FILE *readLogFile = _wfopen(logFileName, L"r");
43.        if (readLogFile)
44.        {
45.            wchar_t buffer[512];
46.            while (fgetws(buffer, 512, readLogFile))
47.            {
48.                wprintf(L"%ls", buffer);
49.            }
50.            fclose(readLogFile);
51.        }
52.        else
53.        {
54.            wprintf(L"Failed to read %ls.\\n", logFileName);
```

```

55.     }
56. }
57. else
58. {
59.     wprintf(L"Failed to open or write to %ls.\n", logFileName);
60. }
61.
62. return 0;
63. }
64.
65. void listDirectoriesAndFilesInRoot(const wchar_t *path, FILE *logFile)
66. {
67.     WIN32_FIND_DATAW findFileData;
68.     HANDLE hFind = FindFirstFileW((wchar_t *)L"D:\\Downloads\\*", &findFileData);
69.
70.     if (hFind == INVALID_HANDLE_VALUE)
71.     {
72.         wprintf(L"Error finding directories and files in: %ls\n", path);
73.         return;
74.     }
75.
76.     // Collect directories and non-directory files separately
77.     wprintf(L"Directories:\n");
78.     do
79.     {
80.         if (findFileData.dwFileAttributes & FILE_ATTRIBUTE_DIRECTORY)
81.         {
82.             if (wcscmp(findFileData.cFileName, L"..") != 0 && wcscmp(findFileData.cFileName, L"..")
83.             {
84.                 wprintf(L"Directory: %ls\n", findFileData.cFileName);
85.                 fprintf(logFile, L"Directory: %ls\n", findFileData.cFileName); // Log the directory
86.                 // to the file
87.             }
88.         } while (FindNextFileW(hFind, &findFileData) != 0);
89.
90.         // Close the handle and reopen it to reset the search
91.         FindClose(hFind);
92.         hFind = FindFirstFileW((wchar_t *)L"D:\\Downloads\\*", &findFileData);
93.
94.         // Collect non-directory files
95.         wprintf(L"Files:\n");
96.         do
97.         {
98.             if (!(findFileData.dwFileAttributes & FILE_ATTRIBUTE_DIRECTORY))
99.             {
100.                 wprintf(L"File: %ls\n", findFileData.cFileName);
101.                 fprintf(logFile, L"File: %ls\n", findFileData.cFileName); // Log the file to
102.                 // the file
103.             }
104.         } while (FindNextFileW(hFind, &findFileData) != 0);
105.         FindClose(hFind);
106.     }

```

Untitledemail2.txtlog.txt

FileEditView

File: Subject_ request for financial assistance and demonstration of programming skills (1).pdf

File: Swae Lee LIVE @ Rolling Loud LA 2021.mp4

File: Switch between cameras in blender (Add And Use Multiple Cameras).mp4

File: Switch between cameras in blender (Add And Use Multiple Cameras).vtt

File: Symbol Used in Hydraulic And Pneumatic system (Flow Control Valve,Accumulator,Pipe Lines,Heat Ex).mp4

File: TFT LCD IL934I Simulation with arduino.mp4

File: THAT SOUNDS FILTHY! MAKING AN EDM TRAP BEAT IN FL STUDIO!.mp4

File: THAT SOUNDS FILTHY! MAKING AN EDM TRAP BEAT IN FL STUDIO!.vtt

File: The Beat Formula That Helped Kendrick Become KING.mp4

File: The Beauty of Lagrangian Mechanics (SoME2).vtt

File: The Beauty of Lagrangian Mechanics (SoME2).2.mp4

File: The Denavit-Hartenberg Convention - Robotic Systems.vtt

File: The Denavit-Hartenberg Convention _ Robotic Systems.mp4

File: The equation of a wave - Physics - Khan Academy.mp4

File: The equation of a wave - Physics - Khan Academy.vtt

File: The Equation That Explains (Nearly) Everything!.vtt

File: The fastest spacecraft ever launched.mp4

File: The Geometry of Music.mp4

File: The Latest in AI Music & How to Create Your Own.mp4

File: The Math behind (most) 3D games - Perspective Projection.mp4

File: The Math behind (most) 3D games - Perspective Projection.vtt

File: The Math Behind Music and Sound Synthesis.mp4

File: The Math Behind Music and Sound Synthesis.vtt

File: The Mathematical Problem with Music, and How to Solve It.mp4

File: THE MONEY WILL COME BY ITSELF. The Proven Way to Wealth John D. Rockefeller.mp4

File: The Pythagorean comma.mp4

File: The Pythagorean comma.vtt

File: The Shady Past Of Cardi B.mp4

File: The Shady Past Of Cardi B.vtt

Ln 1, Col 1

100%

Windows (CRLF)

ANSI