

Alper Tavşanoğlu-210104004142-HW-1-Report

In this program, i am creating a student grade management system in C programming language. The system should allow the user to manage student grades which are stored in a file. In System file operations are to be performed by using the **fork()**. Also system calls such as **open()**, **write()**, and **close()** used. Upon successful completion of a file operation, the child process is terminated. The user should be able to perform the following operations:

1. User should create a file with command **gtuStudentGrades** “name.txt”

```
void createFile(const char* file_name){                                //For Creating File
    pid_t p = fork();
    if (p == -1) {
        perror("fork failed");
        log_record("Fork Failed.(createfile func)", "system.log");
        exit(EXIT_FAILURE);                                          // Failed fork operation
    }
    else if (p == 0){
        int open_file = open(file_name, O_CREAT | O_RDWR | O_TRUNC, 0644); // creat file for student grades
        if (open_file == -1){
            perror("File cannot be created");
            log_record("File could not be opened.(createfile func-No such file or directory)", "system.log"); //save event to log file
            exit(EXIT_FAILURE);
        }
        close(open_file);
        log_record("File created.", "system.log");                  //save event to log file
        printf("File Created Successfully.\n");
        exit(EXIT_SUCCESS);
    }
    else{
        wait(NULL);                                                 //for child process
    }
}
```

I designed the function to create a new file within the file system and designed this file to store student grade information. The **fork()** system call is used to create a new process, which is responsible for the file creation operation. Upon completion of the file creation operation, a log entry is made to record the events.

```
Enter Command -> gtuStudentGrades "grades.txt"
File Created Successfully.

Enter Command ->
```

2. The user should be able to add a new student's grade to the system. The program should prompt the user to enter the student's name and grade, and then add this informaton to the file using a separate process for file manipulation. User should append student and grade to the end of the file with command **addStudentGrade** “Name Surname” “BA” “name.txt” (BA example grade)

```
Enter Command -> addStudentGrade "Alper Tavsanoglu" "BA" "grades.txt"
Student Added/Updated Successfully.
```

The function is to manage students' grades by allowing the user to add a new entry for a student not yet in the file or update the grade of an already listed student. I used the **fork()** system call to create a new process that allows student grades to be added or updated. The function creating a new process using **fork()**.

The child process attempts to open the specified file using the **open()** system call with the **O_RDWR** which allow for reading and writing to the file. It reads the file line by line and checks if the student's name is on any line. If the student is found, their grade is updated by seeking to the appropriate position in the file and writing the new grade. If the student is not found, a new entry is appended to the end of the file with the student's name and grade. The child process exits successfully after completing the task. The parent process waits for the child process to finish before continuing with other commands. A log entry is made to record events while the addition is being performed.

3. The user should be able to search for a student's grade by entering the student's name. The program should then display the student's name and grade if it exists in the file. With command **searchStudent "Name Surname" "name.txt"**

```
Enter Command -> searchStudent "Alper Tavsanoglu" "grades.txt"
Alper Tavsanoglu, BA
Search Completed and Student Found Successfully.
```

The function of the function is to enable users to find a student's grade. I used file operations to read the file and look up the name of the specified student. The **fork()** system call is used to create a new process that performs the search. The function starts by creating a new process using **fork()**. The child process opens the specified file for reading using the **open()** system call. The file is read line by line and each line is checked to see if it contains the student's name. If the student's name is found, the corresponding line (which includes the student's name and grade) is printed to the terminal. If the student's name is not found by the end of the file, a message indicating that the student was not found is printed to the terminal. The child process exits after the search operation is completed. The parent process waits for the child process to finish before continuing with other commands. A log entry is made to record events while the addition is being performed.

4. The user should be able to sort the student grades in the file. The program should provide options to sort by student name or grade, in ascending or descending order. User should print all of the entries sorted by their names with command **sortAll "name.txt"**

Here the sort examples.

```
Enter Command -> sortAll "grades.txt"
Enter 1 to sort by name, 2 to sort by grade: 1
For ascending order enter 1, for descending enter 2: 1
Ali Yas, CC
Alper Tavsanoglu, BA
Ceylan Ol, BB
Kemal Kar, CB
Kerim Ker, FF
Pelin Sari, DD
Selim Sar, DC
Tarkan Sel, AA
Yusuf Geri, BC
Zeynep Bahar, DC
Students Sorted Successfully.
```

```
Enter Command -> sortAll "grades.txt"
Enter 1 to sort by name, 2 to sort by grade: 1
For ascending order enter 1, for descending enter 2: 2
Zeynep Bahar, DC
Yusuf Geri, BC
Tarkan Sel, AA
Selim Sar, DC
Pelin Sari, DD
Kerim Ker, FF
Kemal Kar, CB
Ceylan Ol, BB
Alper Tavsanoglu, BA
Ali Yas, CC
Students Sorted Successfully.
```

```
Enter Command -> sortAll "grades.txt"
Enter 1 to sort by name, 2 to sort by grade: 2
For ascending order enter 1, for descending enter 2: 1
Tarkan Sel, AA
Alper Tavsanoglu, BA
Ceylan Ol, BB
Yusuf Geri, BC
Kemal Kar, CB
Ali Yas, CC
Selim Sar, DC
Zeynep Bahar, DC
Pelin Sari, DD
Kerim Ker, FF
Students Sorted Successfully.
```

```
Enter Command -> sortAll "grades.txt"
Enter 1 to sort by name, 2 to sort by grade: 2
For ascending order enter 1, for descending enter 2: 2
Kerim Ker, FF
Pelin Sari, DD
Zeynep Bahar, DC
Selim Sar, DC
Ali Yas, CC
Kemal Kar, CB
Yusuf Geri, BC
Ceylan Ol, BB
Alper Tavsanoglu, BA
Tarkan Sel, AA
Students Sorted Successfully.
```

The `sortStudentGrades` function allows the user to sort the student grades in a file. It uses the `fork()` system call to create a new process for the sorting operation. The function begins by creating a new process using `fork()`. The child process attempts to open the specified file for reading using the `open()` system call. The file is read, and the entries are stored in an array of `StudentGrade` structures. The `read()` system call is used to read the file contents into a buffer, and `strtok()` is used to tokenize the buffer into lines. Each line is then parsed using `sscanf()` to extract the student's name and grade, which are stored in the array. The user is prompted to enter their choice for sorting by name or grade and to specify the sorting order (ascending or descending). The `qsort()` function is called with the appropriate comparison function based on the user's choices. The comparison functions are used to determine the sorting order. The sorted data is written back to the file using the `open()` system call with the `O_WRONLY | O_TRUNC` flags, which open the file for writing and truncate it to zero length. The `dprintf()` function is used to write the sorted entries back to the file. The sorted entries are also printed to the terminal for the user to view. A log entry is made to record events while the addition is being performed. Here the sort according to the grade in txt.

grades.txt	
1	Tarkan Sel, AA
2	Alper Tavsanoglu, BA
3	Ceylan Ol, BB
4	Yusuf Geri, BC
5	Kemal Kar, CB
6	Ali Yas, CC
7	Selim Sar, DC
8	Zeynep Bahar, DC
9	Pelin Sari, DD
10	Kerim Ker, FF

grades.txt	
1	Kerim Ker, FF
2	Pelin Sari, DD
3	Zeynep Bahar, DC
4	Selim Sar, DC
5	Ali Yas, CC
6	Kemal Kar, CB
7	Yusuf Geri, BC
8	Ceylan Ol, BB
9	Alper Tavsanoglu, BA
10	Tarkan Sel, AA

```

int sort_choice, sort_order;
printf("Enter 1 to sort by name, 2 to sort by grade: ");
scanf("%d", &sort_choice);
printf("For ascending order enter 1, for descending enter 2: ");
scanf("%d", &sort_order);
if (sort_choice == 1){
    if (sort_order == 1){
        qsort(students, count, sizeof(Student), ascen_name);
        log_record("Sorted completed: student grades in ascending order by name.", "system.log");
    }
    else{
        qsort(students, count, sizeof(Student), descen_name);
        log_record("Sorted completed: student grades in descending order by name.", "system.log");
    }
}
else if (sort_choice == 2){
    if (sort_order == 1){
        qsort(students, count, sizeof(Student), grade_ascen);
        log_record("Sorted completed: student grades in ascending order by grade.", "system.log");
    }
    else{
        qsort(students, count, sizeof(Student), grade_descen);
        log_record("Sorted completed: student grades in descending order by grade.", "system.log");
    }
}
else{
    printf("Invalid sort option.\n");
    log_record("Entered sort option is invalid.", "system.log");
    exit(EXIT_FAILURE);
}
open_file = open(filename, O_WRONLY | O_TRUNC);
if (open_file == -1){
    perror("File cannot be opened for writing");
    log_record("File could not be opened.(sort-No such file or directory)", "system.log");
    exit(EXIT_FAILURE);
}

```

5. The user should be able to display all student grades stored in the file. The program should display the student name and grade for each student. With command **showAll** “**name.txt**”

```

Enter Command -> showAll "grades.txt"
All Student Grades
Ali Yas, CC
Alper Tavsanoglu, BA
Ceylan Ol, BB
Kemal Kar, CB
Kerim Ker, FF
Pelin Sari, DD
Selim Sar, DC
Tarkan Sel, AA
Yusuf Geri, BC
Zeynep Bahar, DC
Student Grades Displayed Successfully.

```

Its function is to provide users with the ability to view each student grade entry in the file. The function begins by creating a new process using **fork()**. The child process opens the specified file for reading using the **open()** system call. The file is read into a buffer, and the contents are printed to the terminal. The **read()** system call is used to read the file contents. The buffer contents, which include the student names and grades, are printed to the terminal. The child process exits after the display operation is completed. The parent process waits for the child process to finish using **wait()** before continuing with other commands. A log entry is made to record events while the addition is being performed.

6. The `listGrades` function serves to provide the student grades by displaying the first five entries in the grades file. The user should print first 5 entries with command `listGrades "name.txt"`

```
Enter Command -> listGrades "grades.txt"
First Five Student Grades
Ali Yas, CC
Alper Tavsanoglu, BA
Ceylan Ol, BB
Kemal Kar, CB
Kerim Ker, FF
Displayed Successfully.
```

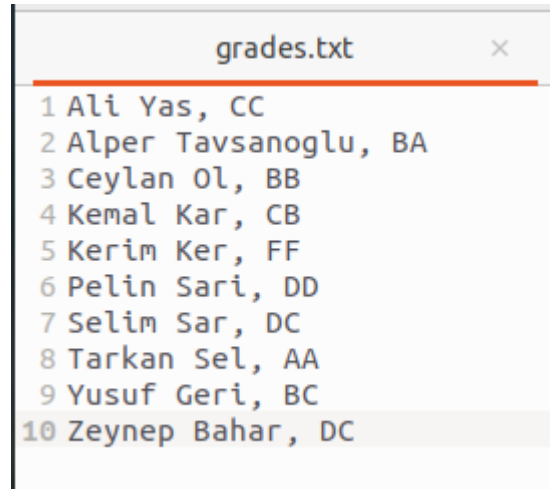
The `fork()` system call is used to create a new process that performs the display operation. The function starts by creating a new process using `fork()`. The child process opens the specified file for reading using the `open()` system call. The file is read into a buffer, and the first five entries are printed to the terminal. The `read()` system call is used to read the file contents, and `strtok()` is used to tokenize the buffer into lines. A counter keeps track of the number of lines printed. The buffer contents, which include the student names and grades, are printed to the terminal until five entries have been displayed or the end of the file is reached. The child process exits after the display operation is completed. The parent process waits for the child process to finish using `wait()` before continuing with other commands. A log entry is made to record events while the addition is being performed.

```
void listGrades(const char* file_name){
    //for display first 5 student
    pid_t p = fork();
    if (p == -1){
        perror("fork failed");
        log_record("Fork failed.(listGrades)", "system.log");
        exit(EXIT_FAILURE);
    }
    else if (p == 0){
        int open_file = open(file_name, O_RDONLY);
        if (open_file == -1){
            perror("File cannot be opened");
            log_record("File could not be opened.(listGrades-No such file or directory)", "system.log");
            exit(EXIT_FAILURE);
        }
        printf("First Five Student Grades\n");
        char str[MAX_BUF];
        ssize_t read_byt;
        int flag = 0;
        while ((read_byt = read(open_file, str, MAX_BUF - 1)) > 0 && flag < 5){
            str[read_byt] = '\0';
            char* line = strtok(str, "\n");
            while (line != NULL && flag < 5){
                printf("%s\n", line);
                line = strtok(NULL, "\n");
                flag++;
            }
            close(open_file);
            log_record("Displayed the first five student grades.", "system.log");
            printf("Displayed Successfully.\n");
            exit(EXIT_SUCCESS);
        }
        else{
            wait(NULL);
        }
    }
}
```

7. The `listSome` function allows users to view a paginated section of the student grades file. For example, if the user wants to see entries 6 through 10, they can request to list 5 entries starting from page 2. With command **`listSome "numofEntries" "pageNumber" "name.txt"`**

```
Enter Command -> listSome 5 2 "grades.txt"

Students
Pelın Sari, DD
Selim Sar, DC
Tarkan Sel, AA
Yusuf Geri, BC
Zeynep Bahar, DC
```



```
grades.txt
1 Ali Yas, CC
2 Alper Tavsanoglu, BA
3 Ceylan Ol, BB
4 Kemal Kar, CB
5 Kerim Ker, FF
6 Pelin Sari, DD
7 Selim Sar, DC
8 Tarkan Sel, AA
9 Yusuf Geri, BC
10 Zeynep Bahar, DC
```

The function starts by creating a new process using **`fork()`**. The child process opens the specified file for reading using the **`open()`** system call. The function calculates the range of entries to display based on the `numofEntries` and `pageNumber` provided by the user. It determines the starting and ending indices for the entries to be displayed. The file is read into a buffer, and the function iterates through the entries. It prints the entries that fall within the specified range to the terminal. The buffer contents, which include the student names and grades, are printed to the terminal for the range of entries specified by the user or the end of the file is reached. The child process exits after the display operation is completed. The parent process waits for the child process to finish using **`wait()`** before continuing with other commands. A log entry is made to record events while the addition is being performed.

8. The user should be able to display all of the available commands by calling `gtuStudentGrades` without an argument. Command **`gtuStudentGrades`** without arguments should list all of the instructions or informaton provided about how commands in program.

```
Enter Command -> gtuStudentGrades

Commands List

gtuStudentGrades "filename"
addStudentGrade "Name Surname" "Grade" "filename"
searchStudent "Name Surname" "filename"
sortAll "filename"
showAll "filename"
listGrades "filename"
listSome numOfEntries pageNumber "filename"
gtuStudentGrades
exit

- Create a file to store student grades. (e.g. gtuStudentGrades "grades.txt")
- Add or update a student's grade. (e.g. addStudentGrade "Alper Tavsanoglu" "AA" "grades.txt")
- Search for a student's grade. (e.g. searchStudent "Alper Tavsanoglu" "grades.txt")
- Sort the student grades by name or grade in ascending or descending order. (e.g. sortAll "graderest.txt")
- Display all student grades stored in the file. (e.g. showAll "graderest.txt")
- Display the first 5 student grades from the file. (e.g. listGrades "graderest.txt")
- List a specific range of student grades. (e.g. listSome 5 2 "graderest.txt")
- List available commands. (e.g. gtuStudentGrades)
- Exit the program. (e.g. exit)
```


9. The System should create a log file that records the completion of each task as desired. The writeToLog function's primary purpose is to create a persistent record of the operations performed by the system. Here the steps how i implement writeToLog function; First open the log file for writing, Obtain the current time and date, Format the log entry string, Write the formatted log entry to the log file. Here the some examples of log file informations.

```
1 [2024-03-22 19:07:09] Sorted completed: student grades in ascending order by name.
2 [2024-03-22 19:07:09] Sorted students writed to the file.
3 [2024-03-22 19:08:27] Sorted completed: student grades in descending order by name.
4 [2024-03-22 19:08:27] Sorted students writed to the file.
5 [2024-03-22 19:10:01] Sorted completed: student grades in ascending order by grade.
6 [2024-03-22 19:10:01] Sorted students writed to the file.
7 [2024-03-22 19:10:56] Sorted completed: student grades in descending order by grade.
8 [2024-03-22 19:10:56] Sorted students writed to the file.
9 [2024-03-22 20:04:40] Sorted completed: student grades in ascending order by grade.
10 [2024-03-22 20:04:40] Sorted students writed to the file.
11 [2024-03-22 20:04:43] Program Finished.
12 [2024-03-22 20:09:04] Sorted completed: student grades in ascending order by name.
13 [2024-03-22 20:09:04] Sorted students writed to the file.
14 [2024-03-22 20:09:15] Sorted completed: student grades in descending order by name.
15 [2024-03-22 20:09:15] Sorted students writed to the file.
16 [2024-03-22 20:09:30] Sorted completed: student grades in ascending order by grade.
17 [2024-03-22 20:09:30] Sorted students writed to the file.
18 [2024-03-22 20:09:40] Sorted completed: student grades in descending order by grade.
19 [2024-03-22 20:09:40] Sorted students writed to the file.
20 [2024-03-22 20:09:44] Program Finished.
21 [2024-03-22 20:34:54] Sorted completed: student grades in ascending order by name.
22 [2024-03-22 20:34:54] Sorted students writed to the file.
23 [2024-03-22 20:37:48] Displayed all student grades.
24 [2024-03-22 21:06:28] Displayed the first five student grades.
25 [2024-03-22 21:14:42] Displayed student grades from entry 6 to 10.
26 [2024-03-22 21:23:42] Usage Printed.
```

10. And the last thing is makefile. Its running the program with **make** command. It also have **make clean** command which is remove everything except .c and makefile.

Some Test Examples

```
alper@alper-VirtualBox: ~/Desktop$ make
-----
Removing compiled files...
-----
Compiling...
-----
Running the tests...
=====
./grade_manage_system

Welcome to Student Grade Management System
Enter gtuStudentGrades command for available commands

Enter Command -> gtuStudentGrades

Commands List

gtuStudentGrades "filename"          - Create a file to store student grades. (e.g. gtuStudentGrades "grades.txt")
addStudentGrade "Name Surname" "Grade" "filename" - Add or update a student's grade. (e.g. addStudentGrade "Alper Tavsanoğlu" "AA" "grades.txt")
searchStudent "Name Surname" "filename" - Search for a student's grade. (e.g. searchStudent "Alper Tavsanoğlu" "grades.txt")
sortAll "filename"                   - Sort the student grades by name or grade in ascending or descending order. (e.g. sortAll "gradedest.txt")
showAll "filename"                   - Display all student grades stored in the file. (e.g. showAll "gradedest.txt")
listGrades "filename"                 - Display the first 5 student grades from the file. (e.g. listGrades "gradedest.txt")
listSome numOfEntries pageNumber "filename" - List a specific range of student grades. (e.g. listSome 5 2 "gradedest.txt")
gtuStudentGrades                     - List available commands. (e.g. gtuStudentGrades)
exit                                 - Exit the program. (e.g. exit)

Enter Command -> gtuStudentGrades "grades.txt"
File Created Successfully.

Enter Command -> addStudentGrade "Alper Tavsanoğlu" "AA" "grades.txt"
Student Added/Updated Successfully.

Enter Command -> searchStudent "Alper Tavsanoğlu" "grades.txt"
Alper Tavsanoğlu, AA
Search Completed and Student Found Successfully.

Enter Command -> addStudentGrade "Kemal Kar" "BB" "grades.txt"
Student Added/Updated Successfully.

Enter Command -> addStudentGrade "Pelin Sarı" "CC" "grades.txt"
Student Added/Updated Successfully.

Enter Command -> addStudentGrade "Tarkan Sel" "DD" "grades.txt"
Student Added/Updated Successfully.
```

```
Enter Command -> addStudentGrade "Zeynep Bahar" "BA" "grades.txt"
Student Added/Updated Successfully.

Enter Command -> addStudentGrade "Yusuf Geri" "CB" "grades.txt"
Student Added/Updated Successfully.

Enter Command -> showAll "grades.txt"
All Student Grades
Alper Tavsanoğlu, AA
Kemal Kar, BB
Pelin Sarı, CC
Tarkan Sel, DD
Zeynep Bahar, BA
Yusuf Geri, CB
Student Grades Displayed Successfully.

Enter Command -> sortAll "grades.txt"
Enter 1 to sort by name, 2 to sort by grade: 1
For ascending order enter 1, for descending enter 2: 1
Alper Tavsanoğlu, AA
Kemal Kar, BB
Pelin Sarı, CC
Tarkan Sel, DD
Yusuf Geri, CB
Zeynep Bahar, BA
Students Sorted Successfully.

Enter Command -> sortAll "grades.txt"
Enter 1 to sort by name, 2 to sort by grade: 1
For ascending order enter 1, for descending enter 2: 2
Zeynep Bahar, BA
Yusuf Geri, CB
Tarkan Sel, DD
Pelin Sarı, CC
Kemal Kar, BB
Alper Tavsanoğlu, AA
Students Sorted Successfully.
```

```
Enter Command -> sortAll "grades.txt"
Enter 1 to sort by name, 2 to sort by grade: 2
For ascending order enter 1, for descending enter 2: 1
Alper Tavsanoğlu, AA
Zeynep Bahar, BA
Kemal Kar, BB
Yusuf Geri, CB
Pelin Sari, CC
Tarkan Sel, DD
Students Sorted Successfully.

Enter Command -> sortAll "grades.txt"
Enter 1 to sort by name, 2 to sort by grade: 2
For ascending order enter 1, for descending enter 2: 2
Tarkan Sel, DD
Pelin Sari, CC
Yusuf Geri, CB
Kemal Kar, BB
Zeynep Bahar, BA
Alper Tavsanoğlu, AA
Students Sorted Successfully.

Enter Command -> showAll "grades.txt"
All Student Grades
Tarkan Sel, DD
Pelin Sari, CC
Yusuf Geri, CB
Kemal Kar, BB
Zeynep Bahar, BA
Alper Tavsanoğlu, AA
Student Grades Displayed Successfully.

Enter Command -> listGrades "grades.txt"
First Five Student Grades
Tarkan Sel, DD
Pelin Sari, CC
Yusuf Geri, CB
Kemal Kar, BB
Zeynep Bahar, BA
Displayed Successfully.

Enter Command -> listSome 2 2 "grades.txt"

Students
Yusuf Geri, CB
Kemal Kar, BB

Enter Command -> exit
Exiting program.
=====
Completed tests....
alper@alper-VirtualBox:~/Desktop$
```

system.log		×
1	[2024-03-22 21:46:32]	Usage Printed.
2	[2024-03-22 21:46:59]	File created.
3	[2024-03-22 21:47:21]	Added/Updated student grade completed.
4	[2024-03-22 21:47:46]	Search completed: student found.
5	[2024-03-22 21:48:56]	Added/Updated student grade completed.
6	[2024-03-22 21:49:21]	Added/Updated student grade completed.
7	[2024-03-22 21:49:56]	Added/Updated student grade completed.
8	[2024-03-22 21:50:20]	Added/Updated student grade completed.
9	[2024-03-22 21:50:50]	Added/Updated student grade completed.
10	[2024-03-22 21:51:10]	Displayed all student grades.
11	[2024-03-22 21:51:24]	Sorted completed: student grades in ascending order by name.
12	[2024-03-22 21:51:24]	Sorted students writed to the file.
13	[2024-03-22 21:51:34]	Sorted completed: student grades in descending order by name.
14	[2024-03-22 21:51:34]	Sorted students writed to the file.
15	[2024-03-22 21:51:45]	Sorted completed: student grades in ascending order by grade.
16	[2024-03-22 21:51:45]	Sorted students writed to the file.
17	[2024-03-22 21:51:58]	Sorted completed: student grades in descending order by grade.
18	[2024-03-22 21:51:58]	Sorted students writed to the file.
19	[2024-03-22 21:52:11]	Displayed all student grades.
20	[2024-03-22 21:52:28]	Displayed the first five student grades.
21	[2024-03-22 21:52:41]	Displayed student grades from entry 3 to 4.
22	[2024-03-22 21:58:12]	Program Finished.