Assignment 1

CS392 Spring 2019

Task 1

Task Details

Implement a simplified version of the “cat” shell command, which reads contents sequentially from the input file and prints the data to the standard output.

Requirement

1. Your program should include one source file “cat.c”. After compiling, it should produce a program “cat”. The program can run using command “./cat <inputfile>”
    - Hint: Command to compile the program: gcc -O0 cat.c -o cat

2. The program should only take one argument, which is the path to the input file. It should report an error if no argument is provided or more than one arguments are provided.
    - Hint: Needs help from the arguments of the main function

3. The program should use stream interfaces to open, read, and close the input file (e.g., fopen, fclose, fread, fgets).

4. The program should report an error and terminate the execution when an stream operation fails (e.g., failed to open the input file).

5. The program needs to use dynamically allocated memory as destination/source of stream read/write.
    - Hint: Use malloc/free

6. The program should be able to print the data from the input file, if no errors occur in stream operations. A list of test cases are provided. Make sure to test each of them.

7. The program should do necessary cleanup (e.g., close all data streams that have been successfully opened; free all the memories that are successfully allocated).

8. The program should have good comments on the key pieces of code.

9. The program should contain multiple functions.

Test Cases

https://drive.google.com/open?id=18MRNmg-39JR7tag6ZDlFxARzzP9EjcLu
Note: You need decompress the package to get all the testcases
Command to decompress the packages in VM:
$ tar -xvf testcases_new.tar.gz

Grading

Points in total: **40**

1. Cannot compile or run: **Will not be graded**
2. No statement of “I pledge my honor that I have abided by the Stevens Honor System.” as comment in the beginning of your code: **Will not be graded**
3. Pass all the test cases above: **+20**
4. Use stream interfaces: **+5**
5. Use dynamically allocated memory as destination/source of stream read/write: **+5**
6. Report errors on incorrect argument number: **+1**
7. Report errors on failure in stream operations: **+2**
8. Show error number when reporting errors: **+1**
9. Good cleanup: **+2**
10. Comments on key code: **+2**
11. Put code in multiple functions: **+2**

Task 2

Task Details

You are given a file with a sequence of unsorted integers. These integers are saved in the format of strings (strings of digits) and each line represents one integer (maximal value 4294967296). After reading those strings, your programs needs to transform them into integers (transfer each string into an integer which can be saved in a variable of type “int”), sort those integers, and save them to another file in the format of binary (save the four-byte memory of an integer to the file). Each integer should take one line in the output file.

Requirement

1. Your program should include one source file “sort.c” and one header file “sort.h”. You should include something into “sort.h”, such typedef, macro, function declaration things. Make sure you will include “sort.h” in “sort.c”.

2. The “sort.h” should use “#ifndef” and “#define” to avoid duplicated inclusion.

3. After compiling, it should produce a program “sort”. The program can run using command “./sort <inputfile> <outputfile>”, where inputfile contains the input data and outputfile will save the results
    - Hint 1: put sort.h in the same directory of sort.c
    - Hint 2: command to compile the program: gcc -O0 sort.c -o sort

4. The program should take two arguments, with the first one specifying the input file and the second one specifying the output file. It should report an error if fewer than or more than two arguments are provided.

5. The program should use stream interfaces to open, read, and close the input file (e.g., fopen, fclose, fread, fgets).
    - Hint: the maximal number in the input file will be less than 4294967296. This helps determine the size of the buffer to hold the data from one line.

6. The program should report an error and terminate the execution when an stream operation fails (e.g., failed to open the input file).

7. The program needs to use dynamically allocated memory for data storage.

8. The program should be able to generate the correct results and save the results to the output file, if no errors occur in stream operations. Test your code with the two testcases below.

9. The program should do necessary cleanup (e.g., close all data streams that have been successfully opened; free all the memories that are successfully dynamically allocated).

10. The program should have good comments on the key pieces of code.

11. The program should contain multiple functions.

Test Cases

https://drive.google.com/open?id=1piMaJKqnB5SIR1MAVsID8hAfPFqAnJrK
https://drive.google.com/open?id=1NYs62_Q0viDE9DThBHAFVcyoACbvujOM

Grading

Points in total: **60**

1. Cannot compile or run: **Will not be graded**
2. No statement of “I pledge my honor that I have abided by the Stevens Honor System.” as comment in the beginning of your code: **Will not be graded**
3. Pass all the test cases: **+40** (besides the test cases above, more undisclosed test cases will be used. They have the same format as the provided test cases)
4. Include the header file “sort.h” and properly avoid duplicated inclusions: **+4**
5. Use stream interfaces: **+2**
6. Use dynamically allocated memory for data storage: **+4**
7. Report errors on incorrect argument number: **+1**
8. Report errors on failure in stream operations: **+2**
9. Show error number when reporting errors: **+1**
10. Good cleanup: **+2**
11. Comments on key code: **+2**
12. Put code in multiple functions: **+2**