HW 2: Finding Files

You must create a file-finding program that will find all files whose names contain a given substring, within the current directory and all its subdirectories. The source code file should be called `filefinder.c`.

The program will take two command-line arguments. The first is the directory to search, which may also be `.` (the current directory) or `..` (the parent directory). The second is a string containing a pattern to match. Any file or directory with a name that contains the pattern as a substring should be printed out, in the given directory or any of its subdirectories. All such names should be printed out with their full locations relative to the current directory. Subdirectory names should be printed out using `/`, as shown in the example. The `$` indicates the command prompt, and the program has been compiled into the executable `filefinder`.

```
$ filefinder . pdf
important-paper.pdf
letter to grandma.pdf
subdir/pdf-list.txt
subdir/secret-artwork.pdf
```

If the user does not enter two command-line arguments, the program should print a helpful error and exit. If the user enters a directory that does not exist or cannot be opened, the program should likewise print a helpful error and exit. If the user enters a search string that does not match any file, the program should exit without printing anything at all.

Your finder should follow POSIX standards, as implemented in Linux, MacOS, and BASH. Specifically, displayed directories should be separated by forward slashes (not backslashes, as Windows does). In addition, you will likely find subdirectories called `.` and `..` within each directory. These should not be followed, since they link to the current directory and the parent directory, respectively.

You will likely want to `#include` the C header file `dirent.h`, which defines a struct and several functions that are applicable to this assignment.

- `struct dirent`, which is short for <u>dir</u>ectory <u>ent</u>ry. It will hold information on a directory entry—usually a file or directory. Useful fields that it contains are:
  - `char d_name[]`: A string representing the name of this directory entry (maximum of 256 characters).
  - `ino_t d_ino`: A unique serial number for this entry. On most systems, `ino_t` may be treated as a `long int`.
  - `unsigned char d_type`: A small integer representing the type of this entry. Common values include `DT_DIR` for a directory, `DT_LNK` for a link to a file or directory, `DT_REG` for a regular file, and `DT_UNKNOWN` if the type cannot be determined. Some systems may not support `d_type`.
- `struct DIR *opendir(const char *name)` This takes a string holding a directory

name, and opens the directory. A `DIR` is a struct that holds information on the opened directory. Returns `NULL` if there is an error opening it.

- `struct dirent *readdir(DIR *dirp)` Returns the next directory entry from the given `DIR*`. Returns `NULL` if there are no remaining entries.
- `int closedir(DIR *dirp)` Closes the `DIR` pointed to by the argument. Every open directory should be closed when the program is done with it. Returns `0` on success, and `-1` if there is an error.

If you need more information on the contents of `dirent.h`, you can find the man pages online.

Remember that this will be graded on a Linux computer. You might want to simulate a Linux machine in order to ensure that it works.

Remember that directory structures are trees. As such, the best approach to traverse it is likely recursive. You will need to be careful to avoid infinite loops.