CS307 2022 SPRING PA5 – File Systems

UĞUR ÖZTUNÇ 28176

In the main part of the program, first, an integer named `databaseSize` is declared and set to the line count of the database.txt file by using `countLinesinFile()` function.

> **int countLinesinFile(char* fileName):**
> This function counts how many lines are there in the file given as parameter. It first opens the file and reads it line by line while increasing the counter at every line. After that it closes the file and returns the counter.

Then, a dynamic memory is allocated for a `struct Record` array by using malloc command and it is assigned to the pointer named database.

> **struct Record:**
> This struct is defined to keep the information of a person. It has a `char` for gender, and 2 `char*` for name and surname.

After defining essential variables a function called `createDatabase()` is called in order to get information of people from database.txt and save them into the database.

> **void createDatabase(struct Record* db, char* fileName):**
> This function takes the database array pointer and a database file name as arguments. It opens the database file and reads it line by line. At each line, a temporary `Record` is created and words are parsed into tokens and the first token is assigned to `tempRecord`'s gender, second is to its name, and the third one is to its surname. After all tokens assigned to `tempRecord`, `tempRecord` is saved to the database array. After all lines are completed and while loop ends, file is closed and function terminates.

After the database is created, `scanDirectory()` function is called in order to scan all files in root directory and its subdirectories (and subdirectories of them again and this goes on) for txt files to correct.

> **void scanDirectory(char* dirPath, struct Record* db, int* dbSize):**
> This function takes "./" `dirPath`, main database pointer, and `databaseSize` initally in main in order to scan all directories and correct txt files. At first, `dirPath` directory is opened with `opendir()` and necessary variables are defined (`dirent*` and `stat`). Then a while loop reads all elements in `dirPath` with a while loop uses `readdir()`. At each loop, first current element in `dirPath` is classified with `fstatat()` function and stored in a `stat` variable. Then some nested if blocks checks the properties of the current element. First, if current element is "." or ".." current loop finishes, otherwise if current element is a folder, which is checked with `S_ISDIR`, the name of the current element is appended at the end of the current `dirPath` (of course with a '/') and `scanDirectory()` function is called recursively with this new path in order to scan subdirectories. If current element is not a folder, than it must be a file. In this case, at first, the extension of the current file is taken and if it is a txt file, next if block checks whether this txt file is the original database.txt file in the root directory or not. If it is not the original database.txt file in the root directory, the name of this txt file is appended at the end of the current path and `correctFile` function is called with this path in order to correct the txt file. After all elements in current directory is scanned and while loop ends, directory is closed with `closedir()` function and function terminates.

`void correctFile(char* fileName, struct Record* db, int* dbSize):`

This function is called when a txt file is encountered in `scanDirectory()` function. After the txt file is opened with the given path (in “r+” mode), a `Record` pointer named `temp` is created, an integer named `status` is created and set to 0, a while loop reads the file word by word with `fscanf()` function. The status integer will help to indicate whether the current word is a part of a person name. At each loop:

- If `status` is 0 and the current word is a title (“Mr.” or “Ms.”), `status` is set to 1; otherwise nothing happens, loop continues to the next word.
- If `status` is 1, which means previous word was a title, a for loop checks whether the current word is the first name of any person in database or not. If so, `status` is set to 2 and the `temp` pointer is set to point the original `Record` of that person in database, otherwise `status` is set to 0.
- If `status` is 2, which means current word is a surname of a person in database, regardless of whether the title or surname is correct or not, they are both being overwrited with correct ones with `temp` pointer and some `fseek()` and `fputs()` operations. After correction is done, file pointer is set to where it was, `temp` is set to `NULL`, and `status` is set to 0.

After all words are checked and while loop ends, txt file is closed and function terminates.

In the main, after `scanDirectory()` function ends, the memory allocated for the `Record` array is freed and program terminates.