Programming Assignment (PA) - 4
File Systems

CS 307 - Operating Systems (Fall 2022)
21 December 2022
DEADLINE: 4 January 2023, 23:55

1 Introduction

In this programming assignment you will make use of some of the file system utilities and functions you have seen in class. As you've seen in lectures, when you open a file, there is a file pointer associated with that open file and as you read the file, the position of the file pointer changes accordingly. Moreover, you can modify the file pointer directly using `lseek` command. What you will be doing in this programming assignment will be highly associated with this pointer. You will also need to iterate over entries of a directory, differentiate different entry types like directories and files and go over all sub-directories recursively.

2 Problem Description

For this PA, you will write a program that accepts the current working directory (the directory that contains this executable) as the root directory, traverses all of its sub-tree (all the directories and files reachable from this root directory) and modifies files with a `txt` extension. You will assume that there is a simple `database.txt` file inside the root directory where you can find several information about some people. You can think of this database.txt file as a table in a database. The rows will be in separate lines and each column will be separated with a single space. The description of the columns are like this:

First column: Gender of the person. 'f' represents female and 'm' represents male.
Second column: First name of the person. e.g: 'Ayse'
Third column: Last name of the person. e.g: 'Yilmaz'

In the incorrectly written txt files, some of the last names and titles (Ms. and Mr.) have been written incorrectly. For example imagine there is a row like this in the database.txt like:

m Ahmet Yilmaz

In one of the `txt` files there might be written something like "... Ms. Ahmet Sasmaz was going to arrive late however...". As it could be seen from the database the title Ms. should have been Mr. since in the database the gender column is 'm' which means that this person is a male and should be referred as"Mr.". The last name of this person has also been written incorrectly, as it should have been Yilmaz and as this is how it was declared in the database.txt file. Note that we always assume that the first name of these people are written correctly, so you might take that as a small hint for what you will search for in the txt file, since the other parts are probably written incorrectly. You can safely assume that there will be no two rows in the database file with the same name.

So in conclusion, your code should be able to iterate through the directory it is in and all of the sub-directories recursively and should correct all of the txt files that are written incorrectly in a manner that the last name and titles are not matching that of those in the database.txt file. An example of what you should expect from the database.txt file:

f Esma Gonul
f Zeynep Sasmaz
m Enes Koyuncu
f Ayse Ozyilmaz
f Alev Yildirim
m mehmet Olca

3 Implementation Details

There are some specific functions we want you to use. The required functions are listed below:

- **fseek**: As you've seen in class, you can manually change the direction of the file pointer. Normally when you read a file sequentially the file descriptor's position also increases sequentially. However,

in this programming assignment you will be searching the names of the people listed in the database and once you find the specific name in the database you have to go backwards to check whether the title(Ms. or Mr.) is correct and you also have to check whether the last name is written correctly. So in order to go backwards and forward with the descriptor, you **must** use fseek. fseek is a simple and direct wrapping of the `lseek` function we have seen in the lectures. The difference is that `lseek` is specific to UNIX whereas fseek works in any OS.

- **fputs**: fputs is useful to place or overwrite the text wherever the file pointer is at. For example if the file pointer is at location 4, and if you write fputs("some text", fptr), starting from the 4^{th} character in the txt file, it will overwrite the next 9 characters as "some text". Keep in mind that fputs does not append, it overwrites on the existing text. For example assuming fptr is at location 0:

"This is PA4 for CS307"
fputs("okay", fptr)
"Okay is PA4 for CS307"

Functions that are not a must but might be useful to use:

- **ftell**: As it could be understood from it's name, ftell gives the current location of the file pointer. This way you keep track of where your file descriptor is.

4 Important Notes

- The incorrectly written last names will be the same length as the correct last names for that person. For example if normally in the database someone's last name is "Yilmaz" then the incorrectly written last name will be something like "Sasmaz". As it could be seen they are the same length, both of them are of length 6. This is to make your lives easier, as fputs overwrites from the position where the file descriptor is pointing at and even if you open the file with the append mode, you will be able to write at the end of the file, which is not what we want you to do.
- As you might have also figured from the note above, you have to open the file in "r+" mode and not in "a" or "w", as the append mode which is "a", will always carry the file descriptor to the end of the file whenever you want to write something in the file with fputs and the write mode which is "w" will erase all of the previously written content in the txt file, which is something we definitely don't want you to do.
- The only thing you should do is to move the location of the file pointer using fseek and write to the correct positions with fputs. You are not allowed to create another txt file with the same name and write the correct content there. If you go with this approach you will get 0.
- Although it has a `txt` extension, you must not modify the `database.txt` placed directly under the root directory. If there are other `database.txt`'s under other sub-directories (other than the root), they must be corrected.

5 Sample Input and Outputs

database.txt :

f Esma Gonul
f Zeynep Sasmaz
m Enes Koyuncu
f Ayse Ozyilmaz

f Alev Yildirim
m Mehmet Olca

Incorrectly written some_random_file.txt:

Ms. Enes karaman was on his way to the office. However there was too much traffic and this time he knew that **Ms. Ayse Akyilmaz** will not tolerate him arriving late to work, once again...

Corrected version of some_random_file.txt:

Mr. Enes Koyuncu was on his way to the office. However there was too much traffic and this time he knew that **Ms. Ayse Ozyilmaz** will not tolerate him arriving late to work, once again...

Assume that your root directory is `"."`. Assume that root directory has the sub-directories `a` and `b` and contains files `"foo.txt"`, `"bar.txt"`, `"database.txt"` and `"foo.exe"`. Also, assume that `b` only contains a file called `"database.txt"` and `c` contains a directory called `d` and a file called `"bar.txt"`. Lastly, `d` contains only three files `"database.txt"`, `"foo.txt"` and `"database.bpl"`. Your program must correct all these files: `"./foo.txt"`, `"./bar.txt"`, `"./b/database.txt"`, `"./c/bar.txt"`, `"./c/d/database.txt"`, `"./c/d/foo.txt"`.

6 Submission

You are expected to submit a zip file named $< YourSUUserName >$_PA4.zip until the deadline.
The content of the zip file is as follows:

- **report.pdf**: In you report clearly explain the process that you use to solve this problem.
- **corrector.c**: You C implementation file.

7 Grading

Some parts of the grading will be automated. If automated tests fail, your code will not be manually inspected for partial points. Some students might be randomly called for an oral exam to explain their implementations and reports.

Submissions will be graded over 100 points:

1. **Compilation (10 pts)**: Your program compiles without an error. When we run the program, we do not get any run-time errors.
2. **Report (20 pts)**: Write step by step what your are doing in your C implementation. Points will be deducted from reports which were written in an unclear, disorganized and poorly manner.
3. **Single File: (30 pts)**: Your program correctly works when there is a single `txt` file under the root (except `database.txt`) and the root directory has no sub-directories.
4. **Multiple files (10 pts)**: Your program correctly works when there are more than one `txt` files under the root (except `database.txt`) and the root directory has no sub-directories.
5. **Single Child Directory (10 pts)**: Your program correctly works when there are more than one `txt` files under the root (except `database.txt`) and the root directory has a single sub-directory with multiple `txt` files inside. However, the sub-directory has no sub-directories.
6. **Multiple Children (10 pts)**: Your program correctly works when there are more than one `txt` files under the root (except `database.txt`) and the root directory has multiple sub-directories with multiple `txt` files inside. However, the sub-directories have no sub-directories.

7. **More Than 1 Depth (10 pts)**: Your program correctly works when there are more than one `txt` files under the root (except `database.txt`) and the root directory has multiple sub-directories with multiple `txt` files inside. Sub-directories might have their own sub-directories with multiple `txt` files and so on. There is no limit on the depth of sub-directories.

Item 1 is a precondition for items $[3, 7]$. For $i \in [3, 6]$, item i is a precondition for item $i + 1$.