Linux Users Group

UT Arlington
www.luguta.org

How to compile C/C++ programs on Linux (basic)

Rohit Rawat

Linux User Group

- Presentations downloadable from:

  http://luguta.org/

  - Installing Ubuntu in a VM
  - Accessing the Omega server
- MavOrgs:

  https://mavorgs.collegiatelink.net/organization/luguta

Outline

- **GNU Compiler Collection**
  - **Suite of compilers – C, C++, Fortran,Java..**
- GNU make
  - Utility to automatically execute build steps
- GNU debugger
- C++ and Java IDEs for Linux, Mac, and Windows.

What is GNU?

- GNU – recursive acronym for Gnu's Not Unix
- Announced by Richard M. Stallman in 1983
- Collection of free software – compilers, editors, utilities, kernel
- GCC - first free C compiler in 1987
  - it paved the way for other free programs
- He created the GNU General Public License (GPL)

GCC C/C++ front-end

- gcc inputfile.c
- g++ inputfile.cpp
- Default executable name 'a.out'
- To run:

  ./a.out

  (if a program is not on the system path i.e. in /bin, /usr/bin, etc., you can't run it just by its name. './' points to the current folder)

GCC C/C++ front-end

- Don't have gcc installed?
  - Ubuntu:

    sudo apt-get install gcc g++
  - Mac:

    Install Xcode & command line tools
  - Windows:

    Install Cygwin or MinGW GCC

Common gcc/g++ options

- -o outputfile (give an alternate name for a.out)
- -c (compiles only, doesn't link)
- -O3 (optimization level, 3 = high)
- -g (add debugging information)
- -Wall (displays all warnings)

Common gcc options

- -I includepath (additional locations to look for include files)
- -L librarypath (additional locations to search for libraries)
- -l libname (link against that library, e.g. -lm -lboost_system)

- Over 9000 options!

pkg-config (when using external libs)

- Unified interface for configuring include directories, library paths, and linker flags
- Example: compile and link OpenCV
  - Without pkg-config (linking all modules):

```
g++ main.cpp -I/usr/local/include/opencv -I/usr/local/include  /usr/local/lib/libopencv_calib3d.so /usr/local/lib/libopencv_contrib.so
/usr/local/lib/libopencv_core.so /usr/local/lib/libopencv_features2d.so /usr/local/lib/libopencv_flann.so /usr/local/lib/libopencv_gpu.so
/usr/local/lib/libopencv_highgui.so /usr/local/lib/libopencv_imgproc.so /usr/local/lib/libopencv_legacy.so /usr/local/lib/libopencv_ml.so
/usr/local/lib/libopencv_nonfree.so /usr/local/lib/libopencv_objdetect.so /usr/local/lib/libopencv_ocl.so /usr/local/lib/libopencv_photo.so
/usr/local/lib/libopencv_stitching.so /usr/local/lib/libopencv_superres.so /usr/local/lib/libopencv_ts.a /usr/local/lib/libopencv_video.so
/usr/local/lib/libopencv_videostab.so -lrt -lpthread -lm -ldl
```

  - With pkg-config:

g++ main.cpp `pkg-config --cflags --libs opencv`

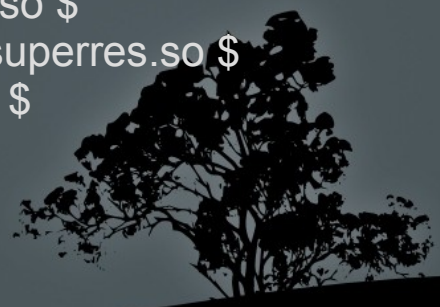
- Not available on Omega :(as they don't allow external libs
  - You can still do 'local installs' of some libraries

pkg-config

$ **sudo apt-get install libopencv-dev**
$ **cat /usr/local/lib/pkgconfig/opencv.pc**
# Package Information for pkg-config

prefix=/usr/local
exec_prefix=${prefix}
libdir=
includedir_old=${prefix}/include/opencv
includedir_new=${prefix}/include

Name: OpenCV
Description: Open Source Computer Vision Library
Version: 2.4.9
Libs: ${exec_prefix}/lib/libopencv_calib3d.so ${exec_prefix}/lib/libopencv_contrib.so $
{exec_prefix}/lib/libopencv_core.so ${exec_prefix}/lib/libopencv_features2d.so $
{exec_prefix}/lib/libopencv_flann.so ${exec_prefix}/lib/libopencv_gpu.so $
{exec_prefix}/lib/libopencv_highgui.so ${exec_prefix}/lib/libopencv_imgproc.so $
{exec_prefix}/lib/libopencv_legacy.so ${exec_prefix}/lib/libopencv_ml.so $
{exec_prefix}/lib/libopencv_nonfree.so ${exec_prefix}/lib/libopencv_objdetect.so $
{exec_prefix}/lib/libopencv_ocl.so ${exec_prefix}/lib/libopencv_photo.so $
{exec_prefix}/lib/libopencv_stitching.so ${exec_prefix}/lib/libopencv_superres.so $
{exec_prefix}/lib/libopencv_ts.a ${exec_prefix}/lib/libopencv_video.so $
{exec_prefix}/lib/libopencv_videostab.so -lrt -lpthread -lm -ldl
Cflags: -I${includedir_old} -I${includedir_new}
$ **pkg-config --list-all**



Editors (non-IDE)

- Emacs
- Vi
- Nano
- Gedit

nano

- Simplest console based editor
- Launched as:
  - nano filename
- Ctrl-O to save the file.
- Ctrl-X to exit.

Java Development Kit

- ~~Sun~~ Oracle Java is now Open Source!
- Omega server already has Java Dev Kit 1.6
- You can install JDK7 on your Ubuntu VM by:

  sudo apt-get install openjdk-7-jdk
- On a Windows/Mac machine you can download an installer from the Oracle JDK site

Java compiler

- javac ClassName.java
- Default output: ClassName.class
- To run:

  java ClassName

Outline

- GNU Compiler Collection
  - Suite of compilers – C, C++, Fortran,Java..
- **GNU make**
  - **Utility to automatically execute build steps**
- GNU debugger
- C++ and Java IDEs for Linux, Mac, and Windows.

GNU make

- Compile a project with multiple source files:

  g++ file1.cpp file2.cpp -o result

  - Slow - all files get recompiled every time!
- Single file compilation:

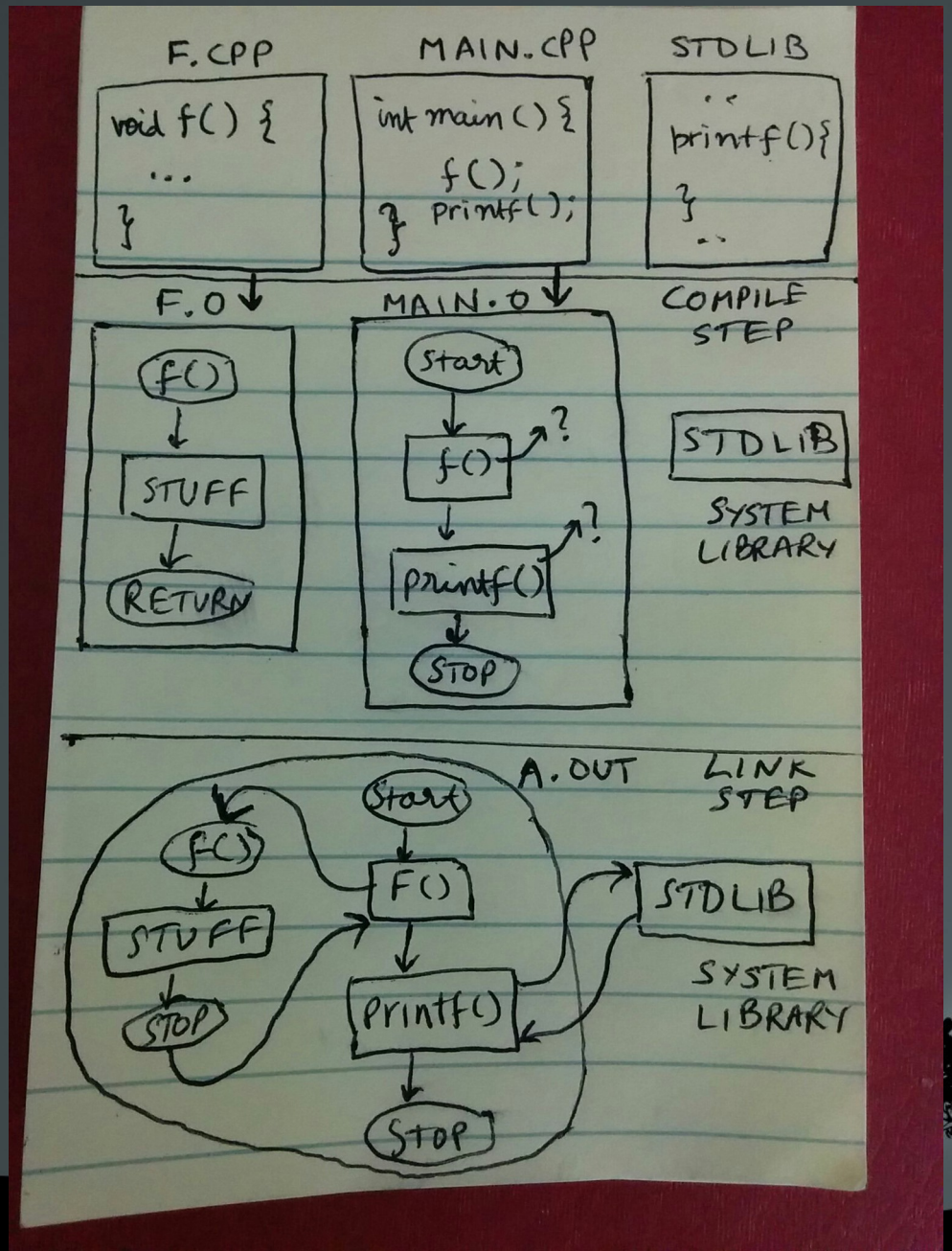
  g++ -c file1.cpp (produces file1.o)

  g++ -c file2.cpp (produces file2.o)

  g++ file1.o file2.o -o result (links file1.o and file2.o into result)

  - Only compile changed files and linker step – still cumbersome!
- 'make' utility – looks for a file 'Makefile' for build instructions
- Automatically detects changed files and rebuilds only those files

Compiling and Linking



Makefile

- Format:

```
target: prerequisites
[tab] commands
```

The tab on the second line is mandatory

- Default target is the first one in Makefile
- You can always specify one by name

```
file1.o: file1.cpp
    g++ -c file1.cpp
```

  make targetname

  - Will search for and build the target 'counter'
- 'clean' is a commonly used target to delete build results and temporaries
- 'install', 'check' are also commonly used target names

Simple makefile

```
hello: helloworld.cpp
    g++ -o hello helloworld.cpp
```

- Target file is 'hello'
- Depends on 'helloworld.cpp'
- The command to produce the target is:

    g++ -o hello helloworld.cpp

Makefile rules

CC=g++ *the default compiler to use*

CFLAGS=-I/include/dir *compiler flags*

LDFLAGS=-lm *linker flags*

$(CC) – using value of CC

```
$(CC) $(CPPFLAGS) file1.o file2.o $(LDFLAGS) -o result
```

.PHONY – defines targets with no output file

Makefile with multiple sources

```
CC=g++
#CPPFLAGS=-g
CPPFLAGS=-O3
INCLUDES=-I/home/rohit/libs/Eigen
LDFLAGS=-lm

result: file1.o file2.o
    $(CC) $(CPPFLAGS) file1.o file2.o $(LDFLAGS) -o result
file1.o: file1.cpp
    $(CC) -c $(CPPFLAGS) $(INCLUDES) file1.cpp
file2.o: file2.cpp
    $(CC) -c $(CPPFLAGS) $(INCLUDES) file2.cpp
.PHONY: clean
clean:
    rm file1.o file2.o result
```

Outline

- GNU Compiler Collection
  - Suite of compilers – C, C++, Fortran,Java..
- GNU make
  - Utility to automatically execute build steps
- **GNU debugger**
- C++ and Java IDEs for Linux, Mac, and Windows.

GNU Debugger - gdb

- Enable debugging information with -g during compilation and linking
- Try not to use optimization flag -O#
- Usage:
  - gdb program_name
- Allows: program listing, execution, stepping, breakpoints etc.

Sample gdb session

gdb hellomake

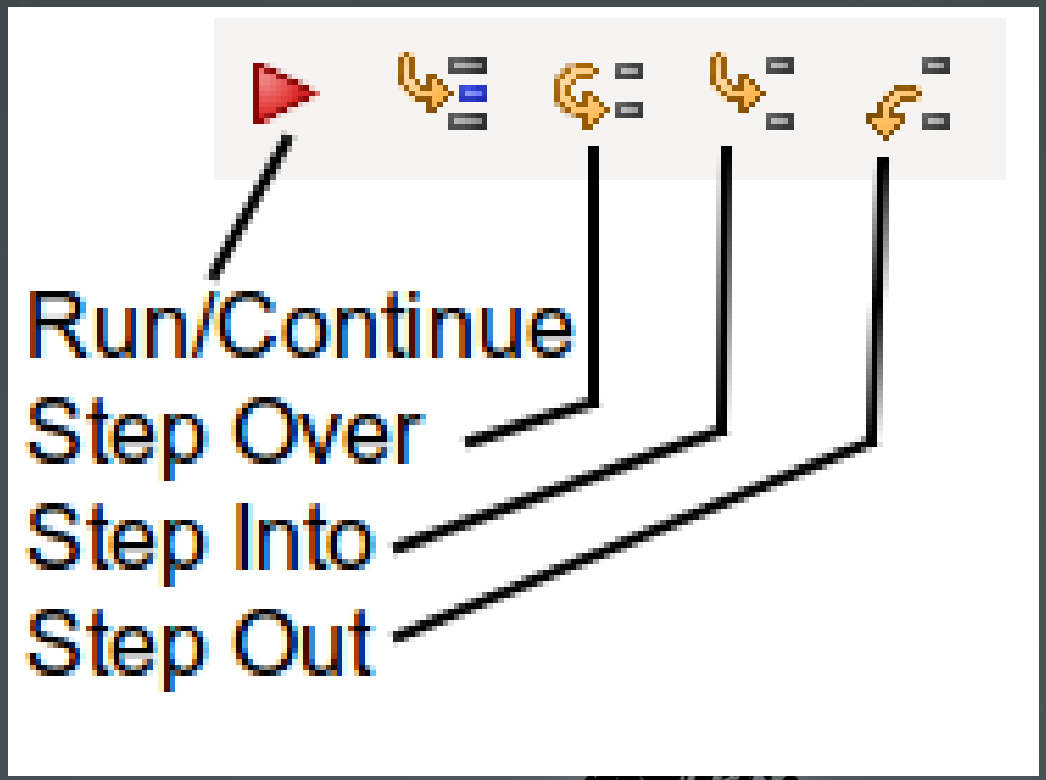
```
(gdb) list
1     #include <hellomake.h>
2
3     int main() {
4         // call a function in another file
5         myPrintHelloMake();
6         return(0);
7     }
8
(gdb) list myPrintHelloMake
1     #include <hellomake.h>
2     #include <stdio.h>
3
4     void myPrintHelloMake(void) {
5         printf("Hello makefiles!\n");
6         printf("Bye!\n");
7         return;
8     }
```

```
(gdb) run
Starting program: /home/rohit/LUG_Fall2013/prog/hellomake
warning: no loadable sections found in added symbol-file system-supplied DSO at 0x7ffff7ffa000
Hello makefiles!
Bye!
[Inferior 1 (process 13780) exited normally]
(gdb) break 6
Breakpoint 1 at 0x400540: file hellofunc.c, line 6.
(gdb) run
Starting program: /home/rohit/LUG_Fall2013/prog/hellomake
warning: no loadable sections found in added symbol-file system-supplied DSO at 0x7ffff7ffa000
Hello makefiles!
Breakpoint 1, myPrintHelloMake () at hellofunc.c:6
6         printf("Bye!\n");
(gdb) continue
Continuing.
Bye!
[Inferior 1 (process 13803) exited normally]
(gdb) info breakpoints
Num     Type           Disp Enb Address            What
1       breakpoint     keep y   0x0000000000400540 in myPrintHelloMake
                                                   at hellofunc.c:6
        breakpoint already hit 1 time
(gdb) quit
```

Other execution control

- Press **Ctrl-C** to break immediately
- **next** (step over)
- **step** (step into)
- **finish** (step out)

- **print x**
- **set x = 10**



Segmentation faults

```
(gdb) list
1     int func()
2     {
3             int *x;
4             x[99] = 100;  // this will cause a segfault because x has not been allocated memory
5             return 0;
6     }
7
8     int main()
9     {
10            func();
(gdb) run
Starting program: /home/rohit/LUG_Fall2013/prog/segfault
warning: no loadable sections found in added symbol-file system-supplied DSO at 0x7ffff7ffa00

Program received signal SIGSEGV, Segmentation fault.
0x0000000004004fa in func () at segfault.c:4
4             x[99] = 100;
(gdb) backtrace
#0  0x0000000004004fa in func () at segfault.c:4
#1  0x0000000000400515 in main () at segfault.c:10
```

References:

- GNU toolchain on Wikipedia: http://en.wikipedia.org/wiki/GNU_toolchain
- Nano editor: http://mintaka.sdsu.edu/reu/nano.html
- Makefiles: http://www.cs.swarthmore.edu/~newhall/unixhelp/howto_makefiles.html
- GDB: http://www.unknownroad.com/rtfm/gdbtut/gdbtoc.html