CS 232: Operating Systems

Reversing the contents of a file

The purpose of this homework is to give the student experience writing a low-level tool in C, using multiple POSIX system calls, and handling all possible failure conditions.

First:

Accept this assignment

Write a `rev` Program in C

For this assignment, you must write a program in C to copy the contents of one file to another file, but in reverse. E.g., suppose you run your program on a file **`headers`:**

```
% ./rev headers rev_headers
```

If **headers** contains the following:

```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <sys/stat.h>
```

Then **rev_headers** will contain:

```
>h.tats/sys< edulcni#
>h.dtsinu< edulcni#
>h.bildts< edulcni#
>h.oidts< edulcni#
```

Details follow:

- The code must be written in C, and compileable with gcc. Do not use C++-isms like `cout <<`, `cin >>`, etc. To print you should use `printf()`, `fprintf()`, `snprintf()`, etc.
- The program takes two arguments on the command-line: source-filename and destination-filename. Thus, the user runs the program this way from a terminal:
  `rev src dest`
  The program opens the `src` file for reading, creates the `dest` file for writing, and copies the contents of `src` (reversed) into `dest`.

- **All** error conditions must be handled and **readable** error messages must be printed out to the user. All error messages must be printed out on stderr.
- NOTE: your code does not have to check the result of each system call against every possible error code. Instead, use **perror().** See https://www.tutorialspoint.com/c_standard_library/c_function_perror.htm for a good example.
- The main program must `return 0` (and prints no message) if no error occurred, and `return -1` if any error occurred.
- If the `src` file is not a regular file (i.e., is a directory, special file, symlink, etc.), an error should be printed and `dest` should **not** be created.
- Your code should be *secure,* i.e., there should be no possibility of buffer overruns, etc.
- Hint: use the `man` command in a terminal to see how a function behaves and what kind of errors it can return. E.g., `man <funcname>` to see the information for <funcname>.
- Here are some of the C library commands in my solution:
  - `fprintf()`
  - `fopen()`
  - `perror()`
  - `fgetc(), fputc()`
  - `access()`
  - `stat()`
  - `exit()`
  - `fseek()`
- The program must execute on the Linux boxes in the Gold or Maroon lab.
- Note that `fseek()` is pretty tricky. Use `SEEK_END` to move the read pointer to the end of the file. To move to the last character in the file, use -1 for the offset.
  Also, when you use `fgetc()`, the read pointer is moved forward automatically. So, you'll have to seek backward by 2 characters to move through the file in reverse.
- Document any sources of code you find online, by putting the URL of the page in a comment above the code.
- To compile a C file called `rev.c` and produce the executable `rev`, open a terminal and use

  ```
  gcc rev.c -o rev
  ```
- To run it from the current directory do: `./rev src dest`
- To get you started, here is my main function:

```
int main(int argc, char *argv[])
{
      /* These functions exit(-1) if there is a problem. */
      check_num_args(argc, argv);
      check_src_file_is_regular(argv[1]);
      copy_src_to_dest(argv[1], argv[2]);
      return 0;   /* no error */
```

```
}
```

Checking In

We will grade this exercise according to the following criteria: (25 pts total)

- 2 pts: Header Documentation - Put your name, the date, etc., in the header of each file that you create/change. Additionally, the code must be submitted to the correct directory (see above).
- 3 pts: Code is clean and neat, with **good consistent indentation**, good descriptive variable and function names, good comments (where needed), and good spacing making the code readable.
- 5 pts: all borrowed code from the Internet is documented with a comment.
- 15 pts: code executes correctly, handles all error cases, and produces nice readable error message output when there is an error.