School of Computer Science, McGill University

COMP-206 Introduction to Software Systems, Winter 2020

Mini Assignment 5: C Programming - File IO

Due Date Nov 17th, 23:55

This is an individual assignment. You need to solve these questions on your own. If you have questions, post them on Piazza, but do not post major parts of the assignment code. Though small parts of code are acceptable, we do not want you sharing your solutions (or large parts of them) on Piazza. If your question cannot be answered without sharing significant amounts of code, please make a private question on Piazza or utilize TA/Instructors office hours. Please ensure to check "Mini 5 general clarifications" pinned post in Piazza before you post a new question. It might have been already clarified there if it is a popular question.

Late penalty is -5% per day. Even if you are late only by a few minutes it will be rounded up to a day. Maximum of 2 late days are allowed.

You MUST use `mimi.cs.mcgill.ca` to create the solution to this assignment. You must not use your Mac command-line, Windows command-line, nor a Linux distro installed locally on your laptop. You can access `mimi.cs.mcgill.ca` from your personal computer using **ssh** or **putty** as seen in class and in Lab A.

DO NOT edit files on your laptop; use `vim` on `mimi`. If TAs are not able to compile your code on `mimi` as-is, you will get 0 points.

Questions in this assignment require you to turn in one C program and one shell script. Instructors/TAs upon their discretion may ask you to demonstrate/explain your solution. **No points are awarded for commands that do not execute at all or programs that do not compile on `mimi`.** (Commands/programs that execute/compile, but provide incorrect behavior/output will be given partial marks.). All questions are graded proportionally. This means that if 40% of the question is correct, you will receive 40% of the grade. Any exceptions are mentioned at the appropriate part of the problem description.

Please read through the entire assignment before you start working on it. You can lose up to 3 points for not following the instructions.

Unless otherwise stated, all the names of the scripts and programs that you write, commands, options, input arguments, etc. are case-sensitive.

Total Points: 20

Ex. 1 — A Data Analytics Program(16 Points)

As you saw in assignment 3, online platforms can track users easily and study and analyze user behaviour. Such data analytics are also relevant with remote delivery of exams. In this assignment, you will analyze an user event file (`data.csv`) for such an exam generated from mycourses (anonymised) to look for students who could have possibly collaborated.

The `CSV` file has a header, and as can be seen below, contains 6 fields in each line which details the user events. The fields have consistent layout, and the file is structured in such a way that all the events related to a particular user is adjacent to each other (i.e. the events associated to a given user are always in contiguous lines in the file).

```
User, Attempt, Date, Time, Event, IP Address
Raveena Mullen,1,"Oct 9, 2020",4:54 AM,Quiz Entry,157.149.164.98
Raveena Mullen,1,"Oct 9, 2020",4:54 AM,Response to Question 1 Saved,157.149.164.98
```

```
Raveena Mullen,1,"Oct 9, 2020",4:55 AM,Response to Question 1 Saved,157.149.164.98
...
Orla Stout,1,"Oct 9, 2020",6:49 AM,Response to Question 6 Saved,51.82.149.197
Orla Stout,1,"Oct 9, 2020",6:53 AM,Response to Question 6 Saved,51.82.149.197
Orla Stout,1,"Oct 9, 2020",6:53 AM,Response to Question 6 Saved,51.82.149.197
...
Hashir Palmer,1,"Oct 9, 2020",4:51 AM,Response to Question 19 Saved,121.242.142.46
Hashir Palmer,1,"Oct 9, 2020",4:52 AM,Response to Question 19 Saved,121.242.142.46
...
```

One of the techniques to look for collaboration in the above data is to analyze the IP address of the events. IP addresses are good indicators of proximity. In this assignment, for simplification, we will define "proximity" as two users sharing the same IP address. In other words, two students will be considered to have collaborated if they have the same IP address. We will not go into the networking details of this data, as it is beyond the scope of this course - but consider COMP 535, COMP 512 if it piques your curiosity.

You can make the following assumptions (use this to simplify your code logic).

- The maximum length of any line in the file will not exceed 200 characters.
- The maximum length of a user's name will not exceed 100 characters.
- The maximum length of an IP address will not exceed 50 characters.
- A user will not be associated with more than one IP address.
- No two students have the same name.
- Events in the file are grouped by user. This means that all of user's events will appear one after another in the file. The CSV data file is not sorted though.

You will write a program, `report.c` that accepts three arguments. `<csvfile>`, `"<student name>"`, `<reportfile>`, where `<csvfile>` is the name of the data file containing the events `"<student name>"` is the name of the student we are searching for and `<reportfile>` is a text file to which the names of the collaborators will be written.

1. Use `vim` to create a C program source file `report.c` . All of your source code logic should be contained in this file. You may split your program logic across multiple functions if you would like to (recommended).
2. **(1 Points)** The source code file should include a comment section at the beginning that describes its purpose (couple of lines), the author (your name), your department, a small "history" section indicating what changes you did on what date. (Similar to the previous assignment). The code should be properly indented for readability as well as contain any additional comments required to understand the program logic.
3. **(1 Point)** The source code should be compilable by (exactly) using the command `gcc -o report report.c`
4. **(1 Point)** Compilation should not produce any warnings!
5. **(1 Point)** All error situations (described below) should result in error code 1 when the program terminates and 0 otherwise. No points for this, if this requirement is violated even for any one test case.
6. **(1 Point)** All the error messages (described below) should be printed to standard error. Again, no points for this, if this is violated for any situation.
7. **(1 Point)** Your program should accept the three arguments mentioned above. If the user does not pass sufficient arguments, or pass incorrect number of arguments, you should display an error message and terminate with error.

```
$ ./report data.csv
Usage ./report <csvfile> "<student name>" <reportfile>
$ echo $?
1
$
```

8. **(1 Point)** Make sure that the data file can be read. If not, display an error message and terminate.

```
$ ./report nosuchdata.csv "Jane Doe" rpt.txt
Error, unable to open csv file nosuchdata.csv
$ echo $?
```

```
1
$
```

9. **(1 Point)** Make sure that the user to search for is present in the data file. If not, display an error message and terminate.

```
$ ./report data.csv "Jane Doe" rpt.txt
Error, unable to locate Jane Doe
$ echo $?
1
$
```

10. **(1 Point)** If all of the above checks have succeeded, then make sure that the output file can be created. If not, display an error message and terminate.

```
$ ./report data.csv "Markus Bender" rpt.txt
Error, unable to open the output file rpt.txt
$ echo $?
1
$
```

11. **(1 Point)** If the program cannot find any collaborators for a given student, then simply write the following contents into the output report file.

```
$ ./report data.csv "Markus Bender" rpt.txt
$ echo $?
0
$ cat rpt.txt
No collaborators found for Markus Bender
$
```

12. **(1 Point)** Your program should work for both relative and absolute paths (for both the input CSV data file and the output report file).

13. **(4 Points)** If the program finds any collaborators for the student, then include their names in the output report file (one name per line).

```
$ ./report data.csv "Adaline Murphy" rpt.txt
$ echo $?
0
$ cat rpt.txt
Huma Foster
Abdullah Allen
$
```

    Do not include the original student name, only the collaborators. The output student names in the report can be in any order.

14. **(0 Points)** Make sure that the output report file does not have any duplicate student names.(I.e., do not repeat a student name in the output). **-2 Points** if any test case produces duplicates in the output.

15. **(1 Point)** Make sure to overwrite any existing output report file if it already exists.

16. **(0 Points)** If your program crashes/aborts for any test case due to bug in your logic, an additional **-2 Points** will be deducted.

17. **(0 Points)** This exercise's solution need to be completely implemented using a C Program. You may not use other Unix utilities like sed, awk, etc. Violating this will be an automatic 0. You are free to use any C library functions and header files that you need to work on this, subject to clause 3.

Ex. 2 — A mini tester script(4 Points)

There is no mini tester supplied with this assignment. Using the Unix scripting skills you have picked up so far, you can write a simple mini tester, `mini5tester.sh` for your program.

At a minimum you should include the above test cases (need not be exact, but similar in principle and objectives). Your tester does not have to automatically decide whether a test case is passed or failed, it just needs to provide enough output for a person who is running it to make that judgement without having to read code (like how the program was invoked, what did it produce, what was the return code of the program, etc.). You can look at the past mini tester scripts for some inspiration, but it need not be very complicated and thorough. This script is not going to be as rigorously evaluated as the C program.
You can assume that the `data.csv` and the `report.c` files are in the current directory. You can also store the output report files in the same directory. Can you think of way you can still test for absolute and relative paths?

HINT

For a basic solution, I suggest your separate out your logic into different functions. For example there could be a function to parse a character array containing a line in the log file and produce the name of the student and ip address associated with that line. Another function can check if the student's name exists in the data file. Yet another function can be used to go through the data file and produce the names of students who might have collaborated. You are not obliged to follow this approach.

I also recommend that you explore the C library function `strtok`.

A possible template to start your work, and divide the tasks into separate functions is given below. This is not yet a compilable code. You can use this as your base if you would like to.

```
struct logrecord
{
  char name[100];
  char IPAddress[50];
};

struct logrecord readLog(char* logline)
{
  // parse a character array that contains an line from the log
  //    and return a structure that contains the fileds of interest to us.
}

bool checkNameExists(FILE* csvfile, char* name, char* ip)
{
  // Read through the CSV data file, keep looking for the name.
  //  If found, store the IP address associated with the name
  //  to the variable ip and return success.
  // Is bool a valid data type in C? How do you indicate true/false concept in C?
}

bool findCollaborators(char* sname, char *sip, FILE* csvfile, FILE* rptfile)
{
  // Go through the CSV data file
  //  look for collaborators of sname by looking for entries with the same ip as sip
  //  if any collaborators are found, write it to the output report file.
}

int main(int argc, char* argv[])
{
  // Do any basic checks, "duct-tape" your full program logic using the above functions,
```

```
  //   any house-keeping, etc.
}
```

WHAT TO HAND IN

Turn in the C program source code `report.c`, and the shell script `mini5tester.sh` named properly. You may zip/tar the file if necessary. The file(s) must be uploaded to mycourses. DO NOT turn in the executable `report`. TAs will compile your C program on their own as indicated in the problem descriptions above. If you zip/tar your submission, you are responsible to download your submission from mycourses and verify that it is not corrupted (happens when students make an error in their zip/tar command). There will be no excuses for corrupted files.

MISC. INFORMATION

There is no tester script that is provided with the assignment, as you will be writing a simple one yourself. TAs will have their own tester script, however they will use the exact same data file, and same syntax for test cases as given in the above examples. They may however, test using different names than the ones given in the above examples.

**** Important **** If your program "hangs" / is stuck while testing (should not take more than 30s for a given test case) then you will loose points associated with it - This is already very far from the time any reasonable program should take to execute this task.

FOOD FOR THOUGHT!

- Data analytics systems often produce an output file that is more sophisticated than a simple text file (like a PDF, an image of a chart, etc.). Can your program check if the extension of the output file given as argument is `html` and produce the output as an html file (write the names in an HTML table)? This is very easy to perform and do not require any extra libraries - you just need to play around with what you write to the output file and include the required HTML tags in it - try it!