CS 3100 Command Interpreter Project
Part 2: Simple Operating System Calls and Directory Access

This assignment continues building the Weber Shell by completing most of the internal functions. Proceed one function at a time by removing the appropriate comments from the `wsh::interpret()` method and then completing and testing the corresponding function.

The simple system calls used in this assignment parallel system calls provided by other operating systems. For example, Windows provide similar functions whose names only differ by adding a leading underscore or adding additional file protection information.

Assignment

Copy `wsh.cpp`, `wsh.h`, `Makefile` and the folder `savedir` from `/var/classes/cs3100/lab2` and complete the remaining functions outlined in the `wsh` class. If your CWD is `lab2`, use this command:

```
cp -r /var/classes/cs3100/lab2 .
```

1. Common features for all functions
   a. Validate that each `wsh` function has the correct number of arguments. Prints "`Invalid argument count`" if the count is incorrect.
   b. Each system call you will use in this assignment returns a value that indicates success or failure. Check this value and print an appropriate error message if the call fails, using `perror()`.
   c. If the command succeeds, there should be no messages displayed.
   d. Do not echo the command if it has been implemented. Only echo unknown commands.
2. Complete the copy function
   a. Verify that all files opened with `istream` or `ostream` opened without error or use `perror()` to display the error message.
   b. The copy function should correctly copy both text and binary files
3. Using man pages, look up the syntax and return value of the following system calls and library functions. Look for these system calls in man section 2 (e.g. `man 2 unlink`).
   a. `unlink`
   b. `remove`
   c. `mkdir`
   d. `rmdir`
   e. `stat`
   f. `opendir`
   g. `readdir`
   h. `perror`
4. New `wsh` commands you are to implement:
   a. `view file`
      i. prints file to the screen (text files only, binary files will print garbage)
      ii. similar to the Windows type or the Unix/Linux cat commands

b. `copy source dest`
   i. copies source to dest and additionally supports recursively copying a folder if source is a folder
   ii. open both files in binary mode and copy source to dest until end of file.
   iii. complete the private `void wsh::copy(char* src, char* dest)` method to actually copy the file (remove the printing of the command).
   iv. if source is a directory, call this special private method `wsh::copy`. The system calls and the recursion needed to solve this problem are similar to the `rmdir -s` operation written in class.
c. `del file`
d. `ren file1 file2`
   i. renames `file1` to `file2`
e. `mkdir folder1`
   i. makes an new directory named `folder1`
f. `rmdir folder1`
   i. removes directory named `folder1`
   ii. the directory is not removed if it is not empty or is the current directory (print out an error message using `perror()`)
g. `rmdir -s dir1`
   i. removes `dir1` (including all files and subdirectories)
   ii. this operation will be written by the instructor (with your help) in class.

5. Ensure your completed and tested `wsh.cpp` and `wsh.h` are found in `~/cs3100/lab2` on `icarus` for grading and mark the Lab 2 assignment complete.

Grading

I often use scripts to help grade programs. Please note the following:

- My scripts use the `g++` compiler on icarus to compile your programs. If you use a different compiler or a different Linux system to create your programs, please test them on icarus before submitting them for grading.
- The automated test bed will fail if you do not follow the naming instructions, so please be sure that the programs are named correctly.

Here is how you earn points for this assignment:

FEATURES	POINTS
Must-Have Features	
Files are named correctly and found in their proper place on `icarus`	5
Compiles without errors or warnings	5
Correctly displays the appropriate prompt	5
Exits properly when `"exit\n"` is typed	5
Required Features	
For **unknown** commands only, echos all words typed on the commandline with one space between them	5
`view` command allows exactly one operand	5
`view file` displays the contents of `file` on the screen	10
`copy` command allows exactly two operands	5
`copy file1 file2` copies the contents of `file1` to `file2`	10
`copy folder1 folder2` copies `folder1` and all of its contents to `folder2`	20
`copy file1 file1` fails because both file names are named the same	5
`copy file1 file2` fails with `perror()` message if any error occurs	5
`copy folder1 folder2` fails with `perror()` message if any error occurs	5
`del` command allows exactly one operand	5
`del file1` deletes `file1`	10
`del file1` fails with `perror()` message if any error occurs	5
`ren` command allows exactly two operands	5

`ren file1 file2` renames `file1` to `file2`	10
`ren file1 file2` fails with `perror()` message if any error occurs	5
`mkdir` command allows exactly one operand	5
`mkdir dir1` creates empty folder `dir1`	10
`mkdir dir1` fails with `perror()` message if attempt to create `dir1` fails	5
`rmdir` command allows exactly one operand	5
`rmdir dir1` deletes empty folder `dir1`	10
`rmdir dir1` fails with `perror()` message if attempt to remove `dir1` fails	5
`rmdir -s` command allows exactly one operand after the `-s`	5
`rmdir -s dir1` deletes folder `dir1` and all of its contents	20
`rmdir -s dir1` fails with `perror()` message if attempt to remove `dir1` fails	5
Grand Total	200