

CSE344 SYSTEMS PROGRAMMING
HOMEWORK 1
REPORT

Burak Demirkaya

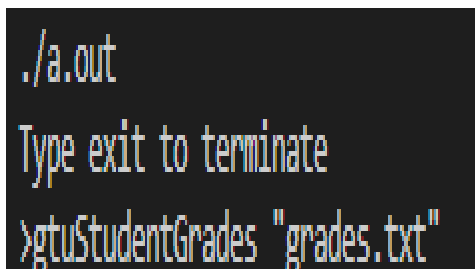
210104004274

21/03/2024

In order to run the program I used makefile to clean, clear, compile and run it. In makefile clean, I deleted every txt file and the executable files that has been created. Since I do not use dynamic allocation, resetting the txt files is my approach. I used an infinite loop inside main to take command inputs from the user. The loop will end if the user types exit. For each process I created a child process in each of the commands function. I have implemented the logic of the commands inside the child block and exit with failure or success depending on the situation. For each file operation, I used fork system call. In order some of the commands to work, rather than the command itself the arguments need to be in between “ ” marks. Sometimes the exit command needs to be written more than one to terminate the program.

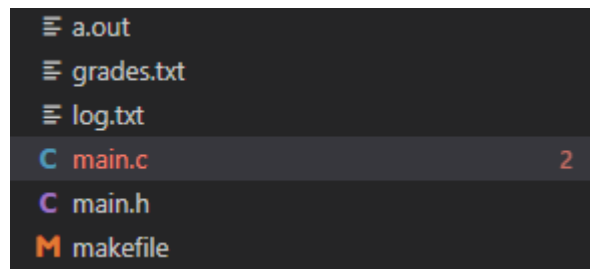
Input Formats and Test of the Commands

1) gtuStudentGrades “filename”



```
./a.out
Type exit to terminate
>gtuStudentGrades "grades.txt"
```

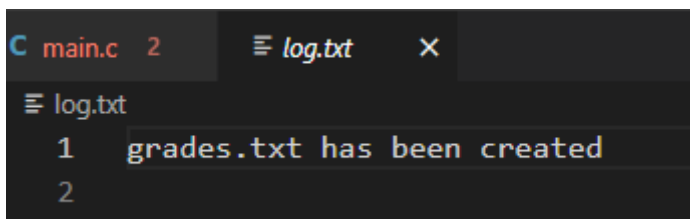
Input format



```
a.out
grades.txt
log.txt
C main.c 2
C main.h
M makefile
```

Files created after successful operation

The log.txt file will store the logging information of each process. The file's entry will be like this as below after this process.



```
log.txt
1 grades.txt has been created
2
```

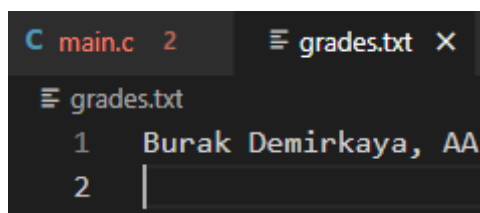
2) addStudentGrade "Name Surname" "Grade" "filename"

```
>addStudentGrade "Burak Demirkaya" "AA" "grades.txt"
```

This command will add the student to the given filename. If the file does not exist, it will print an error message and terminate the child process.

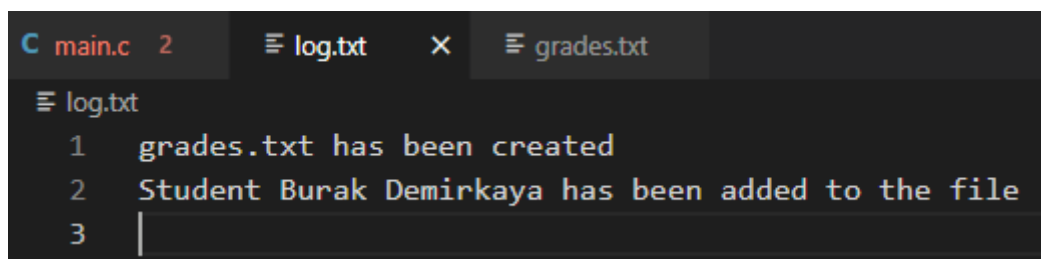
```
char buffer[256];
int byte_written = sprintf(buffer,"%s, %s\n",studentNameSurname,grade);
if(write(fd,buffer,byte_written) == -1){
    perror("write");
    logCommands("Write failed at adding student grade process");
    close(fd);
    exit(EXIT_FAILURE);
}
```

In order to format the given student name and grade, I used "sprintf" function.



```
C main.c 2  grades.txt X
grades.txt
1 Burak Demirkaya, AA
2 |
```

Figure 2.1



```
C main.c 2  log.txt X  grades.txt
log.txt
1 grades.txt has been created
2 Student Burak Demirkaya has been added to the file
3 |
```

Figure 2.2

Figure 2.1-> grades.txt file after successful operation

Figure 2.2-> log.txt file after successful operation

3) searchStudent "Name Surname" "filename"

```
char buffer[256];
char lineBuffer[512]; // Assuming a line won't be longer than 512 characters
ssize_t bytesRead;
int linePos = 0;
int found = 0;

while ((bytesRead = read(fd, buffer, sizeof(buffer) - 1)) > 0 && !found) {
    for (int i = 0; i < bytesRead; ++i) {
        if (buffer[i] == '\n' || linePos == sizeof(lineBuffer) - 1) {
            lineBuffer[linePos] = '\0'; // Null-terminate the line

            // Attempt to find a comma which separates nameSurname and grade
            char* commaPos = strchr(lineBuffer, ',');
            if (commaPos) {
                *commaPos = '\0'; // Temporarily terminate the nameSurname part for comparison
                if (strcmp(lineBuffer, studentNameSurname) == 0) {
                    printf("Student found: %s, %s\n", studentNameSurname, commaPos + 2); // Assuming a space follows the comma, hence +2
                    found = 1;
                    break;
                }
                *commaPos = ','; // Restore the comma in case we need to print or log the line
            }
            linePos = 0;
        } else {
            lineBuffer[linePos++] = buffer[i];
        }
    }
}
```

Tokenize the entries in the file to check whether any of the name surname in the entries has a full match with the input.

```
>searchStudent "Burak Demirkaya" "grades.txt"
Student found: Burak Demirkaya, AA
```

Figure 3.1

```
>searchStudent "Burak" "grades.txt"
Student Burak not found
```

Figure 3.2

Figure 3.1-> If the given student name and surname is found in the file

Figure 3.2-> If the given student name and surname does not match any of the entries in the file

```
C main.c 2  log.txt  grades.txt
log.txt
1  grades.txt has been created
2  Student Burak Demirkaya has been added to the file
3  You have searched for Burak in the file grades.txt.
4  You have searched for Burak Demirkaya in the file grades.txt.
```

log.txt will look like this after searching process.

4) sortAll "filename"

```
typedef struct{
    char nameSurname[100];
    char grade[3];
} StudentEntries;

int compareByNameDes(const void* a, const void* b) {
    const StudentEntries* entryA = (const StudentEntries*)a;
    const StudentEntries* entryB = (const StudentEntries*)b;
    return strcmp(entryA->nameSurname, entryB->nameSurname);
}

int compareByGradeDes(const void* a, const void* b) {
    const StudentEntries* entryA = (const StudentEntries*)a;
    const StudentEntries* entryB = (const StudentEntries*)b;
    return strcmp(entryA->grade, entryB->grade);
}

int compareByNameAsc(const void* a, const void* b) {
    const StudentEntries* entryA = (const StudentEntries*)a;
    const StudentEntries* entryB = (const StudentEntries*)b;
    return strcmp(entryB->nameSurname, entryA->nameSurname);
}

int compareByGradeAsc(const void* a, const void* b) {
    const StudentEntries* entryA = (const StudentEntries*)a;
    const StudentEntries* entryB = (const StudentEntries*)b;
    return strcmp(entryB->grade, entryA->grade);
}
```

```
int (*compareFunc)(const void*, const void*) = NULL;
if (strcmp(type, "name") == 0 && strcmp(order, "ascending") == 0) {
    compareFunc = compareByNameAsc;
}
else if (strcmp(type, "grade") == 0 && strcmp(order, "ascending") == 0) {
    compareFunc = compareByGradeAsc;
}
else if (strcmp(type, "name") == 0 && strcmp(order, "descending") == 0) {
    compareFunc = compareByNameDes;
}
else if (strcmp(type, "grade") == 0 && strcmp(order, "descending") == 0) {
    compareFunc = compareByGradeDes;
}

if (compareFunc) {
    qsort(entries, numOfEntries, sizeof(StudentEntries), compareFunc);
}
```

I have declared a struct and defined the comparison functions in order to use it at "qsort" function.

```

char buffer[1024];
ssize_t bytesRead;
StudentEntries entries[100];
int numOfEntries = 0;

while((bytesRead = read(fd, buffer, sizeof(buffer) - 1)) > 0){
    buffer[bytesRead] = '\0'; // Null-terminate the buffer

    char* line = strtok(buffer, "\n");
    while (line != NULL && numOfEntries < 100) {
        char* commaPos = strchr(line, ',');
        if (commaPos && *(commaPos + 1) == ' ') { // Check for a space after the comma
            *commaPos = '\0'; // Split the string at the comma
            strncpy(entries[numOfEntries].nameSurname, line, sizeof(entries[numOfEntries].nameSurname) - 1);
            entries[numOfEntries].nameSurname[sizeof(entries[numOfEntries].nameSurname) - 1] = '\0';
            strncpy(entries[numOfEntries].grade, commaPos + 2, sizeof(entries[numOfEntries].grade) - 1); // Skip comma and space
            entries[numOfEntries].grade[sizeof(entries[numOfEntries].grade) - 1] = '\0';
            numOfEntries++;
        }
        line = strtok(NULL, "\n");
    }
}

if(bytesRead == -1){
    perror("read");
    logCommands("File read failed at sortAll process");
    close(fd);
    exit(EXIT_FAILURE);
}

```

I have stored every entry in the file into a struct object. Maximum number of entries is 100 since I did not use dynamic allocation.

```

C main.c 2 log.txt grades.txt X
grades.txt
1 Burak Demirkaya, AA
2 Plk ska, BB
3 Asdf Ghj, BA
4 Qwe rty, CC
5 Cde fvr, FF

```

grades.txt file before

```

>sortAll "grades.txt"
Sort by name or grade:name
Ascending or descending:ascending
Asdf Ghj, BA
Burak Demirkaya, AA
Cde fvr, FF
Plk ska, BB
Qwe rty, CC

```

After sorting by name and ascending order

```

>sortAll "grades.txt"
Sort by name or grade:name
Ascending or descending:descending
Qwe rty, CC
Plk ska, BB
Cde fvr, FF
Burak Demirkaya, AA
Asdf Ghj, BA

```

After sorting by name and descending order

Order depends on the ASCII values of the characters.

```
>sortAll "grades.txt"
Sort by name or grade:grade
Ascending or descending:ascending
Cde fvr, FF
Qwe rty, CC
Plk ska, BB
Asdf Ghj, BA
Burak Demirkaya, AA
```

After sorting by grade and ascending order

```
>sortAll "grades.txt"
Sort by name or grade:grade
Ascending or descending:descending
Burak Demirkaya, AA
Asdf Ghj, BA
Plk ska, BB
Qwe rty, CC
Cde fvr, FF
```

After sorting by grade and descending order

Order depending on the grade system which means AA is the highest whereas FF is the lowest.

```
You have printed the entries in sorted way
You have printed the entries in sorted way
You have printed the entries in sorted way
You have printed the entries in sorted way
```

log.txt will look like this after sorting.

5) showAll "filename"

```
>showAll "grades.txt"
Burak Demirkaya, AA
Plk ska, BB
Asdf Ghj, BA
Qwe rty, CC
Cde fvr, FF
```

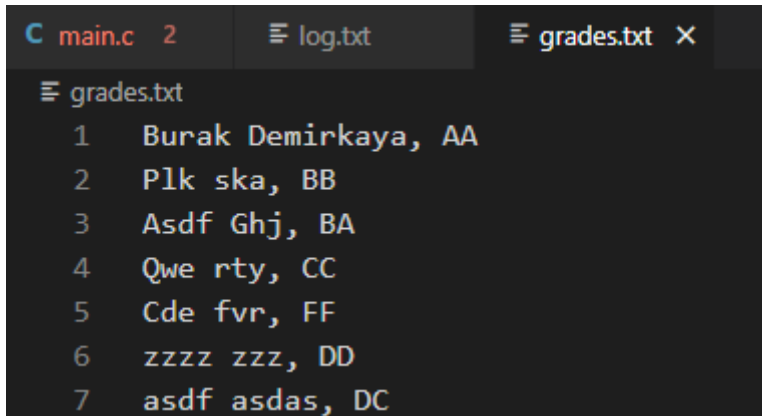
This process simply reads every entry into the buffer and prints them.

```
char buffer[1024];
ssize_t bytes_read;
while((bytes_read = read(fd,buffer,sizeof(buffer))) > 0){
    if(write(STDOUT_FILENO,buffer,bytes_read) != bytes_read){
        perror("write");
        close(fd);
        exit(EXIT_FAILURE);
    }
}
```

Buffer size is fixed and it is 1024.

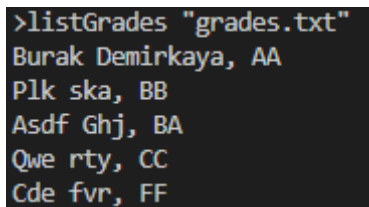
You printed every entry in the file log.txt after printing every entry.

6) listGrades "filename"

A screenshot of a code editor with three tabs: 'main.c 2', 'log.txt', and 'grades.txt'. The 'grades.txt' tab is active, showing a list of 7 entries, each with a line number from 1 to 7. The entries are: 1 Burak Demirkaya, AA; 2 Plk ska, BB; 3 Asdf Ghj, BA; 4 Qwe rty, CC; 5 Cde fvr, FF; 6 zzzz zzz, DD; 7 asdf asdas, DC.

```
C main.c 2 log.txt grades.txt X
grades.txt
1 Burak Demirkaya, AA
2 Plk ska, BB
3 Asdf Ghj, BA
4 Qwe rty, CC
5 Cde fvr, FF
6 zzzz zzz, DD
7 asdf asdas, DC
```

grades.txt file before

A screenshot of a terminal window showing the command '>listGrades "grades.txt"' and its output, which lists the first 5 entries of the grades.txt file: Burak Demirkaya, AA; Plk ska, BB; Asdf Ghj, BA; Qwe rty, CC; Cde fvr, FF.

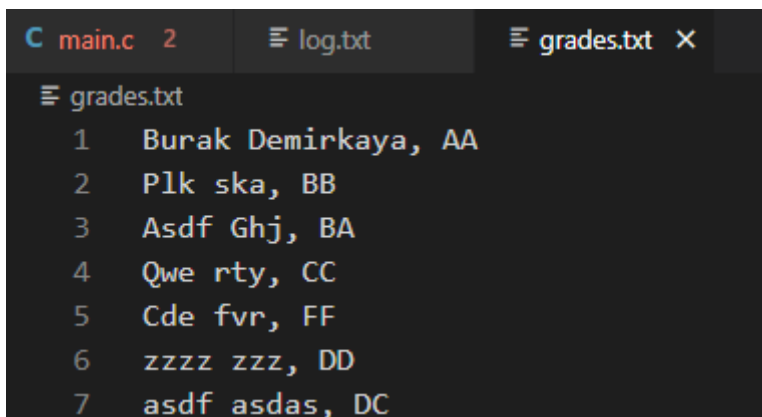
```
>listGrades "grades.txt"
Burak Demirkaya, AA
Plk ska, BB
Asdf Ghj, BA
Qwe rty, CC
Cde fvr, FF
```

After running the command

You printed the first 5 entries in the file log.txt file after listing first 5 entries.

7) listSome numberOfEntries pageNumber "filename"

I considered numberOfEntries input as the number of entries in each page and the page number is the page that we are looking for.

A screenshot of a code editor with three tabs: 'main.c 2', 'log.txt', and 'grades.txt'. The 'grades.txt' tab is active, showing a list of 7 entries, each with a line number from 1 to 7. The entries are: 1 Burak Demirkaya, AA; 2 Plk ska, BB; 3 Asdf Ghj, BA; 4 Qwe rty, CC; 5 Cde fvr, FF; 6 zzzz zzz, DD; 7 asdf asdas, DC.

```
C main.c 2 log.txt grades.txt X
grades.txt
1 Burak Demirkaya, AA
2 Plk ska, BB
3 Asdf Ghj, BA
4 Qwe rty, CC
5 Cde fvr, FF
6 zzzz zzz, DD
7 asdf asdas, DC
```

grades.txt before


```
>listSome 3 2 "grades.txt"
Qwe rty, CC
Cde fvr, FF
zzzz zzz, DD
```

Considered as each page has 3 entries and it will print the second page which will be entries from number 4-7. Number 7 is not included.

```
You have printed some of the entries between 4 and 7
```

 log for log.txt file

8) gtuStudentGrades

This command works without an argument and it will print the list of the commands and information about them.

```
>gtuStudentGrades
Listing all commands...
gtuStudentsGrades "filename" ---> Creates a file
addStudentGrade "Name Surname" "AA" "filename" ---> Appends student and grade at the end of the file
searchStudent "Name Surname" "filename" ---> Prints student name, surname and grade if found in the file
sortAll "filename" ---> Prints all entries sorted by their names
showAll "filename" ---> Prints all of the entries in the file
listGrades "filename" ---> Prints first 5 entries
listSome "numOfEntries" "pageNumber" "filename" ---> Lists some of the entries
gtuStudentGrades ---> Lists all of the instructions
```

Since this process does not involve any file operation I did not use fork system call for this one.

9) Logging

I have logged the actions based on the results, even there occurs some error while opening a file or writing to it, I printed it to log.txt file. I have also printed the log actions of the successful operations.