SEEM3460 Assignment 1

Chang GAO
gaochang@se.cuhk.edu.hk

Pls ensure the followings:

- **Use CUHK network if you are using your own computers**
  - Otherwise you won't be able to connect to the servers
- **Connect to our remote severs**
  - linux03.se.cuhk.edu.hk
  - linux04.se.cuhk.edu.hk
  - linux05.se.cuhk.edu.hk

Overview

- ❏ Review of last tutorial
- ❏ Assignment 1
- ❏ Some tips
- ❏ Submission

SEEM3460 Tutorial

Compiling and Debugging C Programs in Linux

Chang GAO
gaochang@se.cuhk.edu.hk

Download materials for tutorial 2

- Log in Linux machine (linux03~05)
- Type the following commands:
  - wget http://www1.se.cuhk.edu.hk/~seem3460/tutorial/c_debug/tutorial-02-2021.zip
  - unzip tutorial-02-2021.zip
- The folder “tutorial-02-2021” at current directory contains all the materials for this tutorial
- P.S. It is also available on the course website

Compiling C programs in Linux

- Compiler: `gcc` – GNU C Compiler, freeware
- Method 1: `gcc filename`
  - file “a.out” will be generated in the current working directory
  - example: gcc reverse.c
- Method 2: `gcc inputFileName –o outputFileName`
  - You can customize outputFileName
  - example: gcc reverse.c –o reverse1
  - gcc reverse.c –o reverse1.abcde
- Run(execute) the program: `filename`

Example

- Use a text editor to create a **hello.c** file with the following content, compile with gcc and run the compiled program to see the output

```
#include <stdio.h>
int main() {
  printf(“Hello World\n”);
}
```

- **Note:** Copy and Paste may produce strange characters in your editor, so try to type the code by yourself.

Debugging C programs in Linux

- What is bug ?

  “grammar mistakes”

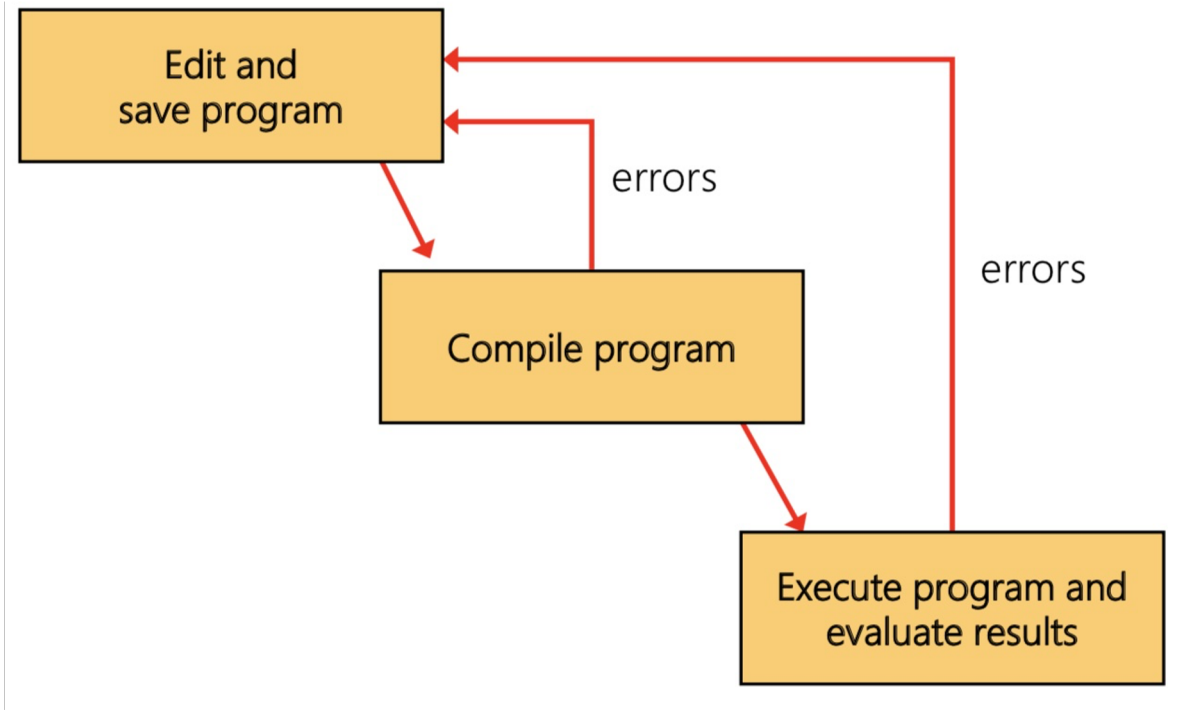
  - Compilation Error or Syntax Error

  “compile successfully but do not output expected result”

  - Runtime Error or Logical Error

- In essence, debugging is to find bugs and fix them

Basic Program Development



Debugging C programs in Linux

- How to debug?
  - Output values of variables (eg. use `printf`)
    - easy to do and effective
    - popular among experienced programmers
  - Use debugger to find bugs

General scheme of debugging

- Step 1. read the source code and understand purpose of the program roughly. (sometimes author will explain in comments or documentation)
- Step 2. try to fix obvious bugs based on your knowledge (eg. syntax error) (use an editor such as nano and vim)
- Step 3. compile the program and see warning messages (-Wall).
- Step 4. locate the lines that may have problem according to warning messages and try to find out the error.
- Step 5. revise until program compiled successfully
- Step 6. execute the program and check if the result is correct
- Step 7. if there are some logical errors, print the values of related variables or use debugger
- Step 8. revise until program can output correct result

Debug by inserting `printf`

- Lab practice: compile `reverse2.c` and debug

- Follow the steps mentioned in “general scheme of debugging” in last slide.

Assignment 1

- Extend the C program **coord.c** in the lecture note
  - The extension is to ask the user to enter three coordinate points. The program will decide whether the triangle formed by the three points is an equilateral triangle (i.e. the three lengths are equal). If yes, output “Yes, it is.” on the terminal. Otherwise, output “No, it is not.”
- To achieve the above aim, a function **check()** should be added
  - This function will call the original function distance() to achieve the aim
- The function **main()** should be completely re-designed.
  - It should allow the user to enter three coordinate points to test whether the function check() you write is correct.

Sample code structure

```
float distance(COORD a,
COORD b)
{
:
}

COORD getcoord(void)
{
:
}
```

```
int check(COORD a, COORD
b, COORD c)
{
:
}

int main(void)
{
:
}
```

Assignment 1

- You MUST keep the name of the function **check()**
- You need to add comments to your code.
- You are encouraged to design a good interaction.
  - Clear and easy to understand
- You can copy and modify the code in the lecture note to achieve your aim.
  - For example, you can change the function getcoord() to tailor the prompt message for input.
  - Apart from this, you should write all the code yourself

Assignment 1

- **Do not plagiarize!!!**
- If a student plagiarizes, there will be a heavy punishment. The definition of plagiarism includes copying all or part of programming assignments from other's work or the Web. The penalty will apply to both the one who copies the work and the one whose work has been copied.

Sample execution

- The following is an example that is not an equilateral triangle

```
Please enter three points (y) or quit (q): y
Enter x and y coordinates of the first point:
1 1
Enter x and y coordinates of the second point:
1 1
Enter x and y coordinates of the third point:
1 1
Output: No, it is not.
```

- The following is an example that is an equilateral triangle

```
Please enter three points (y) or quit (q): y
Enter x and y coordinates of the first point:
0 0
Enter x and y coordinates of the second point:
2 0
Enter x and y coordinates of the third point:
1 1.732
Output: Yes, it is.
```

Some tips

- When comparing whether two floating-point numbers are equal, it is not appropriate to use “==”
  - A better way is to see if their difference is less than a certain small value (e.g. fabs(d1-d2)<1e-3)
- You should consider some special cases
  - For example, all three points entered by the user are the same point. The distance between any two points is equal, but they cannot form a triangle.

Some tips

- You should test at least the following four examples
  - (1, 1) (1, 1) (1, 1) No, it is not.
  - (0, 0) (2, 0) (1, 1.732) Yes, it is.
  - (0, 0) (1, 0) (2, 1) No, it is not.
  - (2, 1) (1, 0) (0, 0) No, it is not.
- If you want to use **sqrt()** or **fabs()**, you should #include <math.h>
  - The math library corresponding to the standard header <math.h> is not linked by default, and an undefined function error will occur if it is not added manually.
  - You should use the following command
    - gcc –lm <C file>

Sample Linux Interaction Output

- The purpose of this is to ensure that you will compile and execute the program in Linux platform
- The Linux command "**script <sample-output-filename>**" should be used to capture the terminal data and information.
  - Once the "script" command is issued, it will set up an environment. From this moment, users can continue to use Linux. Whatever the user types and information displayed on the terminal will be automatically saved into the <sample-output-filename>, which will be treated as the Sample Linux Interaction Output for later submission.
  - Use CTRL-D to quit the terminal capture

Sample Linux Interaction Output

- The file content MUST contain the followings
  - The command “date”
  - The compilation of the source program file
  - The running of the executable file with various sets of sample input and output
- **Note:** You are encouraged to design a good interaction.

Submission Procedure

- Submit two files for grading, namely the Source Program and the Sample Linux Interaction Output
  - The file name for the Source Program MUST be named <student-ID>-assgn1-source.c (e.g. 1155XXXXXX-assgn1-source.c)
  - The file name for the Sample Linux Interaction Output MUST be named <student-ID>-assgn1-interact.txt (e.g. 1155XXXXXX-assgn1-interact.txt)
  - Put these two files into the same folder named <student-ID>-assgn1 and zip the folder. The zipped file MUST be named <student-ID>-assgn1.zip (e.g. 1155XXXXXX-assgn1.zip)
- Follow the steps in the assignment description to submit

Question & Answer

- With any questions about the assignment, please check the following Google Document for Q&A first and write only new questions that have not been asked yet:
  - https://docs.google.com/document/d/1xo_JPAgWIoX7OqIaEAQhLYyde566ILfkK3dHE7o03G4/edit?usp=sharing