Arizona State University SER334:

Operating Systems and System Programming

Intro To C Programming II

Summary: In this assignment, you will develop a program which loads data from a file and lets a console user navigate and use that data in meaningful ways.

1 Background

You will develop a program that loads student, course, and grade data from a text file, then allows the user to perform simple queries on that data through a command line interface. The intention of this assignment is to familiarize you with reading data from files, allocating and using dynamically defined 1D and 2D arrays, and further developing your understanding of structs and pointers.

This document is separated into four sections: Background, Requirements, Data File, Running the Program, and Submission.

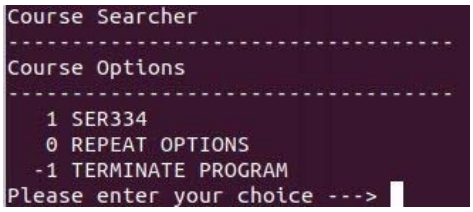
2 Requirements [36 points]

For this assignment, you will develop a program to import and handle course data stored in a text file. Your program must:

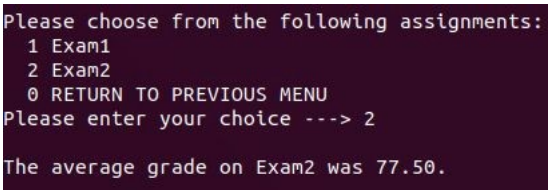
- Compile on GCC 5.3 or greater under Xubuntu (or another variant of Ubuntu) 18.04
- Have a struct type (called Student) to hold information about a student. A dynamically allocated array of these will be a global variable.
- Have an enum type (called Academic Level) that defines four values indicating a Student's grade level – Freshman, Sophomore, Junior and Senior.
- Have a struct type (called Course) which holds course information. A dynamically allocated array of these will be a global variable.
- Have a struct type (called Assign) to hold information about an assignment. A dynamically allocated array of these will be held in your course struct.
- Have a struct type (called ScoreStruct) to hold information about a student's performance on an assignment. A dynamically allocated array of these will be held in your course struct.
- Your program will support any number of courses, students, and assignments. All dynamic allocation will be dependent on values read from the provided data file.

Points for this assignment are divided amongst the following requirements:

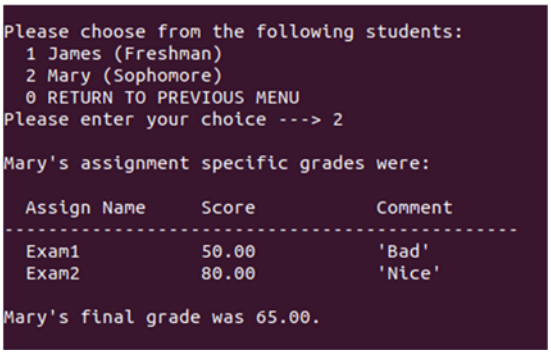
- Properly define the Student, Assign, Grade and Course structs and the Academic Level enumeration type [6 pts]
- Successfully load all of the text data into the appropriate structs [8 pts]
- Successfully display all course names in the main menu [2 pts]



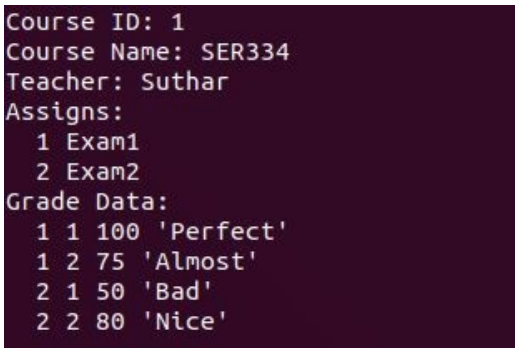
- Implement an assignmentMenu(Course course) function which prints a menu of assignments associated with the course and allows the user to choose one. With that selection, call another function getAssignmentScore(Course course, int assignmentNum) which calculates the average score for that assignment and prints it as seen below. [5 pts]



- Implement a studentMenu(Course course) function which prints a menu of students allows the user to choose one. With that selection, call another function getStudentScores(Course course, int studentNum) which prints that student's scores on each assignment, as well as their final score (the average of all of their scores) in the class [5 pts].



- Implement a printCourse(Course course) function which cleanly prints all data associated with a course [5 pts].



- Free all data before the program's termination. No memory leaks! [5 pts].

- **EXTRA CREDIT** Write a function performInstructions(char* filename) that lets users predefine commands for the program in a text file. The desired program output and input format can be seen below. More details can be found with the provided BaseReader.c code [3 pts]

```
4       -- the number of instructions to be processed
1       -- the course ID to be queried
0       -- the studentNo to be queried
1       -- the assignmentNo to be queried
1       -- the course ID to be queried
1       -- the studentNo to be queried
0       -- the assignmentNo to be queried
1       -- the course ID to be queried
1       -- the studentNo to be queried
1       -- the assignmentNo to be queried
1       -- the course ID to be queried
0       -- the studentNo to be queried
0       -- the assignmentNo to be queried
...     -- any other structions to be processed
```

```
Importing data from data/simple-data.txt

Performing instructions defined in inst.txt:

Processing instruction with courseNum 1, studentNum 0, and assignmentNum 1
The average grade on Exam1 was 75.00.

Processing instruction with courseNum 1, studentNum 1, and assignmentNum 0
James's assignment specific grades were:

  Assign Name     Score          Comment
-----------------------------------------------
  Exam1           100.00         'Perfect'
  Exam2           75.00          'Almost'

James's final grade was 87.50.

Processing instruction with courseNum 1, studentNum 1, and assignmentNum 1
Student with name James received a 100.00 on assignment Exam1

Processing instruction with courseNum 1, studentNum 0, and assignmentNum 0
Course ID: 1
Course Name: SER334
Teacher: Suthar
Assigns:
  1 Exam1
  2 Exam2
Grade Data:
  1 1 100.00 'Perfect'
  1 2 75.00 'Almost'
  2 1 50.00 'Bad'
  2 2 80.00 'Nice'

End of instructions reached. Terminating.
```

3 Reading From the Data File and the Command Line

Your program is required to read from a text file which contains a description of a set of students and courses that need to be imported into your program. The data is structured as follows:

```
2        -- the number of students defined in the file
1        -- a student ID #
James    -- a student name
1        -- a student academic level (Freshman=1, Sophomore=2, Junior=3, Senior=4)
2        -- a student ID #
Mary     -- a student name
2        -- a student academic level
1        -- the number of courses defined in the file
1        -- a course ID #
...      -- any more students that need to be read
SER334   -- a course name
Suthar   -- an instructor's name
2        -- the number of assignments defined for this course
1        -- an assignment ID #
Exam1    -- the name of an assignment
2        -- an assignment ID #
Exam2    -- the name of an assignment
...      -- any more assignments that need to be read
2        -- the assignment number associated with a score
1        -- the student number associated with a score
75.0     -- a double representing the percent the above student received on the above assignment
'Almost' -- a comment associated with the above data
1        -- the assignment number associated with a score
2        -- the student number associated with a score
50.0     -- a double representing the percent the above student received on the above assignment
'Bad'    -- a comment associated with the above data
1        -- the assignment number associated with a score
1        -- the student number associated with a score
100.0    -- a double representing the percent the above student received on the above assignment
'Perfect'-- a comment associated with the above data
2        -- the assignment number associated with a score
2        -- the student number associated with a score
80.0     -- a double representing the percent the above student received on the above assignment
'Nice'   -- a comment associated with the above data
...      -- any more grades that need to be read
...      -- any more courses that need to be read
```

The example data used above is provided with the assignment, annotated and not, for you to test your program. It is assumed that every student has a grade for every assignment in every course.

Since the data is stored as an ASCII text file, all values can be read from the file using the function *fscanf(FILE * stream, const char * format, …).* Some helpful resources to understand how this function works can be found at:

a. http://www.cplusplus.com/reference/cstdio/fscanf/
b. https://www.tutorialspoint.com/c_standard_library/c_function_fscanf.htm

Similarly, for some of the functionality your program is expected to support, you will need to be able to query the terminal user for menu selections. This requires that you use the function *scanf(const char * format, …)*. The provided code has several examples of this function in action but you may find the following resources helpful:

a. http://www.cplusplus.com/reference/cstdio/scanf/
b. https://computer.howstuffworks.com/c7.htm

A good resource on getopt functionality : https://www.cs.rutgers.edu/~pxk/416/notes/c-tutorials/getopt.html

4 Running the Program

It is suggested that you compile and run this program from the terminal. In Ubuntu based systems, the terminal can be opened with the shortcut CTRL+ALT+t. After the terminal opens, you can perform the following steps to run the program:

1. You will start in your root directory. Navigate to the directory containing your source code and text data with the commands cd, ls, .., etc… If you are unfamiliar with this process and these commands, refer to the related instructional materials provided in the course shell.
2. Once in the correct directory, we want to compile BaseReader.c. To do this we will execute the GCC compiler suite on the source file. The simplest command to do so is "**gcc BaseReader.c -o reader**", which will compile the file BaseReader.c and save the execute output as 'reader'.
3. Normally, to run the file we have built, we just need to enter "./reader". The dot slash means look for a file in the current folder. However, this program requires the name of the file containing our

text data so we need to offer that somehow. With this code in particular, the author has chosen to use ‘getopt’, a function that lets you use ‘options’ as a means of passing in information or flags. To run the program, we will type “**./reader -d simple-data.txt**”.

To read more about getopt (you will need it in the next assignment, too) you might find the following resources helpful:

a. https://www.tutorialspoint.com/getopt-function-in-c-to-parse-command-line-arguments
b. https://www.gnu.org/software/libc/manual/html_node/Example-of-Getopt.html

In addition, do your best to follow and understand the given code – it will help you down the line.

6 Submission

The submission for this assignment has one part: a source code submission. The file should be attached to the homework submission link on Canvas.

Writeup: For this assignment, no write up is required.

Source Code: Please name your main class as “LastNameCourseReader.c” (e.g. “AcunaCourseReader.c”).

• If your program fails to compile out-of-the-box, there will be a mandatory deduction 20% from the assignment points.

• If you do not follow the file submission standards (e.g., the submission contains project files, lacks a proper header), there will be a mandatory deduction of 10% from the assignment points.

• If compiling or running your program differs from stated on the assignment and it is not specified as a requirement, please include a readme file with steps to execute the program. If you do not, there will be a mandatory 10% deduction from the assignment points.

Example readme

To run the base program:
gcc BaseReader.c -o reader
./reader -d ./data/<data_txt_file>

To run the complete program with CLI:
gcc CompleteReader.c -o reader
./reader -d ./data/<data_txt_file>

To run the complete program with file instructions:
gcc CompleteReader.c -o reader
./reader -d ./data/<data_txt_file> -i ./inst/<inst_txt_file>