CSE 344

Homework #1 Report

Berru Lafcı

1901042681

Logic of the code:

In the main, I read commands in infinite while loop.

```
// Read a line from the terminal
bytes_read = read(STDIN_FILENO, command, sizeof(command));
```

According to the command coming from the terminal, I take the first word (with strtok) and enter the if-else blocks accordingly. And then, I tokenize the other words (with strtok) according to their commands. Example:

```
if (strcmp(token, "gtuStudentGrades") == 0) {

    filename = strtok(NULL, "\n");
```

```
else if(strcmp(token, "addStudentGrade") == 0) {
    filename = strtok(NULL, " ");
    studentName = strtok(NULL, " ");
    studentSurname = strtok(NULL, " ");
    grade = strtok(NULL, "\n");
```

I didn't use " (double-quote) for commands. So only 1 name and 1 surname are accepted as written in the pdf.

For the fork() and process creation part, I did all the forks inside functions except "gtuStudentGrades grades.txt" command. This fork happens in main.

I did the necessary jobs in the child. When the job finishes, I exited from the child then go back to the parent.

If you want to see the fork() happened successful or not, you can look at the log file. I wrote their ids accordingly. Also, when user enters "quit" it ends the program with the same parent id:

```
175  Fri Mar 22 14:04:58 2024 Main process started. ID: 1627
176  Fri Mar 22 14:05:10 2024 Child process finished: 1682, File created: grades.txt
177  Fri Mar 22 14:06:18 2024 Child process finished: 1744, Student grade added: Berru Lafci, AA
178  Fri Mar 22 14:12:43 2024 Child process finished: 1859, Student grade added: Nana Osaki, BB
179  Fri Mar 22 14:13:03 2024 Child process finished: 1936, Student grade added: Lee Chan, DC
180  Fri Mar 22 14:16:01 2024 Child process finished: 2008, Usage shown
181  Fri Mar 22 14:16:18 2024 Main process finished: 1627
```

You can run the program with “make run” and clean with “make clean”:

```
sh-5.2$ make run
gcc -Wall -Wextra -Werror -c gtuStudentGrades.c -o gtuStudentGrades.o
gcc -Wall -Wextra -Werror -o gtuStudentGrades gtuStudentGrades.o
./gtuStudentGrades
Enter a command, or type 'quit' to exit the program.
quit
sh-5.2$ make clean
rm -f gtuStudentGrades gtuStudentGrades.o
sh-5.2$
```

When unsuccessful command name happens, it calls the usage() function. Then waits for a command again:

```
gtuStudent grades.txt

Usage:
gtuStudentGrades grades.txt: Create file.
addStudentGrade grades.txt studentName studentSurname grade: Add student grade to file. Grade should be between AA and FF.
searchStudent grades.txt studentName studentSurname: Search student grade in file.
showAll grades.txt: Print all of the entries in the file.
listGrades grades.txt: Print 5 entries in the beginning of the file.
listSome numofEntries pageNumber grades.txt: Print numofEntries entries starting from pageNumber in the file.
sortAll grades.txt: Sort student name or grade, in ascending or descending order.
quit: Exit the program.
```

When invalid txt name happens, it gives open file error and writes it on log.txt:

```
showAll hhh
Error opening file: No such file or directory
```

```
215  Fri Mar 22 15:08:44 2024 Child process failed: 3570, Failed to show student grades: hhh
216
```

I used POSIX functions like open, read and write in necessary places:

```c
// Open the file for appending
int fd = open("log.txt", O_WRONLY | O_APPEND | O_CREAT, 0666);
if (fd == -1) {
    perror("Error opening file");
    exit(EXIT_FAILURE);
}
```

```c
// Read the file byte by byte
while ((bytes_read = read(fd, line, 1)) > 0) {
    // Check if a newline character is encountered
    if (*line == '\n') {
        *line = '\0'; // Null-terminate the string
```

```c
if(write(STDOUT_FILENO, buffer_print, strlen(buffer_print)) == -1) {
    perror("Error writing to terminal");
    exit(EXIT_FAILURE);
}
```

Logic of functions:

write_to_log:

This function appends a message to a log file named "log.txt" along with a timestamp.

- It forks a child process.
- In the child process, it opens the log file for appending, constructs the log message with a timestamp, writes the message to the file, and then closes the file before exiting.
- In the parent process, it waits for the child process to finish and checks if the child process terminated successfully.

sortAll:
These functions and the sortAll function are for sorting and printing student records either by name or grade in ascending or descending order, and then logging the sorting operation.

- compare_students_by_name: Compares two student records by their names. Used for sorting students in ascending order by name.
- compare_students_by_name_desc: Compares two student records by their names in descending order.
- compare_students_by_grade: Compares two student records by their grades. Used for sorting students in ascending order by grade.
- compare_students_by_grade_desc: Compares two student records by their grades in descending order.
- sortAll: Reads student records from a file specified by filename, sorts them based on criteria specified by isStudent (1 for sorting by name, 0 for sorting by grade) and isAscending (1 for ascending order, 0 for descending order), prints the sorted records to the terminal, and logs the sorting operation.

```
typedef struct {
    char name[50];
    char grade[5];
} Student;
```

Sorting logic:

- The sorting functions (compare_students_by_name, compare_students_by_name_desc, compare_students_by_grade, compare_students_by_grade_desc) are comparison functions used by the qsort function to determine the order of elements in the array.
- These functions take two const void pointers as arguments, which represent elements of the array to be compared.
- Within these functions, these pointers are cast to pointers of the Student struct type.

- The comparison functions then access the relevant fields (name or grade) of the Student structs pointed to by these pointers and compare them using strcmp.
- Depending on the desired sorting order (ascending or descending), these functions return the result of the comparison.

```c
// Compare function for sorting students by name in ascending order
int compare_students_by_name(const void *a, const void *b) {

    // Cast the void pointers to Student pointers
    const Student *student_a = (const Student *)a;
    const Student *student_b = (const Student *)b;

    // Compare the names
    return strcmp(student_a->name, student_b->name);
}
```

```c
// Sort the students
if (isStudent) {
    if (isAscending) {
        // printf("Sorting students by name in ascending order\n");
        qsort(students, student_count, sizeof(Student), compare_students_by_name);
    } else {
        // printf("Sorting students by name in descending order\n");
        qsort(students, student_count, sizeof(Student), compare_students_by_name_desc);
    }
}
```

addStudentGrade:
This function is designed to add a student's grade to a file. It takes three parameters: the filename of the file to which the grade will be added, the name of the student, and the grade itself.

- It forks a child process to handle the file writing operation, allowing the parent process to continue its execution.
- In the child process, it opens the specified file in append mode, writes the student's name and grade to it, and then closes the file.
- The parent process waits for the child process to finish executing. Upon completion, it checks if the child process terminated successfully or not.
- Logs the adding operation.

searchStudent:
This function searches for a student's grade in a file. It takes two arguments: the filename of the file to search in and the name of the student to search for.

- It forks a child process to perform the search.
- In the child process, it reads the file line by line, parsing each line to extract the student's name and grade. If the student's name matches the search query, it prints the student's name and grade and exits successfully.
- In the parent process. It waits for the child process to finish. It logs a message indicating whether the search was successful or not.

displayStudentGrades:

- This function displays all entries(showAll) or 5 entries (listGrades) in the specified file.
- It forks a child process to perform the file reading and displaying operation.
- The child process reads the file line by line and prints each line to the standard output.
- If isListGrades is set to 1, it only displays the first 5 entries. Otherwise, it displays all entries.
- After the child process finishes, the parent process logs a message indicating whether the operation was successful or not.

displaySomeStudentGrades:

- This function displays a specified number of entries from a specific page in the file.
- Similar to the previous function, it forks a child process to perform the file reading and displaying operation.
- The child process reads the file line by line, keeping track of the line number.
- It calculates which lines correspond to the specified page and number of entries and prints them to stdout:

```
// Calculate the line number and print the line if it is within the specified range
if(line_number > (pageNumber - 1) * numofEntries && line_number <= pageNumber * numofEntries) {
    //printf("%s\n", buffer);
    if(write(STDOUT_FILENO, buffer, strlen(buffer)) == -1) {
```

- After the child process finishes, the parent process logs a message indicating whether the operation was successful or not.

usage:
This function is designed to display usage instructions for a program. It forks a child process to print out these instructions while the parent process waits for the child to finish. After the child process prints the instructions, it exits, and the parent process logs whether the child process succeeded or failed in displaying the usage information.

Example run:

1-) gtuStudentGrades grades.txt

```
sh-5.2$ make run
gcc -Wall -Wextra -Werror -c gtuStudentGrades.c -o gtuStudentGrades.o
gcc -Wall -Wextra -Werror -o gtuStudentGrades gtuStudentGrades.o
./gtuStudentGrades
Enter a command, or type 'quit' to exit the program.
gtuStudentGrades grades.txt
```

```
176  Fri Mar 22 14:05:10 2024 Child process finished: 1682, File created: grades.txt
```

2-) addStudentGrade grades.txt Berru Lafci AA

```
addStudentGrade grades.txt Berru Lafci AA
addStudentGrade grades.txt Nana Osaki BB
addStudentGrade grades.txt Lee Chan DC
```

```
177  Fri Mar 22 14:06:18 2024 Child process finished: 1744, Student grade added: Berru Lafci, AA
178  Fri Mar 22 14:12:43 2024 Child process finished: 1859, Student grade added: Nana Osaki, BB
179  Fri Mar 22 14:13:03 2024 Child process finished: 1936, Student grade added: Lee Chan, DC
```

grades.txt

```
1  Berru Lafci, AA
2  Nana Osaki, BB
3  Lee Chan, DC
4
```

Unsuccessful run:

```
addStudentGrade grades.txt Kim Namjoon GG

Usage:
gtuStudentGrades grades.txt: Create file.
addStudentGrade grades.txt studentName studentSurname grade: Add student grade to file. Grade should be between AA and FF.
searchStudent grades.txt studentName studentSurname: Search student grade in file.
showAll grades.txt: Print all of the entries in the file.
listGrades grades.txt: Print 5 entries in the beginning of the file.
listSome numofEntries pageNumber grades.txt: Print numofEntries entries starting from pageNumber in the file.
sortAll grades.txt: Sort student name or grade, in ascending or descending order.
quit: Exit the program.
```

3-) searchStudent grades.txt Berru Lafci

```
searchStudent grades.txt Nana Osaki
Student found! Student name: Nana Osaki, Grade:  BB
searchStudent grades.txt Nur Lafci
```

```
183  Fri Mar 22 14:19:09 2024 Child process finished: 2100, Student grade found: Nana Osaki
184  Fri Mar 22 14:20:22 2024 Child process failed: 2170, Failed to find student grade: Nur Lafci
185
```

4-) sortAll grades.txt

```
sortAll grades.txt
Please enter 1 for name, 0 for grade: 1
Please enter 1 for ascending, 0 for descending: 1
Berru Lafci, AA
Lee Chan, DC
Nana Osaki, BB
```

```
sortAll grades.txt
Please enter 1 for name, 0 for grade: 0
Please enter 1 for ascending, 0 for descending: 1
Berru Lafci, AA
Nana Osaki, BB
Lee Chan, DC
```

```
sortAll grades.txt
Please enter 1 for name, 0 for grade: 1
Please enter 1 for ascending, 0 for descending: 0
Nana Osaki, BB
Lee Chan, DC
Berru Lafci, AA
```

```
sortAll grades.txt
Please enter 1 for name, 0 for grade: 0
Please enter 1 for ascending, 0 for descending: 0
Lee Chan, DC
Nana Osaki, BB
Berru Lafci, AA
```

Unsuccessful run:

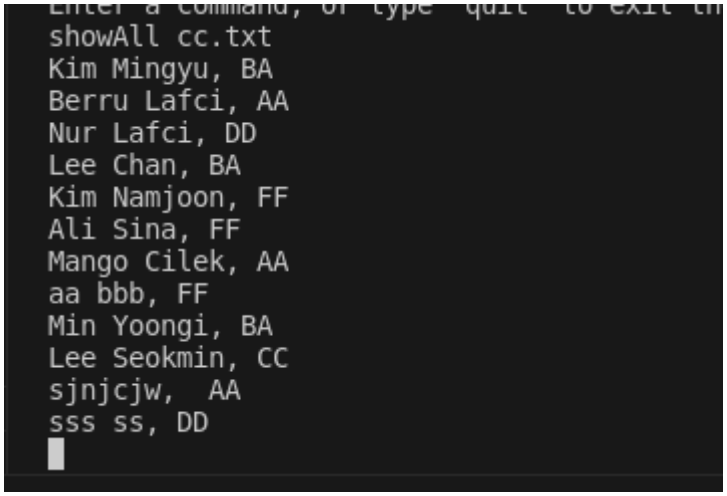
```
sortAll hfbhw.txt
Please enter 1 for name, 0 for grade: 1
Please enter 1 for ascending, 0 for descending: 1
Error opening file: No such file or directory
```

```
Fri Mar 22 14:22:19 2024 Child process finished: 2254, Students sorted: grades.txt
Fri Mar 22 14:23:15 2024 Child process finished: 2324, Students sorted: grades.txt
Fri Mar 22 14:23:49 2024 Child process failed: 2393, Failed to sort students: hfbhw.txt
```

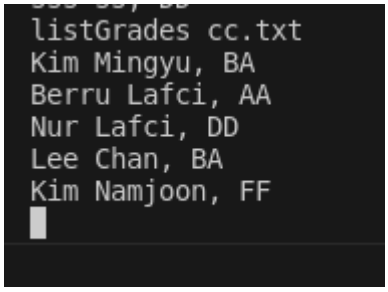
5-) showAll cc.txt

cc.txt:

Command:



6-) listGrades cc.txt



7-) listSome numOfEntries pageNumber cc.txt

```
listSome 3 2 cc.txt
Lee Chan, BA
Kim Namjoon, FF
Ali Sina, FF
```

log.txt:

```
Fri Mar 22 14:41:56 2024 Child process finished: 2550, All student grades shown: cc.txt
Fri Mar 22 14:43:48 2024 Child process finished: 2605, 5 student grades shown: cc.txt
Fri Mar 22 14:45:07 2024 Child process finished: 2683, Some student grades shown: cc.txt
```

8-) gtuStudentGrades

```
gtuStudentGrades

Usage:
gtuStudentGrades grades.txt: Create file.
addStudentGrade grades.txt studentName studentSurname grade: Add student grade to file.
searchStudent grades.txt studentName studentSurname: Search student grade in file.
showAll grades.txt: Print all of the entries in the file.
listGrades grades.txt: Print 5 entries in the beginning of the file.
listSome numofEntries pageNumber grades.txt: Print numofEntries entries starting from pageNumber in the file.
sortAll grades.txt: Sort student name or grade, in ascending or descending order.
quit: Exit the program.
```

```
Fri Mar 22 14:48:11 2024 Child process finished: 2701, Usage shown
```

9-) quit

```
quit
sh-5.2$
```

```
Fri Mar 22 14:48:59 2024 Main process finished: 2522
```