

CSE 344 HW 1 PROJECT REPORT

STUDENT MANAGEMENT SYSTEM

Buket GENCER 210104004298

Introduction

This program creates a platform for managing student grades, (developed in the C programming language) allowing users to execute various operations such as adding grades, searching for specific student grades, sorting them, and displaying them in an organized manner.

The program utilizes **fork()** for concurrent grade management, boosting performance. It employs system calls such as **open()** for file access, **read()** and **write()** to manipulate data, and **close()** to conclude operations, optimizing tasks like grade addition and search through efficient file management.

The features include creating a new file for grades, appending student grades, retrieving and displaying grade information, sorting grades by name or value, and listing grades in various formats. Through these functionalities, the program aims to offer a comprehensive solution for efficient grade management.

Project Description

Main functionalities of the Project:

- **Creating a File:** Initializes a new file or opens an existing one to store student grades.

```
//function to write to file
void writeToFile(char *fileName, char *content){
    int fileDescriptor = open(fileName, O_WRONLY | O_APPEND | O_CREAT, 0644); //open the file or create a new one
    if(fileDescriptor == -1){ //if the file does not exist, give error
        perror("Error opening file"); //print the error message to terminal
        exit(1); //exit the program
    }
    write(fileDescriptor, content, strlen(content)); //write the content to the file
    close(fileDescriptor); //close the file
}
```

This helper function is used in commands.

- **Adding Student Grades:** Allows the addition of a new student's name and grade to the file.

```
//addStudentGrade function. the arguments are name, grade, and file name.  
void addStudentGrade(char *name, char *grade, char *fileName){
```

- **Searching for Grades:** Enables searching for a specific student's grade by their name.

```
//searchStudent function. the arguments are name and file name.  
void searchStudent(char *name, char *fileName){
```

- **Sorting Grades:** Provides functionality to sort the stored grades by student names or grade values, in either ascending or descending order. By default, students are sorted by name in ascending order.

```
//sortAll function. the arguments are file name, option1, and option2.  
void sortAll(char *fileName, char *option1, char *option2) {
```

- **Displaying Grades:** Provides displaying of the entire file content.

```
//showAll function. the arguments are file name.  
void showAll(char *fileName){
```

- **Displaying Grades:** Displays the first five entries from the list, showing the initial set of student grades.

```
//listGrades function to print to terminal the first 5 entries(lines) in the file  
void listGrades(char *fileName)
```

Provides options to view grades in a specified range, enabling targeted examination of the data.

```
//listSome function to print to terminal the entries between the given page number and number of entries  
void listSome(int numOfEntries, int pageNumber, char *fileName){
```

- **Makefile Usage:** Simplifies the compilation and execution process with predefined commands for building the project, cleaning up object files, and managing dependencies.

```
all : clear compile run clean

clear:
    @clear

clean:
    @rm -f *.o
    @rm -f *.out

compile:
    @gcc -c stuManSis.c
    @gcc -o stuManSis.out stuManSis.o

run:
    @./stuManSis.out
```

- **Usage:** Program displays a list of all available commands that the user can execute.

```
write(1, "Usage:\n", 7);
write(1, "addStudentGrade \"Name Surname\" \"Grade\" \"fileName.txt\"\n", 55);
write(1, "searchStudent \"Name Surname\" \"fileName.txt\"\n", 45);
write(1, "sortAll \"fileName.txt\"\n", 24);
write(1, "sortAll \"Name Surname\" \"ascending or descending\" \"fileName.txt\"\n", 65);
write(1, "sortAll \"Grade\" \"ascending or descending\" \"fileName.txt\"\n", 58);
write(1, "showAll \"fileName.txt\"\n", 24);
write(1, "listGrades \"fileName.txt\"\n", 27);
write(1, "listSome numofEntries pageNumber \"fileName.txt\"\n", 48);
write(1, "q\n", 2);
```

- **Logging Operations:** Records each operation performed within the system to a log file for future reference and auditing.

```
//function to write to log file
void writeLog(char *command, char *message){
```

```
searchStudent Student grade searched successfully
searchStudent Student can not found in the file
searchStudent Student grade searched successfully
addStudentGrade Student grade added successfully
addStudentGrade Student grade added successfully
```

```
listGrades First 5 student grades displayed successfully
listSome Some student grades displayed successfully
listSome Some student grades displayed successfully
addStudentGrade Invalid grade format
```

- **Concurrent Processing with fork():** Enhances the system's multitasking abilities by allowing multiple student grade management operations to be processed simultaneously, improving performance and response times.

Implementation Details

In the Student Grade Management System, core tasks are managed through system calls. The **fork()** call is crucial for multitasking, creating child processes for tasks like adding or searching for grades. The **open()**, **read()**, **write()**, and **close()** calls facilitate file access, data retrieval, updating, and secure closure after operations. Child processes handle specific file operations, improving the system's efficiency and response. Error handling was a focus to ensure stability and reliability.

Testing and Results

- **GtuStudentGrades**

```
Enter a command: gtuStudentGrades
Usage:
addStudentGrade "Name Surname" "Grade" "fileName.txt"
searchStudent "Name Surname" "fileName.txt"
sortAll "fileName.txt"
sortAll "Name Surname" "ascending or descending" "fileName.txt"
sortAll "Grade" "ascending or descending" "fileName.txt"
showAll "fileName.txt"
listGrades "fileName.txt"
listSome numofEntries pageNumber "fileName.txt"
q
Enter a command: █
```

- **gtuStudentGrades with an argument**


```
Enter a command: gtuStudentGrades "grades.txt"
Enter a command: gtuStudentGrades "newGrades.txt"
```

newGrades.txt file created. Log file message :

```
gtuStudentGrades File created successfully
```

- **addStudent Grade**

```
Enter a command: addStudentGrade "Ayse Can" "BB" "grades.txt"
```

 grades.txt

```
1 Matthew Williams, BB
2 Patricia Robinson, AA
3 Daniel Moore, VF
4 Anthony Clark, CB
5 Anthony Lewis, VF
6 Matthew Lee, FF
7 Anthony Gonzalez, NA
8 Betty Anderson, NA
9 Christopher Anderson, BA
10 Mary Perez, FF
11 Barbara Hernandez, VF
12 Robert Davis, BA
13 Matthew Brown, DD
14 Sarah Ramirez, BB
15 James Gonzalez, BB
16 Matthew Moore, CB
17 Ayse Can, BB
18
```

```
Enter a command: addStudentGrade "Ayse Can" "BB" "grades.txt"
Enter a command: addStudentGrade "Ayse Can" "BBB" "grades.txt"
Invalid grade format
Enter a command: addStudentGrade "Ayse Can" "BB" "nonexistfile.txt"
Enter a command: AddStudentGrade "Ayse Can"
Invalid command
Enter a command: AddStudentGrade
Invalid command
Enter a command: 
```

Nonexistfile.txt created and student information adding this new file.

Log file:

```
addStudentGrade Student grade added successfully
addStudentGrade Invalid grade format
addStudentGrade Student grade added successfully
addStudentGrade Student grade added successfully
Invalid command Invalid command
Invalid command Invalid command
```

- **searchStudent :**

```
Enter a command: searchStudent "Ayse Can" "grades.txt"
Ayse Can, BB
Enter a command: searchStudent "No One" "grades.txt"
Student cannot be found in the file
Enter a command: █
```

- **sortAll:**

```
Enter a command: sortAll "grades.txt"
```

```
Anthony Clark, CB  
Anthony Gonzalez, NA  
Anthony Lewis, VF  
Ayse Can, BB  
Barbara Hernandez, VF  
Betty Anderson, NA  
Christopher Anderson, BA  
Daniel Moore, VF  
James Gonzalez, BB  
Mary Perez, FF  
Matthew Brown, DD  
Matthew Lee, FF  
Matthew Moore, CB  
Matthew Williams, BB  
Patricia Robinson, AA  
Robert Davis, BA  
Sarah Ramirez, BB
```

```
Enter a command: sortAll "grade" "ascending" "grades.txt"
```

```
Patricia Robinson, AA  
Christopher Anderson, BA  
Robert Davis, BA  
Ayse Can, BB  
Matthew Williams, BB  
Sarah Ramirez, BB  
James Gonzalez, BB  
Anthony Clark, CB  
Matthew Moore, CB  
Matthew Brown, DD  
Matthew Lee, FF  
Mary Perez, FF  
Anthony Gonzalez, NA  
Betty Anderson, NA  
Daniel Moore, VF  
Anthony Lewis, VF  
Barbara Hernandez, VF
```

```
Enter a command: sortAll "grade" "descending" "grades.txt"
Daniel Moore, VF
Anthony Lewis, VF
Barbara Hernandez, VF
Anthony Gonzalez, NA
Betty Anderson, NA
Matthew Lee, FF
Mary Perez, FF
Matthew Brown, DD
Anthony Clark, CB
Matthew Moore, CB
Matthew Williams, BB
Sarah Ramirez, BB
James Gonzalez, BB
Ayse Can, BB
Christopher Anderson, BA
Robert Davis, BA
Patricia Robinson, AA
Enter a command: 
```

- **showAll:**


```
Patricia Robinson, AA
Enter a command: showAll "grades.txt"
Matthew Williams, BB
Patricia Robinson, AA
Daniel Moore, VF
Anthony Clark, CB
Anthony Lewis, VF
Matthew Lee, FF
Anthony Gonzalez, NA
Betty Anderson, NA
Christopher Anderson, BA
Mary Perez, FF
Barbara Hernandez, VF
Robert Davis, BA
Matthew Brown, DD
Sarah Ramirez, BB
James Gonzalez, BB
Matthew Moore, CB
Ayse Can, BB
Enter a command: 
```

- **listGrades :**

```
Enter a command: listGrades "grades.txt"
Matthew Williams, BB
Patricia Robinson, AA
Daniel Moore, VF
Anthony Clark, CB
Anthony Lewis, VF
```

- **listSome :**

```
Enter a command: listSome 3 3 "grades.txt"
Anthony Gonzalez, NA
Betty Anderson, NA
Christopher Anderson, BA
Enter a command: 
```

```
Enter a command: listSome 20 20 "grades.txt"
The desired range is not in the file
```

- command “q” quit the program.

Log File Example

```
100 listGrades First 5 student grades displayed successfully
101 listSome Some student grades displayed successfully
102 listSome Student grades displayed successfully
103 searchStudent Student grade searched successfully
104 searchStudent Student grade searched successfully
105 Invalid command Invalid command
106 sortAll Student grades sorted successfully
107 sortAll Student grades sorted successfully
108 gtuStudentGrades Usage displayed
109 addStudentGrade Student grade added successfully
110 searchStudent Student grade searched successfully
111 sortAll Student grades sorted successfully
112 showAll All student grades displayed successfully
113 listGrades First 5 student grades displayed successfully
114 listGrades First 5 student grades displayed successfully
115 listSome Some student grades displayed successfully
116 listSome Student grades displayed successfully
117 listGrades First 5 student grades displayed successfully
118 ✨ listGrades First 5 student grades displayed successfully
119 |
```

Memory Check via Valgrind

After use all command valgrind result:

```

Enter a command: q
==64588== Memcheck, a memory error detector
==64588== Copyright (C) 2002-2017, and GNU GPL'd, by Julian Seward et al.
==64588== Using Valgrind-3.18.1 and LibVEX; rerun with -h for copyright info
==64588== Command: ./stuManSis.out
==64588==
q
Enter a command: ==64588==
==64588== HEAP SUMMARY:
==64588==      in use at exit: 0 bytes in 0 blocks
==64588==    total heap usage: 5 allocs, 5 frees, 5,120 bytes allocated
==64588==
==64588== All heap blocks were freed -- no leaks are possible
==64588==
==64588== For lists of detected and suppressed errors, rerun with: -s
==64588== ERROR SUMMARY: 0 errors from 0 contexts (suppressed: 0 from 0)

```

Makefile Valgrind:

```

clean:
    @rm -f *.o
    @rm -f *.out

compile:
    @gcc -g -c stuManSis.c
    @gcc -g -o stuManSis.out stuManSis.o

run:
    ✨ @./stuManSis.out

valgrind:
    @valgrind --leak-check=full --show-leak-kinds=all --track-origins=yes ./stuManSis.out

```

Conclusion

This project has been a comprehensive journey through the intricacies of developing a Student Grade Management System utilizing the C programming language. The integration of fundamental system programming concepts such as process creation with **fork()**, inter-process communication, and file manipulation has not only solidified my understanding of these areas but also highlighted the importance of efficient resource management and error handling in software development.

Through the implementation of various functionalities such as adding, searching, sorting, and displaying student grades, I have encountered and overcome several challenges. These include ensuring data consistency during concurrent file access, managing memory leaks, and optimizing the performance of the system. The use of tools like Valgrind for memory leak detection and debugging has been instrumental in enhancing the robustness of the application.

In conclusion, working on this project taught me several key aspects of system programming, such as process creation with `fork()`, file manipulation, and incorporating external tools like `valgrind` for memory leak detection. Learning to use these tools and understanding the principles behind managing processes and memory were crucial takeaways. This experience has not only expanded my technical skills but also emphasized the importance of problem-solving and continuous improvement in programming.