CS 214 Homework 2

Fall 2022

Introduction

In this assignment, you'll be working with files and directories in Linux. Your task is to write the following C programs:

- `ls`: list contents of a directory
- `find`: find files/directories matching a pattern
- `tree`: print files/directories as a tree

Notes:

- Please be careful to follow the I/O formats exactly.
- You can assume that inputs will be correctly formatted and all file/directory names will consist only of letters, numbers, and periods.

You can work in small groups (1 – 3 people). Please include a README file with your code that contains all partners' names and netIDs. Only one person from each group should submit the assignment.

For the input/output examples given in this document, assume you have the following directory structure:

```
cs214
├── hw1
│   ├── factor.c
│   ├── grep.c
│   ├── monster.c
│   ├── sort.c
│   └── uniq.c
└── hw2
    ├── find.c
    ├── ls.c
    ├── Makefile
    └── tree.c
```

ls with no command-line options

The `ls` command prints all files/directories in the current working directory.

Running it in your hw2 directory, you might see this output:

```
find.c
ls.c
Makefile
tree.c
```

The files/directories should be sorted in case-insensitive lexicographic order:

$$. < 0 < 1 < \cdots < 9 < \mathrm{a} < \mathrm{b} < \cdots < \mathrm{z}$$

ls with the `-l` command-line option

If the user invokes `ls -l` (lowercase "l" for "long"), you should print a "long format" with extra information about each file.

```
-rw-rw-r-- bob users 1562 Sep 29 12:00 find.c
-rw-rw-r-- bob users 1024 Sep 29 12:05 ls.c
-rw-rw-r-- bob users 176 Sep 28 11:27 Makefile
-rw-rw-r-- bob users 2044 Sep 27 18:23 tree.c
```

The first column contains a 10-character permissions string.

- the first character is '-' for files, 'd' for directories
- the next three are read, write, and execute permissions for the user
- the next three are read, write, and execute permissions for the group
- the next three are read, write, and execute permissions for others

Permissions should be denoted by a 'r', 'w', or 'x' if present, or '-' otherwise.

The second and third columns show the file owner's user name and group name (or user ID / group ID if a name can't be found).

The next column shows the file size in bytes.

Next is the file's modification time (mtime) formatted as in the example above.

Finally the filename is listed.

You may find `strftime`, `getpwuid`, and `getgrgid` useful.

find

The `find` command takes a pattern as a command-line argument and recursively searches through directories to find a filename matching that pattern. It should print a relative path starting with "`./`" for every file/directory that matches.

For example, running `./find ls.c` from within the `hw2` directory should print `./ls.c`. If run from the parent directory, the output would be `./hw2/ls.c`.

There may be multiple matches. If we run `./find .c` from within the `hw2` directory, we should see:

```
./find.c
./ls.c
./tree.c
```

The output does not need to be sorted. If nothing matches, it shouldn't print anything. Pattern matching should be case sensitive.

tree

The `tree` command prints all files/directories contained in the current directory as a tree. If we run `./tree` from the `cs214` directory, we should see:

```
.
- hw1
  - factor.c
  - grep.c
  - monster.c
  - sort.c
  - uniq.c
- hw2
  - find.c
  - ls.c
  - Makefile
  - tree.c
```

The first line of output is always "`.`" to denote the current directory. `tree` recurses into directories to print all files/directories within them. For each subdirectory, two spaces of indentation are added. For example, if we had a file `hw1/test/src/foo.c`, `tree` would print:

```
.
- hw1
  - test
    - src
      - foo.c
```

Files/directories within each subdirectory should be sorted with the same case-insensitive lexicographic order as `ls`.

Compiling and testing

You should use similar CFLAGS to gcc as in homework 1, e.g.:

```
-g -Wall -Wvla -fsanitize=address
```

You should create a Makefile so that running "make" or "make all" builds all of the programs.

Submission

If you develop on your local machine, please be sure to test your code on ilab before submitting.

Please submit the assignment on Canvas as a tar file `hw2.tar` that, when expanded, produces a `hw2` directory:

```
hw2
├── find.c
├── ls.c
├── Makefile
└── tree.c
```

It can also contain any other files necessary to build your programs. If you prefer, you can put each program in its own directory.

If you work in a group, please include a `README.txt` file with the names and netIDs of everyone in your group. Only one person needs to submit on Canvas.