CPE 100 Introduction to Computer Programming
International Sections August 2020
Laboratory Exercise 11

Objective

This lab gives you practice using structures, and introduces the reading and writing of binary files.

Instructions

Summary: Create a program called **readwrite.c**. This program should read all the data from my binary file **students.dat,** into an array of **STUDENT_T** structures. (The **STUDENT_T** datatype is defined in my header file **student.h**.) The program should print the information for each student on the screen. Then it should write all the student structures back to another file, **newstudents.dat**. If your program is correct, the new file should be identical in size and content to the original file.

Details

1. Download the input file (**students.dat)** and the header file (**student.h**) from the course website.

2. Look at the contents of header file. It defines a structure called STUDENT_T which you will use in this assignment. You should ***include*** this header file in your program, which you should call **readwrite.c**. ***You should not define or copy the structure in your own program*.**

```
#include <stdio.h>
#include "student.h"   /* for non-system headers, you use quotes around the header file name */
```

3. Try to type out the input file **students.dat.** You should see mostly garbage characters. This is because **students.dat** is a binary file. Try running the Linux 'file' command on **students.dat**.

4. In your program, declare an array of STUDENT_T records to hold the data you will read. This array should have MAXCOUNT elements. (MAXCOUNT is defined in **student.h**. Don't redefine it in your program!)

5. Open the input file, using the **fopen** function. Read STUDENT_T records from the file, one at a time, into your array, using the **fread** function. Stop when you have read MAXCOUNT records or when the **fread** function returns 0 instead of 1. (*Hint: This means your loop will have two stopping conditions.*) Close the file.

6. Create a function called **displayStudent.** This function should accept as an argument a single STUDENT_T structure. It should print all the information in that structure in some nice format.

```
void displayStudent(STUDENT_T  oneStudent);
```

7. In your main function, loop through all the students in your array. Call **displayStudent** for each one. This will allow you to check that your program read the data correctly. (If you prefer, you can call **displayStudent** immediately after you read each student's information.)

8. Open a new binary file called **newstudents**.**dat**, for writing. Write all the records in your array to that file. You should do this with one call to **fwrite**. Do not use a loop! Don't forget to close the file after writing to it.

9. Compile and test your program. The file your program creates should be exactly the same size and content as the input file **students**.**dat**. Use the **diff** program (or the **fc** program on Windows) to check this.

10. Upload your C source file and the output file **newstudents**.**dat** to the web server.

IMPORTANT NOTE: If you try to complete this lab on Windows, Mac OS/X or on a 32-bit version of Linux, you will have problems. The version of **students.dat** on the course website was created in 64-bit Linux and will not be compatible with your environment. In this case, you should create a new version of **students.dat** by compiling and running (on your computer) the program ***writestudents.c***, which you will find on the website. This will create a new version of **students.dat** appropriate for your computer. You can then run your own program to create **newstudents.dat**.