CSE 102 Programming Assignment 8

DUE

December 31, 2019, 23:55

Description

- This is an individual assignment. Please do not collaborate.
- If you think that this document does not clearly describes the assignment, ask questions before its too late.

Be careful with file names. You won't given a chance to correct any mistakes.

- Your program reads two files:
  - `files.txt`
  - `commands.txt`
- According to content in `files.txt`, the program **dynamically** creates a directory structure and evaluates the commands listed in `commands.txt`.
- Your program prints the list of files and directories with their full addresses to a file called `output.txt`.

`files.txt`

- Each line is a file or directory name. This file lists all the necessary information for you to create a full directory structure. There won't be any errors but the order can be arbitrary. You have to figure out the root directory and the rest of the hierarchy from this file. For this initial file, you can assume all the directory names will be unique. You can read the file line by line and create independent structures for each line, then you can combine all the created structures in order to create the final structure. Assume, there isn't any empty directory. Each line ends with a file name.
- Example:
- Below is the contents of a `files.txt` file:

```
directory1/directory2/file1
directory2/file3
directory0/directory1/file5
directory0/file4
```

- For this example, the structure is as follows:

```
directory0
    directory1
        directory2
            file1
            file3
        file5
    file4
```

`commands.txt`

This file includes several commands which work on the directory structure you read from `files.txt`. Each line is a command. The following should be recognized:

```
copy A B
move A B
delete A
cd A
```

copy A B

This command has two operants: `A` and `B`. `B` is a directory. `A` can be a directory or a file. This command copies A and places it under directory `B`. If `A` is a directory, all the structure under `A` is copied under `B`. `A` and `B` can be provided as full directory or relative directory.

move A B

Similar to `copy A B`. This moves `A` under `B` and deletes the original copy of `A`

delete A

Removes `A` from the structure.

cd A

This command changes the current directory to `A`. `A` is definitely a directory not a file. There are special replacements for `A`:

- `..` : changes the current director to the parent directory.(If there is no parent, current directory does not change)
- `/` : changes the current directory to root. Root directory is the top directory. There isn't a parent directory above the root directory.

Example:

Assume that your program reads the example `files.txt` given above. Current directory is the top(root) directory which is `directory0`

- Issuing the command `copy /directory0/directory1/directory2/file1 /directory0`, copies `file1` and places it under `directory0`. Following is the result of this command:

```
directory0
    directory1
        directory2
            file1
            file3
        file5
    file1
    file4
```

- similarly, the command `copy file4 directory1` copies `file4` to the directory `directory1`. Here, current directory is `directory0` so, `file4` and `directory1` are given as relative addresses. After this command, the structure changes to the following version:

```
directory0
    directory1
        directory2
            file1
            file3
```

```
        file4
        file5
    file1
    file4
```

- If there is a command `delete directory1`, it will change the structure as follows:

```
directory0
    file1
    file4
```

`output.txt`

This file lists all the files and directories with their full addresses.

If your directory structure is as follows:

```
directory0
    directory1
        directory2
            file1
            file3
        file4
        file5
    file1
    file4
```

You should create the following `output.txt` file:

```
directory0
directory0/directory1
directory0/file1
directory0/file4
directory0/directory1/directory2
directory0/directory1/file4
directory0/directory1/file5
directory0/directory1/directory2/file1
directory0/directory1/directory2/file4
```

- The order is not important.

Remarks

- This assignment is about using linked lists. You should use structures and/or unions.
- You need to keep the track of `current directory` and issue the commands.
- You can assume that all the files and directories under the same parent directory will have different(unique if in the same directory) names.
- Do not submit your code without testing it with several different scenarios.
- Write comments in your code.
- Do not submit any of the files you used for testing.
- Do not submit your output file.

Turn in:

- Source code of a complete C program. Name of the file should be in this format: `<full_name>_<id>.c`.
- Example: `gokhan_kaya_000000.c`. Please do not use any Turkish special characters.
- You don't need to use an IDE for this assignment. Your code will be compiled and run in a command window.

- Your code will be compiled and tested on a Linux machine(Ubuntu). GCC will be used.
- Make sure you don't get compile errors when you issue this command : `gcc <full_name>_<id>.c`.
- A script will be used in order to check the correctness of your results. So, be careful not to violate the expected output format.
- Provide comments unless you are not interested in partial credit. (If I cannot easily understand your design, you may loose points.)
- You may not get full credit if your implementation contradicts with the statements in this document.

Late Submission

- Not accepted.

Grading (Tentative)

- `Max Grade : 100`.
- Multiple tests(at least `5`) will be performed.

All of the followings are possible deductions from `Max Grade`.

- `#define HARD_CODED_VALUES -10`. (Do **NOT** use hard-coded values)
- No submission: `-100`. (be consistent in doing this and your overall grade will converge to `N/A`) (To be specific: if you miss `3` assignments you'll get `N/A`)
- Compile errors: `-100`.
- Irrelevant code: `-100`.
- Major parts are missing: `-100`.
- Unnecessarily long code: `-30`.
- Inefficient implementation: `-20`.
- Using language elements and libraries which are not allowed: `-100`.
- Not caring about the structure and efficiency: `-30`. (avoid using hard-coded values, avoid hard-to-follow expressions, avoid code repetition, avoid unnecessary loops).
- Significant number of compiler warnings: `-10`.
- Not commented enough: `-5`. (Comments are in English).
- Source code encoding is not `UTF-8` and characters are not properly displayed: `-5`. (You can use 'Visual Studio Code', 'Sublime Text', 'Atom' etc... Check the character encoding of your text editor and set it to `UTF-8`).
- Missing or wrong output values: `Fails the test`.
- Output format is wrong: `-30`.
- Infinite loop: `Fails the test`.
- Segmentation fault: `Fails the test`.
- Fails 5 or more random tests: `-100`.
- Fails the test: `deduction up to 20`.
- Prints anything extra: `-30`.
- Requires space/newline at the end of the file: `-20`.
- Requires specific newline marking (CR/LF): `-20`.
- Unwanted chars and spaces in output: `-30`.
- Submission includes files other than the expected: `-10`.
- Submission does not follow the file naming convention: `-10`.
- Sharing or inheriting code: `-200`.