Lab 1 GCC, GDB, Man, Make, Git, Simple C Program

1 Overview

The purpose of this lab is to get used to a terminal based development environment you will be using for the class.

2 GCC or Clang

GCC will be our primary compiler for this class. You can play around with it with the lab1.tar file which you can download here and extract its contents into a local directory (tar -xvf lab1.tar).

Before we play with make. Lets learn the compiler first and some helpful flags.

- -Wall : Turn on all warnings
- -Werror : Turn warnings into errors
- -O3 : Turn on the highest optimization setting
- -E : Run the preprocessor
- -S : Turn into assembly

Run these flags on the given lab1.c file in the tar.

3 Make

make is a program that is commonly used to automate compiling. More specifically, make executes commands based on a set of rules. These rules are defined as a set of dependencies and a target. If any of the dependencies is fresher (the timestamp is more recent) than the target, then the commands for the rule are executed. This is incredibly useful when a program is separated into many compilation units.

Examine the contents of Makefile (you can do so by executing more Makefile). You should see a number of rules that specify, in essence, how to build a program (though make is used for more general purposes).

Type make in the directory that contains the Makefile and the source files. Notice which commands were executed. Now type touch fact.c (touch updates the timestamp on a file). Again, type make. By viewing the Makefile, you should notice that only those commands for rules that depended on fact.c were executed.

4 GDB

```
% gdb example
...
(gdb) b main
(gdb) run
(gdb) print i
(gdb) s
(gdb) print i
(gdb) s
(gdb) list
(gdb) print n
(gdb) back
(gdb) b fact.c:7
(gdb) c
(gdb) print n
(gdb) back
(gdb) c
(gdb) c
(gdb) back
(gdb) c
(gdb) c
(gdb) back
(gdb) c
(gdb) c
(gdb) back
(gdb) quit

% gdb example
(gdb) b main
(gdb) r
(gdb) watch i
(gdb) c
(gdb) c
(gdb) c
(gdb) c
(gdb) c
```

```
(gdb) c
(gdb) c
(gdb) quit
```

gdb supports many more features than those shown here. You can type help at the gdb prompt to get a list of the features. You might also find it useful to use n instead of s to step through statements. The difference is that s will step into a function invocation whereas n will step over a function invocation. More interesting uses will be demonstrated once we discuss pointers and structures. Consider the following.

```
% gdb example
(gdb) b main
(gdb) r
(gdb) print &i
$1 = (int *) 0xXXXXXXXX <--- not actual output, keep track of the real number
(gdb) b fact
(gdb) commands 2
Type commands for when breakpoint 2 is hit, one per line.
End with a line saying just "end".
>print *(0xXXXXXXXX) <--- actual number from above
>end
(gdb) c
(gdb) c
... continue until you understand exactly what is happening
```

Demonstration After you have completed the above gdb example, call me over so that I can record your completion of this lab. You can (and should) continue with next two parts while waiting or after your demonstration.

5 Man

The manual pages are a helpful resource when you need to find additional information about commands and standard C library functions.

On a Shell prompt, you should run the following commands and read as much as you want about it.

```
// Basic Commands
% man ls
% man mkdir
% man cp
% man mv
% man rm

// Programs used in this lab
% man vi
% man tar
% man more
% man make
% man touch
% man gcc
% man gdb
% man man
% man git

// Few C library functions
% man getchar
% man putchar
% man printf
% man strcpy
```

6 Git

In the course, you might want to set up a git account. Because all projects in this class is individual work, please set up an account at Bitbucket. Here you have an unlimited amount of private repositories. Feel free to ask a TA on how to set one up if you are stuck.

You can follow Github's Tutorial

Then you should try to set up a repo for the next part with BitBucket.

7 Simple C Program

You will write a program that will take in files via commandline argument and report any lines over 80 characters. Print out which file its from, the line number and the length of the line.

An example output would be:

```
% ./Linecount main.c other.c
 In main.c, line 15 has a line length of 85
 In other.c, line 85 has a line length of 81
```

When you are done, please demo it to a TA.

8 References

Parts 3, 4, and 5 were influenced by Professor Aaron Keen's 357 homework 1.