From CS 537 - Introduction to Operating Systems - Fall 2017

Site: Project0Shuffling

You will write a shuffling program. It should alternate printing lines starting at the beginning of the file moving forward and lines at the end of the file moving backwards. So, your output should alternate with lines from the beginning of the file and lines from the end. With this input:

```
first
second
third
fourth
fifth
```

the program should produce this output:

```
first
fifth
second
fourth
third
```

Note: this means you need to read in the entire file before producing output.

This program should be invoked as follows:

```
shell% ./shuffle -i inputfile -o outputfile
```

The above line means the users typed in the name of the shuffling program `./shuffle` and gave it two inputs: an input file to shuffle called `inputfile` and an output file to put the shuffled results into called `outputfile`.

Input files can be any text file - you can try it on your program, even.

In your shuffling program, you must use `fopen()` , `fread()` , `fwrite()` , and `fclose()` to access files. These do not read files a line at a time, so you will need to figure out how to find what is on each line.

We will provide the same automated tests that we will use to grade the program by Friday, 9/15 at 5 pm.

Hints

If you want to figure out how big in the input file is before reading it in, use the `stat()` or `fstat()` calls.

You will need to create a data structure in memory, such as an array, to hold the file contents for shuffling.

To exit, call `exit()` with a single argument. This argument to `exit()` is then available to the user to see if the program returned an error (i.e., return 1 by calling `exit(1)`) or exited cleanly (i.e., returned 0 by calling `exit(0)`).

The routine `malloc()` is useful for memory allocation. Make sure to exit cleanly if malloc fails!

If you don't know how to use these functions, use the man pages. For example, typing man fopen at the command line will give you a lot of information on how to open files.

Assumptions and Errors

- **Line length:** You can assume that the lines in the file will be less or equal to 511 characters (so 512 with the trailing null `'\0'`)
- **File length:** Input files can be any length, but will fit in memory. Two copies of the file may not fit in memory, so however you store file data in memory, you should only have one copy
- **Number of lines**: there can be any number of lines, but the number of lines will be less than the size of the file. You can use the `fstat` function to determine how long the file is, and this will tell you the maximum number of lines in the file. It is also reasonable to re-allocate data structures if the file is longer than you initially expect.
- **Line format:** All lines in the input will end in a trailing newline `'\n'`
- **Invalid files:** If the user specifies an input or output file that you cannot open (for whatever reason), the shuffle program should EXACTLY print: **Error: Cannot open file foo** , with no extra spaces (if the file was named foo) and then exit with `exit(1)` to return an error.
- **Too few or many arguments passed to program:** If the user runs `shuffle` without any arguments, or in some other way passes incorrect flags and such to `shuffle`, print `Usage: shuffle -i inputfile -o outputfile` and exit with `exit(1)` to return an error.
- **Printing errors:** On any error code, you should print the error to the screen using `fprintf()`, and send the error message to stderr (standard error) and not stdout (standard output). This is accomplished in your C code as follows:

```
fprintf(stderr, “whatever the error message is\n”);
```

General Advice

Start small, and get things working incrementally. For example, first get a program that simply reads in the input file, one line at a time, and prints out what it reads in. Then, slowly add features and test them as you go.

Testing is critical. One great programmer I once knew said you have to write 5-10 lines of test code for every line of code you produce; testing your code to make sure it works is crucial. Write tests to see if your code handles all the cases you think it should. Be as comprehensive as you can be. Of course, when grading your projects, we will be. Thus, it is better if you find your bugs first, before we do.

Keep old versions around. Keep copies of older versions of your program around, as you may introduce bugs and not be able to easily undo them. A simple way to do this is to keep copies around, by explicitly making copies of the file at various points during development. For example, let's say you get a simple version of `shuffle.c` working (say, that just reads in the file); type `cp shuffle.c shuffle.v1.c` to make a copy into the file `shuffle.v1.c` . More sophisticated developers use version control systems like git , but we'll not get into that here (yet).

Keep your source code in a private directory. An easy way to do this is to log into your account and first change directories into `private/` and then make a directory therein (say `p1` , by typing `mkdir p1` after you've typed `cd private/` to change into the private directory). However, you can always check who can read the contents of your AFS directory by using the `fs` command. For example, by typing in `fs listacl` . you will see who can access files in your current directory. If you see that `system:anyuser` can read (r) files, your directory contents are readable by anybody. To fix this, you would type `fs setacl . system:anyuser ""` in the directory you wish to make private. The dot “.” referred to in both of these examples is just shorthand for the current working directory.

Turning it in

You should only turn in one file file.

- A file, containing your code, called `shuffle.c`. We will compile it in the following way:

  ```
  shell% gcc -Wall -o shuffle shuffle.c -O
  ```

  so make sure it compiles in such a manner.

You should copy these files into your handin directory. These will be located in `~cs537-1/handin/<your CS login name>/p1/linux`. For example, if your login is `stevejobs`, you would copy your code into `~cs537-1/handin/stevejobs/p1/linux`. You can copy the file with the `cp` program, as follows:

```
shell% cp shuffle.c ~cs537-1/handin/$USER/p1/linux
```

WHAT WE WILL LOOK FOR

- Does the code work on simple files?
- Does the code work on more complex files (zero-length lines, long lines, very long files)
