SEEM3460 Tutorial

Compiling and Debugging C Programs in Linux

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

❑ To compile a C program

❑ To debug a C program

❑ Lab practice

Required Software

- ❑ **SSH client (required)**
  - ❑ **PuTTY** (FREE)
  - ❑ SSH Communications Security
  - ❑ ***Update:*** The built-in SSH client is now enabled by default in Windows 10's April 2018 Update, you can now connect to an Secure Shell server from Windows without installing PuTTY if you are using the new version.

Review: Useful commands for Linux

- ❏ ls: to list files in the directory
- ❏ pwd: print the path of current working directory
- ❏ cd: go to another directory (change working directory)
- ❏ cat: view content of file
- ❏ mv: move file
- ❏ rm: delete file
- ❏ cp: copy file
- ❏ wget: download file from the Web

Download materials for this tutorial

- ❑ 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.

Compiling C programs in Linux

- ❑ General case: to compile multi-module C program
- ❑ `gcc file1 file2 … fileN -o outputFileName`

- ❑ Only compile source code file (.c) , header file(.h) need not to be mentioned because they should be included in .c file
- ❑ example: `gcc part1.c part2.c -o program1`

- ❑ C Design Guideline: `.h` file contains C function declarations and macro definitions to be shared between several source files. `.c` file contains C function implementation

Debugging C programs in Linux

- ❑ What is bug ?

  “grammar mistakes”

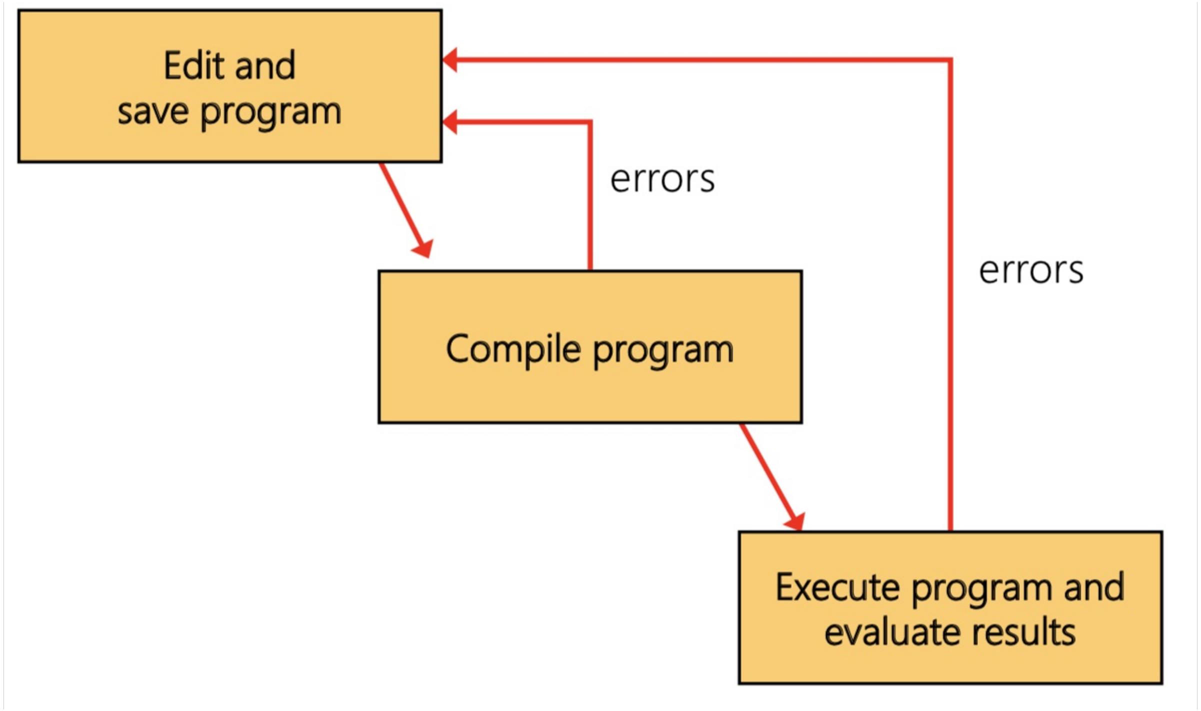
  - Compilation Error or Syntax Error

  “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.

Debug by debugger

- Debugger lets you to know:
    - Which statement or expression did the program crash on?
    - If an error occurs while executing a function, which line contains the call to that function, and values of parameters
    - What is the value of a particular expression/variable in a program?

Debug by debugger

- ❑ Debuggers available on your Unix workstations: **gdb**

- ❑ To use debugger, add “-g” flag when compiling the program
- ❑ example: `gcc -g reverse.c -o reverse`
- ❑ “-g” means “record extra information while compiling”, it’s used by gdb to locate and set breakpoint

- ❑ And then start the debugger by typing: gdb
- ❑ For detailed tutorial , see folder “gdb-tutorial”

Useful commands on gdb

❑ file filename	- load an executable file
❑ r(or run)	- run the program
❑ q(or quit)	- quit
❑ b(or break) functionName/address/lines	- set a breakpoint
❑ c(or continue)	- continue
❑ p(or print) variable	- print the value of variable
❑ s(or step)	- step into a function

Debug by debugger

- ❑ Step 1. type “gdb” in command-line mode
- ❑ Step 2. type “file filename” to load an executable file
- ❑ Step 3. type “break functionName/address/lines” to set a breakpoint
- ❑ Step 4. type “run” to execute the program
- ❑ Step 5. when program stop at breakpoint, type “print variableName” to see the current value of a variable or type “watch variableName” to track the value of a variable
- ❑ Step 6. type “c” to continue running until program terminated
- ❑ Step 7. type “q” to exit from gdb