EE442 HOMEWORK 1

Directory Listing

Due: April 9, 2018, 23:55

Aycan Doğa Hakyemez **ARC-201** **hdoga@metu.edu.tr**

Submission

- Send your homework compressed in an archive file with name "**eXXXXXXX_ee442_hw1.tar.gz**", where X's are your **7-digit student ID number**. You will **not** get full credit if you fail to submit your project folder as required.
- Your work will be graded on its correctness, efficiency, clarity and readability as a whole.
- You should insert comments in your source code at appropriate places without including any unnecessary detail.
- Late submissions are welcome, but are penalized according to the following policy:
    - 1 day late submission : HW will be evaluated out of 70.
    - 2 days late submission : HW will be evaluated out of 50.
    - 3 days late submission : HW will be evaluated out of 30.
    - Later submissions : HW will NOT be evaluated.
- The homework must be written in **C** (and **not** C++ or any other language).
- You should **not** call any external programs in your code.
- **Check** what you upload in case of corrupted files, sending wrong files or leaving out necessary files.
- The homework is to be prepared **individually**. It is **not** allowed to prepare the homework as a group.
- The design must be your original work. If you make use of a code from the Web, **reference** the website in the comments.
- **METU honor code is essential**. Do **not** share your code. Any kind of involvement in cheating will result in a **zero** grade, for **both** givers and receivers.

Introduction

In this homework, you are asked to implement a program which lists the files in a directory and write the list to another file. The objective is to learn about some basic system calls.

Program argument

The program will take one argument which is the path of the directory to be listed. The path may be relative or absolute. For simplicity, you can assume the path only has ASCII characters and does not include whitespace.

Listing file

The listing file should be in the parent directory of the input directory. Its name should be the name of the directory with the extension ".list". The file should have the following format:

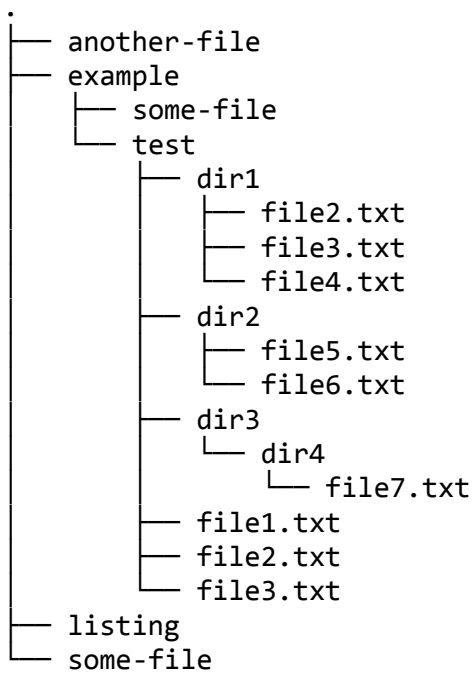
```
<Your 7-digit student ID number>
<Absolute path of the input directory>

<filename relative to input directory >
<filename relative to input directory >
<filename relative to input directory >
.
.
.

<filename relative to input directory >
<filename relative to input directory >
<filename relative to input directory >
.
.
.
```

If this file does not exist, the program should create it. Otherwise, the program should check the first two lines to ensure they are proper. If they are not proper, the file should be overwritten.

When the program is run for a directory, after checking the first two lines, listing file should be filled with the names of the files in the directory, including the files in the subdirectories. Subsequent runs of the program should only append the files, separated by an empty line from the previous entries.

Example

Suppose a directory in the following structure:



`listing` is the executable file. After the program is run with `./listing ./example/test`, the contents of `test.list` will be:

```
1234567
/home/ee442/HW1/example/test

dir3/dir4/file7.txt
dir2/file5.txt
dir2/file6.txt
dir1/file4.txt
dir1/file3.txt
dir1/file2.txt
file3.txt
file1.txt
file2.txt
```

The files need not to be ordered in any way.

After adding two files to test directory (`file8.txt` and `file9.txt`), the directory looks like this (note that `test.list` is created by `listing` program):

```
.
├── another-file
├── example
│   ├── some-file
│   ├── test
│   │   ├── dir1
│   │   │   ├── file2.txt
│   │   │   ├── file3.txt
│   │   │   └── file4.txt
│   │   ├── dir2
│   │   │   ├── file5.txt
│   │   │   └── file6.txt
│   │   ├── dir3
│   │   │   └── dir4
│   │   │       └── file7.txt
│   │   ├── file1.txt
│   │   ├── file2.txt
│   │   ├── file3.txt
│   │   ├── file8.txt
│   │   └── file9.txt
│   └── test.list
└── some-file
```

After running the program again with `./listing ./example/test`, the contents of `test.list` are as follows:

```
1234567
/home/ee442/HW1/example/test

dir3/dir4/file7.txt
dir2/file5.txt
dir2/file6.txt
dir1/file4.txt
dir1/file3.txt
dir1/file2.txt
file3.txt
file1.txt
file2.txt

dir3/dir4/file7.txt
dir2/file5.txt
dir2/file6.txt
dir1/file4.txt
dir1/file3.txt
dir1/file2.txt
file3.txt
```

```
file1.txt
file9.txt
file2.txt
file8.txt
```

Note that the previous order may change, you do not have to force any ordering.

Remarks

- You can compile your code with the command `gcc logger.c -o listing`.
- You can find information about headers, functions and types here.
- Send only the source code. Do not send any executable, since your code will be recompiled.