COMP2100 Assignment 1 FAQ

Richard Han, richard.han@mq.edu.au 26 July 2021

General

Where can I find out more about the errors that the automarker is reporting?

Refer to the file `marking-guide.txt` which is included in each stage tar file that you download from the lab command. This file describes each of the marking elements and how the mark is computed.

What is the Stack Smashing error reported by my program?

The gcc runtime system detects a runtime error called "stack smashing". This occurs when your program overwrites information on the subroutine call-and-return stack so that it damages the stack. This error can result from different problems including accessing outside the bounds of an array, following an invalid pointer, or returning a pointer to data in the stack. To understand this error fully, you need to know how the stack is used for local variables inside a procedure – that topic is covered in the second half of this unit. However, you can and should interpret "stack smashing" error as indicating a problem with array bounds or with pointers.

Stage 1

What am I supposed to do in this assignment?

There is a lot of documentation to support you in this assignment. Start by reading the assignment specification from beginning to end. Don't read it looking for specific information the first time. Read it to find out what information is in there.

Towards the end there is a section for each stage of the assignment. If you are just getting started, you will be working on stage 1. But there is important information earlier in the assignment specification to help you get started, so don't just read the section on stage 1!

After you've read the assignment, you should realise that you need to write C programs that work with structs and data files. In stage 1, you will write a C program that declares a struct, initialises it with specific values, and prints out the contents of the struct in exactly the way shown in the sample output. You should know how to find out your specific task details: what fields (members) are in your struct, what types they are, and the initial values.

There are support documents on iLearn that will also help you once you know what you need to do. They include:

- *Lab Command Manual*: How to use the lab command, to fetch and submit your stages, to check your marks, to claim free extensions, and other things you might need to do. Highly recommended.
- *Decimal, Binary, Octal and Hex*: How to convert between the different forms of integer numbers; how negative numbers are represented in the computer; and other relevant information about integers.
- *C Programming Notes for Data File Lab*: Covers important topics
    - `struct`: What a struct is, how to declare one in C, how to initialise the values of a struct, and other important information.

    - `printf`: How to do formatted printing in C. Formatters for different data types, for decimal, octal and hex representations of integers. Object size specifiers.
    - The `main` function and command line parameters.
    - Pointers, sorting and memory allocation, with an detailed example and discussion of sorting.
- *Compile, Run, Make C Programs on Linux*: How to compile and run your programs under Linux.

The initial value of my field is supposed to written as (or converted to) hex or octal?

- *Also: The value in my field is written as hex/octal but the output is a decimal number?*

A key concept to remember is that

- All data is just bits inside the computer.

We write integers in decimal, but the computer stores them in binary. It does not matter whether we write the value as decimal, or as octal or hexadecimal – if the value that we write corresponds to the same integer then the binary representation will be the same! All the computer cares about is the bit patterns in memory. This is quite a powerful concept, and C exposes you to the bits much more than other high-level languages.

For example, the following statements all initialise the integer `a` to the decimal value 65 which is `0000 0000 0000 0000 0000 0000 0100 0001` in binary.

```
int a = 65;   // Decimal
int a = 0x41; // Hexadecimal
int a = 0101; // Octal
int a = 'A';  // Character constant
```

All the four examples above result in exactly the same binary representation inside the computer.

In stage 1, you are asked to initialise the fields of your structure with some non-decimal initialisers.

The document "*Decimal, Binary, Octal and Hex*" explains how to convert between the different representations of integers.

My expected-output.txt does not make sense. Is there an error in the assignment?

- *Also: The value in my field is decimal but the output is not a decimal number?*

There are three tasks in stage 1: declare the struct, initialise the struct with specified values, and print out the values. These are separate tasks. Declaring the struct specifies the types of each field – whether it is integer, floating point or a character string; and how big it is. Initialising the struct loads values into the fields. Printing formats the values according to the printf formatters.

The initialisation-specification.txt tells you what notation to use for integer values (decimal, octal or hex). In the end, it makes no difference to what is stored in the computer memory. The automarker actually interprets your C code to check that you have followed the instructions.

The expected-output.txt file shows you what the output of printf should be. You use formatter to specify whether you are outputting an integer, floating-point or character string. For integers, you can also use formatters to output them as decimal, octal or hexadecimal, or even as a character. Formatters can also specify how large the object is in memory. See the document "*C Programming Notes for Data File Lab".* Some other questions below may also be relevant.

So, it is not an error if the initialisation value is specified as decimal but the final output is hexadecimal or octal. Similarly, it is not an error if the initialisation value is octal or hexadecimal but the final output is in a different format. The two things are quite separate from each other.

My number is negative but the output looks like hex?

The document "*Decimal, Binary, Octal and Hex*" discusses negative numbers. Remember that "all data is just bits inside the computer" and, if you know the bit pattern that represents a negative number, you will be able to understand what it looks like as hexadecimal.

See the document "*C Programming Notes for Data File Lab".*

My output is supposed to be two or four hex digits but it comes out as eight?

There are modifiers for printf formatters that specify the size of the data object. You need to tell printf what representation you want to use (e.g. decimal, octal, hex) but also the size of the integer that you are passing to printf. C will convert anything shorter than an int to int when it passes it to printf (or to any other subroutine).

See the documents "*Decimal, Binary, Octal and Hex*" and "*C Programming Notes for Data File Lab*" for more information.

How can I print out the field names, as shown in the sample output?

In C, you just print a literal string.

C does not have reflection (the capability, in other languages such as Java, to find out the names of members of a class). There is no worthwhile way to associate each of the field names, as a string, with the fields in the structure declaration or initialisation. Just use a C string to print out the names.

Of course, the line of data values should be printed from the struct fields using printf with formatters.

I get an error when submitting "the file appears to be windows text"

A text file consists of lines of printable characters, with some special character(s) at the end of each line. In Unix (including Linux), the end of line is represented by the ASCII control character LF (look up an ASCII table online for more detail). In Windows, the end of line is represented by a sequence of two characters: CR followed by LF. The reason for the difference is historical. If a Windows text file is transferred to Unix, the extra CR characters need to be removed, otherwise the Unix programs see annoying CR characters at the end of every line of text.

The lab command checks the file that you are submitting, and gives you an error if it is not a simple Unix text file. You can convert the file to Unix format on your Windows machine using the Notepad++ utility (a highly recommended editor for program code on Windows).

This error should not occur if you are editing your source code directly on ash or iceberg servers (e.g. using vi/vim). It should also not occur if you are working on a Mac because Mac runs Unix underneath.

Stage 2

I'm not getting marks for printing the error message or my usage information?

There are style requirements for systems programs and for error messages in particular. Please read the style guide documents: *Some Important Comments on Code Style* and *Systems Programming Style*.

I am not getting the mark for printing the error message when file does not exist?

One of the errors that a user of your stage 2 program could make is to name a file that does not actually exist. (This is a different error condition than if the user does not name of file on the command line at all.) This user error may happen because they mistype the name, or for any other reason, but the reason is not important. What is important is, if the file that is named on the command line does not exist, you need to provide an error message consistent with the style requirements for systems programs and for error messages in particular. Please read the style guide documents: *Some Important Comments on Code Style* and *Systems Programming Style*.

My program prints out the last row of data twice?

This problem typically arises from not recognising end-of-file correctly. In particular, if you use `feof()`, you need to know how it works. It probably does not do what you think it does.

Unix does not report end-of-file when the file cursor is after the last character of the file – it only reports end-of-file when you try to read past the end of the file. You might think that the following loop would read all the characters of a file `infile`, and print them out, stopping when it reaches the end-of-file.

```
while (! feof(infile))
    putchar (fgetc (infile));
```

However, this code does not stop after the last character in the file! It actually attempts to read after the end of the file. When it does, `getc` will return `EOF`, and then `putchar` will output `EOF` as if it was a character, so the output will be incorrect.

The code will only perform correctly if we check for `fgetc` returning `EOF`. We can do that easily – here is one way to write it.

```
int c;
while ((c = fgetc(infile)) != EOF)
    putchar (c);
```

Why is it so? In Unix, there are not only disk files. There is also terminal input which is a file, and you can read from a network connection as a file. When the input comes from a terminal, the program cannot know whether the user has decided to type the end-of-file character (control-D in Unix) until it tries to read input and the user types that character. Similarly, when reading from a network connection, you cannot know whether the program at the other end is going to send more data or close the connection until you actually attempt to read input and instead find that the connection is closed.

Stage 3

My data is not sorting correctly?

There can be many reasons for errors in the sorted output. You need to debug your sorting.

One thing you should try is to look at the output that your program is producing and compare it with what is expected. How can you do that? What tools do you have available to help you view the contents of a binary file? (Hint: stage 2).

There can be many reasons for errors. If your binary output file has the correct rows of data in it, but they are in the wrong sequence, then your error would seem to be specifically in your comparison function that is used by qsort. You will get partial marks if the sorting is partially correct and, more importantly, you should be able to work out what your program is actually doing and then work out how to fix it. Check the sorting order for the columns, and try to identify whether one particular column is being sorted incorrectly. Because you are using lexical sorting, an error in one column will usually affect other columns also. The automarker will help you by identifying columns that are not sorted correctly.

You will gain the best debugging skills if you try to understand what your program is doing that is actually wrong. If you don't understand the mistake, you may not fix it properly. Also, in your career you will need to read and understand other people's code, so practice reading and understanding your own code! Other people's code is much harder to understand, but you have to start somewhere…

If you cannot recognise the data in your binary output file, it might help to dump the file using the `od` command. You can look at all the bytes as hex, as ASCII, etc. If all the bytes are zero, or there are obvious repeating patterns, then the patterns may help you identify what kind of thing is going wrong.

If the data in your binary output file is garbage, or does not make sense, one thing to look for is problems with pointers or memory allocation in your code.

But, there can be many reasons for errors. And there are many approaches to debugging. If you cannot work out what is happening from examining your binary output file and comparing it with the binary input file, then you can use a debugger such as `gdb` to step through the execution of your program and watch it actually sort the data; or you can add debugging statements to your code to see what is happening inside it. Just remember to delete those debugging statements when you don't need them any longer – don't leave "dead code" (including debugging statements) in your code.

My question is not answered here?

First read the questions above. You might be thinking about the same thing but in a different way.

You can ask questions on the forum (just be careful not to post your code or give away the solution to part of the assignment). You can ask question in a workshop class or in email to the unit convenor. If your question is similar to ones that other people ask, it may end up appearing here.

Len Hamey 2020