Fatih Doğaç
CSE344 HW#1 REPORT

1901042654

My program starts with SIGINT handling sigaction.

```
int main()
{
    struct sigaction sa;
    memset(&sa, 0, sizeof(sa));
    sa.sa_handler = &sigint_handler;
    sigaction(SIGINT, &sa, NULL);
```

To handle the SIGINT, I kept all the child proccesses id's in a global pid_t array and I kept the child number too.

```
pid_t childs[100];
int child_num = 0;
```

And here is the sigint_handler();

```
void sigint_handler() {
    printf("\nSIGINT received. Killing all child processes...\n");
    cleanChilds();
    exit(0);
}

void cleanChilds(){
    for (int i = 0; i < child_num; i++) {
        kill(childs[i], SIGKILL);
    }
}
```

It notifies the user with printf that the SIGINT received and kills all child proccesses than exits.

Then the menu() runs.

Menu function is basically a never ending while loop.

Beginning of the while loop, the program checks if the input is a one word and checks if there is ' " ' in the input. And if it is a one word, the program only supports two option: gtuStudentGrades and exit. Other than these two is error.

```
Type gtuStudentGrades to see the options. Type exit to exit..:
^C
SIGINT received. Killing all child processes...
koso@koso-ABRA-A5-V17-2:~/Desktop/24/sys/hw1$
```

```c
int i = 0;
int flag = 0; // There is no " in the input.
while(i < strln(option)){
    if(option[i] == '\"'){
        flag = 1; // There is " in the input.
    }
    i++;
}

if(flag == 0){
    if (strcmpp(option, "gtuStudentGrades") == 0)
    {
        printf("\ngtuStudentGrades \"file_name\"\n");
        printf("addStudentGrade \"file_name\" \"Name Surname\" \"grade\"\n");
        printf("searchStudent \"file_name\" \"Name Surname\"\n");
        printf("sortAll \"file_name\" \"0/1\" (0 = Sort Grades / 1 = Sort Names) \"0/1\" (0 = Descending / 1= Ascending)\n");
        printf("showAll \"file_name\"\n");
        printf("listGrades \"file_name\"\n");
        printf("listSome \"number of entries\" \"page number\" \"file_name\"\n\n");
    }
    else if(strcmpp(option, "exit") == 0){
        printf("\nBye!\n");
        cleanChilds();
        exit(0);
    }
    else{
        perror("Bad input");
        continue;
    }
```

```c
else{

    for (int i = 0; i < 10; i++) // Reset the inputs
    {
        my_memset(inputs[i], 0, sizeof(inputs[i]));
    }

    index = 0;
    for (int i = 0; i < strln(option); i++)
    {
        if(i == 0){
            while(option[i+1] != '\"'){
                inputs[index][i] = option[i];
                i++;
            }
        }

        if(option[i] == '\"'){
            int k = 0;
            index++;

            while(option[i+1] != '\"'){
                inputs[index][k] = option[i+1];
                i++;
                k++;
                if (i >= strln(option)) break;
            }
            i++;
            inputs[index][k] = '\0';
        }
    }
```

And if it is not a one word input, the program goes into this else.
Memset is for cleaning the inputs array. It prevents the current input to not overlay with old inputs.

Program takes the whole input into option string. And it has a limit of 100 chars.

And then the program checks the option string char by char and if it encounters a quotation mark, it runs until it finds another quotation mark. Stores the word between quotation marks in inputs array.

```
if (strcmpp(inputs[0], "gtuStudentGrades") == 0)
{
    if(index > 1){
        perror("Too many inputs");
        cleanChilds();
        exit(0);
    }

    createFile(inputs[1]);
}
else if(strcmpp(inputs[0], "addStudentGrade") == 0){
    if(index > 3){
        perror("Too many inputs");
        cleanChilds();
        exit(0);
    }
    addStudent(inputs[1], inputs[2], inputs[3]);
}
else if(
    if(i
        void cleanChilds()
        void logIt(char *str, int flag);
        cleanChilds();
        exit(0);
    }
    searchStudent(inputs[1], inputs[2]);
}
else if(strcmpp(inputs[0], "sortAll") == 0){
    if(index > 3){
        perror("Too many inputs");
        cleanChilds();
        exit(0);
    }
    sortAll(inputs[1], atoi(inputs[2]) , atoi(inputs[3]));
}
```

```
else if(strcmpp(inputs[0], "showAll") == 0){
    if(index > 1){
        perror("Too many inputs");
        cleanChilds();
        exit(0);
    }
    showAll(inputs[1]);
}
else if(strcmpp(inputs[0], "listGrades") == 0){
    if(index > 1){
        perror("Too many inputs");
        cleanChilds();
        exit(0);
    }
    listGrades(inputs[1]);
}
else{ // List Some
    if(index > 3){
        perror("Too many inputs");
        cleanChilds();
        exit(0);
    }
    listSome(inputs[1], inputs[2], inputs[3]);
}
```

And all the setup is ready. The only matter left is to see what the user wanted to do.

For some reason my strcmpp() didn't work with "listSome". I know it is a bad practice but I tried to understand why it works with anything else and not with listSome. Then some time passed and I just gave up and put that command in the else.

As you can see above, every command has its own function and every one of them checks if the inputs are at the correct number. If not cleans the childs and exits.

```
Type gtuStudentGrades to see the options. Type exit to exit..:
gtuStudentGrades

gtuStudentGrades "file_name"
addStudentGrade "file_name" "Name Surname" "grade"
searchStudent "file_name" "Name Surname"
sortAll "file_name" "0/1" (0 = Sort Grades / 1 = Sort Names) "0/1" (0 = Descending / 1= Ascending)
showAll "file_name"
listGrades "file_name"
listSome "number of entries" "page number" "file_name"


Type gtuStudentGrades to see the options. Type exit to exit..:
```

1. gtuStudentGrades “grades.txt” command

The program creates a child with fork() and stores the pid of the child and increases the child number. And then it creates the desired file with the proper permissions then lefts. And parent waits for child to finish.

```
void createFile(char file_name[100]){

    childs[child_num] = fork();

    if(childs[child_num++] == 0){
        int fd = open(file_name, O_CREAT , 0666);
        close(fd);
        printf("\nFile %s created.",file_name);
        exit(0);
    }
    else{
        wait(NULL);
    }

}
```

```
koso@koso-ABRA-A5-V17-2:~/Desktop/24/sys/hw1$ ./gradeManager
Welcome...

Type gtuStudentGrades to see the options. Type exit to exit..:
gtuStudentGrades "grades.txt"

File grades.txt created.
Type gtuStudentGrades to see the options. Type exit to exit..:
```

2. addStudentGrade “Name Surname” “AA” command

```c
void addStudent(char* file_name, char* name, char* grade){

    childs[child_num++] = fork();

    if(childs[child_num - 1] == 0){
        int fd = open(file_name, O_WRONLY | O_APPEND);

        if (fd < 0)
        {
            perror("Error openning file");
            exit(1);
        }

        write(fd, name, strln(name));
        write(fd, ",",1);
        write(fd, grade, strln(grade));
        write(fd, "\n",1);

        printf("\nStudent added to the file.\n\n");

        close(fd);
        exit(0);
    }
    else{
        wait(NULL);
    }
}
```

It creates a child and then checks if the file opened successfully. If yes it stores the data in Name Surname,Grade order. And informs the user. Closes the file descriptor and exits. And its parent waits for the child.

```
Type gtuStudentGrades to see the options. Type exit to exit..:
addStudentGrade "grades.txt" "Fatih Dogac" "AA"

Student added to the file.


Type gtuStudentGrades to see the options. Type exit to exit..:
```

```
nal  Help
C sys.c  3    grades.txt ×    M makefile
grades.txt
1   Fatih Dogac,AA
2
```

3.searchStudent “Name Surname” command

```c
void searchStudent(char* file_name, char* name){

    childs[child_num++] = fork();
    char c;
    int bytes_read, k=0;
    char buffer[1024];

    if(childs[child_num - 1] == 0){
        int fd = open(file_name, O_RDONLY);

        while ((bytes_read = read(fd, &c , 1)) != 0)
        {
            if(c != '\n' && c != EOF){
                buffer[k++] = c;
            }
            else{
                buffer[k] = '\0';

                char* token = strtokk(buffer, ',');
                if(strcmpp(token, name) == 0){
                    token = strtokk(NULL, ',');
                    printf("\n%-25s%s %s\n\n",name , " |", token);
                }

                k = 0;
            }

        }
            if (k > 0) { // Last line
                buffer[k] = '\0';

                char* token = strtokk(buffer, ',');

                if(strcmpp(token, name) == 0){
                    token = strtokk(NULL, ',');
                    printf("%s - %s\n", name, token);
                }
            }

            close(fd);
            exit(0);
        }
        else{
            wait(NULL);
        }
    }
```

This function reads until it encounters an end of line character or end of file, then it slices the input and compares the name in the current line with the wanted name. If it matches then prints the name and the grade.

The current file:

```
grades.txt
1  Fatih Dogac,AA
2  Fa Do,BB
3  Fatih Dogac,CB
4  Fatihh Dogac,CC
5  Fatihhh Dogac,DD
6  Fatihhhh Dogac,FF
```

```
Type gtuStudentGrades to see the options. Type exit to exit..:
searchStudent "grades.txt" "Fatih Dogac"

Fatih Dogac                | AA


Type gtuStudentGrades to see the options. Type exit to exit..:
```

4.sortAll “gradest.txt” “0” “1” command

My sortAll function takes sort order and what to sort as input from user as 0 and 1.

0 stands for Sort Names and 1 for Sort Grades
and the next input 0 stands for Descending order and 1 for Ascending order.

I used bubbleSort for this sorting and put a if in it to sort in ascending or descending order.

```
Type gtuStudentGrades to see the options. Type exit to exit..:
sortAll "grades.txt" "1" "0"


_____________________________________________
Fatihhhh Dogac                            | FF |
Fatihhh Dogac                             | DD |
Fatihh Dogac                              | CC |
Fatihhhhhh Dogac                          | CB |
Fa Do                                     | BB |
Fatih Dogac                               | AA |


Type gtuStudentGrades to see the options. Type exit to exit..:
sortAll "grades.txt" "0" "0"


_____________________________________________
Fatihhhhhh Dogac                          | CB |
Fatihhhh Dogac                            | FF |
Fatihhh Dogac                             | DD |
Fatihh Dogac                              | CC |
Fatih Dogac                               | AA |
Fa Do                                     | BB |


Type gtuStudentGrades to see the options. Type exit to exit..:
```

5.Display Commands

I used a printFile() function to avoid code repeatation. I couldn't do it for other commands because they needed other handling situations.

showAll "grades.txt"

```
Type gtuStudentGrades to see the options. Type exit to exit..:
showAll "grades.txt"


_____________________________________________
Fatih Dogac                               | AA |
Fa Do                                     | BB |
Fatihhhhhh Dogac                          | CB |
Fatihh Dogac                              | CC |
Fatihhh Dogac                             | DD |
Fatihhhh Dogac                            | FF |


Type gtuStudentGrades to see the options. Type exit to exit..:
```

listGrades “grades.txt”

```
Type gtuStudentGrades to see the options. Type exit to exit..:
listGrades "grades.txt"


_____________________________________
Fatih Dogac                          | AA |
Fa Do                                | BB |
Fatihhhhhh Dogac                     | CB |
Fatihh Dogac                         | CC |
Fatihhh Dogac                        | DD |


Type gtuStudentGrades to see the options. Type exit to exit..:
```

listSome “numofEntries” “pageNumber” “grades.txt”

```
Type gtuStudentGrades to see the options. Type exit to exit..:
listSome "5" "1" "grades.txt"


_____________________________________
Fatih Dogac                          | AA |
Fa Do                                | BB |
Fatihhhhhh Dogac                     | CB |
Fatihh Dogac                         | CC |
Fatihhh Dogac                        | DD |


Type gtuStudentGrades to see the options. Type exit to exit..:
listSome "5" "2" "grades.txt"


_____________________________________
Fatihhhh Dogac                       | FF |


Type gtuStudentGrades to see the options. Type exit to exit..:
```

I checked program with gcc -Wall -Wextra and it is okay. Valgrind doesn’t show any leaks. And I tested the program with large files such as 5mb.
I don’t have a logging operation.

This is the makefile:

```
makefile
CC = gcc
CFLAGS = -Wall -Wextra

run: sys.c string.c
	$(CC) $(CFLAGS) -o gradeManager sys.c string.c

clean:
	rm -f *.txt *.o a.out gradeManager *.log
```