



OPERATING SYSTEMS

LAB 1: SYSTEM CALLS

Mazin Bersy – 100558650
Malek Mahmoud – 100558649
Laila Sayed - 100558440

Contents

| | |
|---|----|
| Description of the code | 2 |
| Create file code description | 2 |
| Combine file code description..... | 2 |
| Test Cases | 3 |
| Testing crear..... | 3 |
| Testing combine..... | 4 |
| Problems found and their solutions | 11 |
| Conclusion | 12 |

Description of the code

Create file code description

First, the main function has two parameters passed in it: the number of arguments passed and an array of characters pointers to store the passed argument as a string. Then, the code consists of four main parts:

1. The first part is the if statement to identify the arguments of the program: file name and mode. The if condition is to identify that the program only accepts three arguments and if more arguments are passed it will give an exit statement (-1).
2. The second part is to assign the first passed argument for the file name and the second argument for the file mode.

The second if statement is to convert the string mode to octal integer and assign an error statement that is shown if the conversion failed.

The "unmask" statement is to set the mask to zero so all the file permissions are controlled, and the original unmask is stored in "mask".

3. The third main part is for the file creation after making sure the file doesn't exist. If it already exists the program prints an error message on the screen.
4. The last main part is for setting file permissions and reassure it was built correctly by using "chmod" command. If the command failed the program prints an errors message and exits.

Combine file code description

The code first defines a struct to store students' information: name of student, student's grade and a variable for extra student's extra information like exam calls.

- The first main function is the "compare students" function to compare the grades of two different students. The grades are passed as arguments to the function then the function returns -1, 1 or 0 if student a grade is higher than student b, student's b grade is higher, or two students have the same grade respectively.
- The second main function is the "main function". The function checks that the program receives only three arguments: two input file names and one output file name, if not it prints the correct usage and exists.

Then the code start opening the first file and executes an error message if there was an error while opening the file. If the file is opened successfully it starts to extract 100 students' names from the file and puts them in the "students" array.

The code repeats the same thing for the second file and merge the read data with the “students” array.

The code then calls the “qsort” function to sort the “students” array by grade using the “compare students” function that was created before.

Then an output file is created to store the result of “qsort” function. There is also a for loop for writing all the data and ensures that all students records are written.

Consequently, a CSV file is used for recording students’ grades statistics and prints an error message if the file isn’t created. Then another loop is used to calculate how many students there are in each grade category.

Finally, the program calculates the percentage of students in each grade category and writes these data in the CSV file and prints a message for ensuring that the two files have been merged successfully.

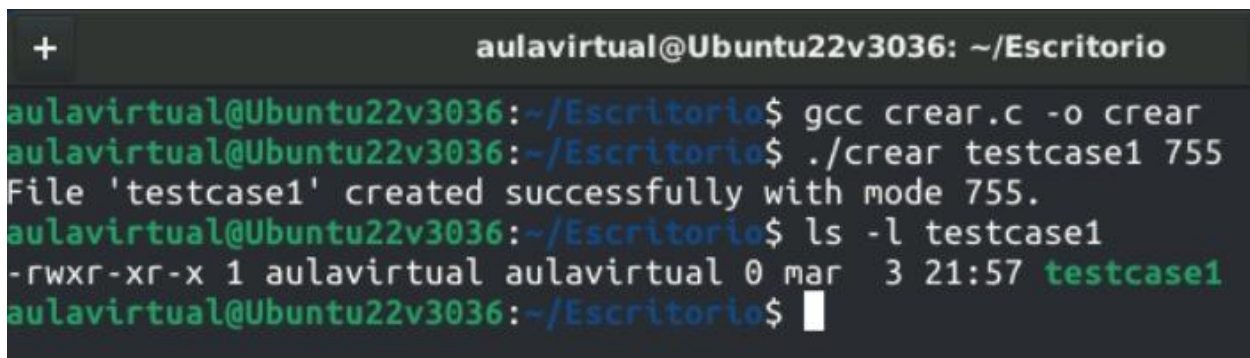
Test Cases

Testing crear

Test case 1:

In the first test case we aim to generally test the permission of the output file, whether it matches the user input. We run the ls-l command in order to check this.

Expected output → `rw-r--r--=755`



```

+ aulavirtual@Ubuntu22v3036: ~/Escritorio
aulavirtual@Ubuntu22v3036:~/Escritorio$ gcc crear.c -o crear
aulavirtual@Ubuntu22v3036:~/Escritorio$ ./crear testcase1 755
File 'testcase1' created successfully with mode 755.
aulavirtual@Ubuntu22v3036:~/Escritorio$ ls -l testcase1
-rwxr-xr-x 1 aulavirtual aulavirtual 0 mar  3 21:57 testcase1
aulavirtual@Ubuntu22v3036:~/Escritorio$

```

Test case 2:

For the second test case, the goal is to check the output when a nonvalid argument is given.

Expected Output → Error message and return -1 without creating the file.

```

+ aulavirtual@Ubuntu22v3036: ~/Escritorio
aulavirtual@Ubuntu22v3036:~/Escritorio$ ./crear testcase2 905
Error: Invalid mode format: 905

```

Test case 3:

This test case covers the possibility of no input arguments and wrong number of arguments

Expected Output → Error message and return -1 without creating the file.

```

+ aulavirtual@Ubuntu22v3036: ~/Escritorio
aulavirtual@Ubuntu22v3036:~/Escritorio$ ./crear testcase3
Usage: ./crear <filename> <mode>
aulavirtual@Ubuntu22v3036:~/Escritorio$ ./crear
Usage: ./crear <filename> <mode>
aulavirtual@Ubuntu22v3036:~/Escritorio$ 

```

Test case 4:

Lastly, this test case tries to create a file that already exists.

Expected Output → Error message and return -1 without creating the file.

```

+ aulavirtual@Ubuntu22v3036: ~/Escritorio
aulavirtual@Ubuntu22v3036:~/Escritorio$ ./crear testcase2 644
Error creating file
: File exists

```

Testing combine

Test case 1:

This test case aims to deal with a capacity of students higher than a 100.

Input file1 →

```

struct alumnos lista[] = {
    {"Alejandro Morales", 6, 1},
    {"Beatriz Ortega", 8, 2},

```

{"Carlos Dominguez", 7, 3},
{"Diana Ramirez", 9, 1},
{"Eduardo Santos", 5, 2},
{"Fernanda Jimenez", 10, 3},
{"Gabriel Herrera", 4, 1},
{"Hilda Vasquez", 7, 2},
{"Ismael Rojas", 8, 3},
{"Juana Flores", 6, 1},
{"Kevin Castro", 9, 2},
{"Laura Mendez", 5, 3},
{"Mario Nunez", 10, 1},
{"Nadia Ponce", 7, 2},
{"Oscar Velazquez", 6, 3},
{"Paula Guzman", 8, 1},
{"Quintin Salazar", 9, 2},
{"Rosa Delgado", 7, 3},
{"Santiago Rios", 5, 1},
{"Tania Espinoza", 10, 2},
{"Ulises Vega", 6, 3},
{"Valeria Cordero", 8, 1},
{"Walter Marquez", 9, 2},
{"Ximena Solis", 7, 3},
{"Yahir Bravo", 4, 1},
{"Zulema Fuentes", 10, 2},
{"Antonio Suarez", 6, 3},
{"Barbara Castillo", 7, 1},
{"Cesar Mejia", 9, 2},
{"Dolores Navarro", 5, 3},
{"Esteban Duarte", 10, 1},
{"Felicia Roldan", 8, 2},
{"Gonzalo Ibanez", 7, 3},
{"Helena Montoya", 6, 1},
{"Ivan Estrada", 9, 2},
{"Jacqueline Gutierrez", 8, 3},
{"Karla Campos", 7, 1},
{"Luis Figueroa", 10, 2},
{"Marta Rosales", 5, 3},
{"Nicolas Dominguez", 6, 1},
{"Olga Ortega", 8, 2},
{"Pedro Jimenez", 7, 3},

```

{"Ramona Vasquez", 9, 1},
{"Saul Herrera", 5, 2},
{"Tamara Rojas", 10, 3},
{"Ursula Velazquez", 6, 1},
{"Victor Ponce", 8, 2},
{"Wendy Guzman", 7, 3},
{"Xavier Salazar", 9, 1},
{"Yolanda Delgado", 5, 2},
{"Zacarias Rios", 10, 3}
};

```

Input file2 →

```

struct alumnos lista[] = {
    {"Alejandro Morales", 6, 1},
    {"Beatriz Ortega", 8, 2},
    {"Carlos Dominguez", 7, 3},
    {"Diana Ramirez", 9, 1},
    {"Eduardo Santos", 5, 2},
    {"Fernanda Jimenez", 10, 3},
    {"Gabriel Herrera", 4, 1},
    {"Hilda Vasquez", 7, 2},
    {"Ismael Rojas", 8, 3},
    {"Juana Flores", 6, 1},
    {"Kevin Castro", 9, 2},
    {"Laura Mendez", 5, 3},
    {"Mario Nunez", 10, 1},
    {"Nadia Ponce", 7, 2},
    {"Oscar Velazquez", 6, 3},
    {"Paula Guzman", 8, 1},
    {"Quintin Salazar", 9, 2},
    {"Rosa Delgado", 7, 3},
    {"Santiago Rios", 5, 1},
    {"Tania Espinoza", 10, 2},
    {"Ulises Vega", 6, 3},
    {"Valeria Cordero", 8, 1},
    {"Walter Marquez", 9, 2},
    {"Ximena Solis", 7, 3},
    {"Yahir Bravo", 4, 1},
    {"Zulema Fuentes", 10, 2},

```

```
{"Antonio Suarez", 6, 3},  
{"Barbara Castillo", 7, 1},  
{"Cesar Mejia", 9, 2},  
{"Dolores Navarro", 5, 3},  
{"Esteban Duarte", 10, 1},  
{"Felicia Roldan", 8, 2},  
{"Gonzalo Ibanez", 7, 3},  
{"Helena Montoya", 6, 1},  
{"Ivan Estrada", 9, 2},  
{"Jacqueline Gutierrez", 8, 3},  
{"Karla Campos", 7, 1},  
{"Luis Figueroa", 10, 2},  
{"Marta Rosales", 5, 3},  
{"Nicolas Dominguez", 6, 1},  
{"Olga Ortega", 8, 2},  
{"Pedro Jimenez", 7, 3},  
{"Ramona Vasquez", 9, 1},  
{"Saul Herrera", 5, 2},  
{"Tamara Rojas", 10, 3},  
{"Ursula Velazquez", 6, 1},  
{"Victor Ponce", 8, 2},  
{"Wendy Guzman", 7, 3},  
{"Xavier Salazar", 9, 1},  
{"Yolanda Delgado", 5, 2},  
{"Zacarias Rios", 10, 3}  
};
```

Expected Output → Error message and return -1 without creating the file.


```

MINGW64:/c/Users/mazin/OneDrive/Desktop/OS/Lab Assignment 1
mazin@LAPTOP-JEPHKCS9 MINGW64 ~/OneDrive/Desktop/OS/Lab Assignment 1
$ gcc file_creator.c -o file_creator

mazin@LAPTOP-JEPHKCS9 MINGW64 ~/OneDrive/Desktop/OS/Lab Assignment 1
$ ./file_creator file1
Datos guardados en 'file1' correctamente.

mazin@LAPTOP-JEPHKCS9 MINGW64 ~/OneDrive/Desktop/OS/Lab Assignment 1
$ ./file_creator file2
Datos guardados en 'file2' correctamente.

mazin@LAPTOP-JEPHKCS9 MINGW64 ~/OneDrive/Desktop/OS/Lab Assignment 1
$ gcc combine.c -o combine

mazin@LAPTOP-JEPHKCS9 MINGW64 ~/OneDrive/Desktop/OS/Lab Assignment 1
$ ./combine file1 file2 output1
Error: Maximum student limit exceeded.

mazin@LAPTOP-JEPHKCS9 MINGW64 ~/OneDrive/Desktop/OS/Lab Assignment 1
$ |

```

Test case 2:

This test case operates successfully as correct input is provided within capacity.

Input f1 →

```

struct alumnos lista[] = {
    {"Juan Perez", 5, 1},
    {"Maria Lopez", 9, 2},
    {"Carlos Garcia", 7, 1},
    {"Ana Fernandez", 4, 3}
};

```

Input f2 →

```

struct alumnos lista[] = {
    {"Pedro Perez", 8, 1},
    {"Carlos Lopez", 9, 2},
    {"Alfonos Garcia", 7, 1},
    {"Maribel Fernandez", 5, 3}
};

```

Expected Output → Files successfully created

```

mazin@LAPTOP-JEPHKCS9 MINGW64 ~/OneDrive/Desktop/OS/Lab Assignment 1
$ gcc combine.c -o combine

mazin@LAPTOP-JEPHKCS9 MINGW64 ~/OneDrive/Desktop/OS/Lab Assignment 1
$ ./combine f1 f2 output5
Successfully merged and sorted students into 'output5'.
Statistics saved in 'estadisticas.csv'.

mazin@LAPTOP-JEPHKCS9 MINGW64 ~/OneDrive/Desktop/OS/Lab Assignment 1
$ |

```

```

M;0;0.00%
S;2;25.00%
N;3;37.50%
A;2;25.00%
F;1;12.50%

```

Test case 3:

For this test case, its goal is to check how the code handles wrong values given for students such as more than 10 or less than 0.

Input f1 →

```

struct alumnos lista[] = {
    {"Juan Perez", 5, 1},
    {"Maria Lopez", 9, 2},
    {"Carlos Garcia", 7, 1},
    {"Ana Fernandez", 4, 3}
};

```

Input file7 →

```

struct alumnos lista[] = {
    {"Pedro Perez", 11, 1},
    {"Carlos Lopez", -1, 2},
};

```

```

{"Alfonos Garcia", 7, 1},
{"Maribel Fernandez", 5, 3}
};

```

Expected Output → Disregards the invalid grades when doing calculations.

```

mazin@LAPTOP-JEPHKCS9 MINGW64 ~/OneDrive/Desktop/OS/Lab Assignment 1
$ gcc combine.c -o combine

mazin@LAPTOP-JEPHKCS9 MINGW64 ~/OneDrive/Desktop/OS/Lab Assignment 1
$ ./combine f1 f2 output5
Successfully merged and sorted students into 'output5'.
Statistics saved in 'estadisticas.csv'.

mazin@LAPTOP-JEPHKCS9 MINGW64 ~/OneDrive/Desktop/OS/Lab Assignment 1
$ gcc file_creator.c -o file_creator

mazin@LAPTOP-JEPHKCS9 MINGW64 ~/OneDrive/Desktop/OS/Lab Assignment 1
$ ./file_creator file7
Datos guardados en 'file7' correctamente.

mazin@LAPTOP-JEPHKCS9 MINGW64 ~/OneDrive/Desktop/OS/Lab Assignment 1
$ ./combine f1 file7 output5
Successfully merged and sorted students into 'output5'.
Statistics saved in 'estadisticas.csv'.

mazin@LAPTOP-JEPHKCS9 MINGW64 ~/OneDrive/Desktop/OS/Lab Assignment 1
$

```

```

M;0;0.00%
S;1;16.67%
N;2;33.33%
A;2;33.33%
F;1;16.67%

```

Problems found and their solutions

1. Problem:

While constructing the code one of the problems that appeared was trying to use the open () function when the file already exists.

Solution:

Uses the O_EXCL flag, which is exclusive, so when trying to use it while the file already exists it gives an error message.

2. Problem:

Binary Representation of Doubles:

- Double numbers are stored in memory in binary format.
- To write them to a text-based CSV file, they need to be converted into **string** with proper formatting because we cannot use write () function directly it would not work correctly since write () operates on **bytes**, not formatted text.

Solution:

We added snprintf() function to the program. snprintf() function formats a string into a **buffer** while ensuring it does not exceed the buffer size.

3. Problem:

The umask setting may modify the permissions unintentionally

Solution:

Setting the unmask value to be equal zero so it will not change any permissions.

4. Problem:

In the note there might be some students' invalid values that are greater than 10 or less than zero and these values might give misleading statistics percentages and values.

Solution:

For this student invalid value, it takes the value "F" and is excluded from the statistics calculations. Consequently, It will not affect the statistics final percentage value.

Conclusion

The first C program is designed to create a new file with user-specified permissions while ensuring it is not accidentally overwritten. It validates input, converts permission modes correctly, and applies them using `open()` and `chmod()`. The second C program efficiently merges student records from two binary input files, sorts them based on grades (nota), and writes the sorted data to an output file. Additionally, it generates a CSV file (`estadisticas.csv`) with statistical summaries of student performance. The use of system calls such as `open()`, `read()`, and `write()` makes the program lightweight and efficient for handling structured binary data. Throughout the programs there are multiple safety nets set to detect errors early on, such as an invalid number of arguments. Moreover, the code takes an extra step to check if the input is valid, such as verifying whether the capacity of the students exceeds 100. Another example is checking if the user inputs an octal number for permissions. Overall, the two programs showcase a grasp of system level programming and are well validated in order to ensure correct handling of errors.