Baimakanov Arsen CS-2113

## Report: Develop a file archiver utility

The necessary header files are included: stdio.h for standard input/output functions, stdlib.h for memory allocation, string.h for string manipulation, and windows.h for Windows-specific functions.

The macro BUFFER\_SIZE is defined to set the size of the buffer used for reading output from the powershell command.

The main() function begins.

Two character arrays, sourcePath and destinationPath, are declared to store the paths entered by the user for the source file or folder and the destination archive file.

The user is prompted to enter the path of the file or folder to be archived using printf(). The input is read using fgets() and stored in the sourcePath array. The trailing newline character is removed by replacing it with a null character.

Similarly, the user is prompted to enter the path and name for the archive file using printf(), and the input is read into the destinationPath array. Again, the trailing newline character is removed.

A new character array zipCommand is declared to store the command that will be executed using powershell. The snprintf() function is used to format the command by inserting the source path and destination path into the command string.

The user is informed that the files are being archived by printing a message to the console.

The powershell command is executed by opening a pipe using \_popen() with the zipCommand as the command to execute and "r" as the mode to read the output.

If the pipe fails to open (i.e., \_popen() returns NULL), an error message is printed, and the program exits with a return value of 1.

A loop is started to read the output from the pipe using fgets(). Each line of output is read into the buffer array, and if it is not NULL, it is printed to the console using printf().

Once all the output has been read, the pipe is closed using \_pclose().

A success message is printed to the console.

The main() function ends, and the program terminates.

The code relies on the powershell command and the Compress-Archive cmdlet to perform the file compression. It uses Windows-specific functions and features to interact with the operating system and execute the necessary commands.

**CODE:**

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include <windows.h>

#define BUFFER\_SIZE 1024

int main()

{

char sourcePath[MAX\_PATH];

char destinationPath[MAX\_PATH];

char zipCommand[MAX\_PATH + 20];

printf("Enter the path of the file or folder to be archived: ");

fgets(sourcePath, sizeof(sourcePath), stdin);

sourcePath[strcspn(sourcePath, "\n")] = '\0';

printf("Enter the path and name for the archive file (including .zip extension): ");

fgets(destinationPath, sizeof(destinationPath), stdin);

destinationPath[strcspn(destinationPath, "\n")] = '\0';

snprintf(zipCommand, sizeof(zipCommand), "powershell Compress-Archive -Path \"%s\" -DestinationPath \"%s\"", sourcePath, destinationPath);

printf("Archiving files...\n");

FILE\* pipe = \_popen(zipCommand, "r");

if (pipe == NULL)

{

printf("Failed to execute compression command.\n");

return 1;

}

char buffer[BUFFER\_SIZE];

while (fgets(buffer, BUFFER\_SIZE, pipe) != NULL)

{

printf("%s", buffer);

}

\_pclose(pipe);

printf("Files archived successfully.\n");

return 0;

}