Assignment 1
Intro to System Calls

Operating Systems and Networks

Monsoon 2021

Deadline: 30th August, 11:55 PM

There would be **NO** deadline extensions

Part 1:

Given a file, you need to reverse the file's contents and store the result in a new file in the directory named **`"Assignment"`** created in the **current working directory**. The percentage of the file written should be printed on the console during file writing. The directory created should have read, write and execute permissions for the user who created it. The new file created should have the read and write permissions for the user who created it. The program will be tested on LARGE (> 1GB) files whose size will be greater than the RAM size.

The input file path would be given to you as an argument to your program:

```
./a.out <Input_file_path>
```

The output file must be named:

```
1_<Input_file_name>
```

Example:

Input:

```
'A.txt' -> "My name is abcd"
```

Output:

```
'Assignment/1_A.txt' -> "dcba si eman yM"
```

Note : During execution, the percentage of the file written should be overwritten each time (shouldn't write multiple times).

Part 2:

Given the same file, you need to reverse the contents **of a specific part** of the file and store the result in a new file in the "**`Assignment`**" directory in the **current working directory**. The percentage of the file written should be printed on the console during file writing. The directory created should have read, write and execute permissions for the user who created it. The new file created should have the read and write permissions for the user who created it. The program will be tested on LARGE (> 1GB) files whose size will be greater than the RAM size.

The input filename would be given to you as an argument to your program: Here, the first argument is the input file path, the second argument is the number of parts the file is divided into, and the third argument denotes the part of the file that has to be reversed (1 based-indexing)

```
./a.out <Input_file_path> <No_of_parts> <Part_to_be_reversed>
```

The output file must be named:
`2_<Input_file_name>`

Example:

Input:
`'A.txt' 4 2 ->  "My name is abcde"`

Output:
`'Assignment/2_A.txt' -> " ema"`

Note: A space is also present in the output.

Explanation for Example: The input text contains 16 characters. On dividing the text into 4 parts, we get 4 blocks of 4 characters each. We need to reverse the second block of the file, i.e., "`ame `". This is reversed as "` ema`" and written to the desired output file.

Note :

- You can assume that the file is always exactly divisible into the parts mentioned.
- During execution, the percentage of the file written should be overwritten each time (shouldn't write multiple times).

Part 3:

Write a program to check the permissions for the output files from the first two parts and the directory. The input file path would be given as an argument to your program. The path of the input file is the same as the input given in the previous 2 parts.

```
./a.out <Input_file_path>
```

Example:

Input:

'A.txt'

Output:

Check permissions of the files `'Assignment/1_A.txt' (output_file_1)` and `'Assignment/2_A.txt' (output_file_2)`, and the `'Assignment' (directory)` and print them in the following format in the same order as specified above:

```
Directory is created: Yes

User has read permission on output_file_1: Yes
User has write permission on output_file_1: Yes
User has execute permission on output_file_1: No

Group has read permission on output_file_1: No
Group has write permission on output_file_1: No
Group has execute permission on output_file_1: No

Others has read permission on output_file_1: No
Others has write permission on output_file_1: No
Others has execute permission on output_file_1: No
```

Note : The above 9 should be printed for the **`output_file_2`** and **`directory`** also.

Guidelines :

1. **All Programs must use system calls only**. The use of printf and scanf are restricted
    a. Use of sprintf is **only** allowed for formatting strings and printing progress
    b. Use of string.h library is permitted
2. Useful commands: read, write, lseek, stat, fflush, perror.
3. Use man pages exclusively.
4. Assignment should be coded in C. Indent your codes and add comments wherever necessary to promote readability.
5. For error handling, please use the "`perror`" command wherever possible as demonstrated in the tutorial.
6. Add a README.md File **(compulsory)** which contains any assumptions made.
7. Submission_format: <**RollNo>_Assignment1.tar.gz**
8. Submission by email to TAs will not be accepted
9. Kindly adhere to the following naming guidelines and directory structure:

```
<Roll-No>_Assignment1
        ├── q1.c
        ├── q2.c
        ├── q3.c
        └── README.md
```

10. As the assignment would be evaluated on large files so please write optimized code to assure reasonable execution time.
11. Any copy cases found will lead to **severe consequences**