Software Construction Laboratory

Week 3 Part 2

Lab 6

Mushi Zhou

Winter 2017

UCLA

Assignment 10

- Research in recent computer science developments
- Assignment Page
- Read a recent news/research development article
- Search for related work
- Write a review 400 to 1200 words to cover the main idea and give your reaction to it, focusing on possible applications
- Present the topic by summarizing the meat of the topic

Presentation

- Sign up for a presentation time
- 3 presentations per lab starting Monday Week 5
- Google Signup Form (You need to use your UCLA google account)
- Deadline next Wednesday Feb, 1st
- First come first serve
- Come up with a topic, put your name, topic and link of the article
- No duplicate topics (Look at others' topics first)
- I'll highlight your entries once your topic is approved
- Check back for comments to modify if not approved

Presentation Guidelines

- Presentations should be 7-10 minutes
- Remember to include references (also for the report)
- Need to use PPT
- 3-10 slides are fine
- Don’t put too much text on each slides
- Graphical illustrations are recommended
- Be technical, but not too much (Comprehensible for your peers)
- Make it interesting
- Prepare for questions at the end

Submissions

- Presentations are from week 5 to week 9 at beginning of each lab
- If you can’t make it, email me a head of time
- Both report and presentation slides are due a week after your presentations (i.e. Midnight of Monday and Wednesday on CCLE)
- Grading scripts (for both presentation and report) are uploaded to the CCLE for Lab3 folder
- Materials in presentations could appear on your final exam

Outline for Today

- Make
- Automake and Autoconf
- Cmake
- Patch

Make

- Make is a build automation tool that automatically builds executable programs and libraries from source code by reading files called Makefiles which specify how to derive the target program
- Makefile is a script that contains multiple rules to process, compile, assemble, and link object files
- Basic tool
- Widely used for C/C++ programs

Questions

- What happens when we issue make?
- How does make know what to build?
- Why do we want to use make?

Sample Makefile

- # the compiler: gcc for C program, define as g++ for C++
- **CC = gcc**

- # compiler flags:
- # -g   adds debugging information to the executable file
- # -Wall turns on most, but not all, compiler warnings
- **CFLAGS = -g -Wall**

- # the build target executable:
- **TARGET = myprog**

- **all: $(TARGET)**

- **$(TARGET): $(TARGET).c**
- **	$(CC) $(CFLAGS) -o $(TARGET) $(TARGET).c**

- **clean:**
- **	$(RM) $(TARGET)**

Basics of Makefile

- Should be named Makefile/makefile
- Syntax: *target: dependencies*

  [tab] system command
- make -> make all (by default)
- Target is build only if all dependencies exists
- Fail if cannot build/find dependencies
- Usually contains a clean command
- Can define macros for clarity
- We will learn how to write Makefile in week 8

Tools to Automate Compilations

- **Automake** is a programming tool to automate parts of the compilation process

- It automatically generates one or more *Makefile.in* from files that contains useful information for the compiled software, such as compiler and linker flags, dependencies and versions, etc

- The generated *Makefile.ins* are used by configure scripts to generate a working Makefile

Tools to Automate Compilations

- **Autoconf** is a tool for producing configure scripts for building, installing and packaging software
- User can use the generated configure script to generate a makefile
- This is how production software was distributed in the past
- Together, Automake and Libtool, and Autoconf form the GNU build system

- More Info http://mij.oltrelinux.com/devel/autoconf-automake/

Cmake

- CMake is cross-platform, free, and open-source software for managing the build process of software using a compiler-independent method
- Supports directory hierarchies and applications that depend on multiple libraries
- Compatible with Linux, Windows, Mac OX, etc.
- It has minimal dependencies, requiring only a C++ compiler on its own build system
- It is not a command tool, but an integrated software what meant to be user friendly and automate the entire compilation and building process

Simple Example of Cmake

```
cmake_minimum_required(VERSION 2.8.9)
project (hello)
add_executable(hello helloworld.cpp)
```

- Stored in .txt files in each hierarchy
- Many powerful functions instead of commands
- Automates the entire compilation process
- cmake to invoke the script to generate Makefiles

Patch

- Updates source files according to instructions contained in a separate file

- Question:

  Why do we want to use a patch instead of a new source file?

Patch Command

- Syntax:

  $ patch –p[num] < patchfile

- Patches can be undone, or reversed, with the '-R' option:

  $ patch -R < pathfile

- The without –p option, the file to be patched has to be in the current directory, otherwise use –p0 to allow patch to find the file to update using the entire path specified in the patchfile

Creating a Patch File

- $ diff -u [oldfile] [newfile] > patch.diff

For Example:

- $ diff -u hello.c hello-new.c > hello.patch

- patch < hello.patch

Sample Patch File

- --- hello.c 2014-10-07 18:17:49.000000000 +0530
- +++ hello_new.c 2014-10-07 18:17:54.000000000 +0530
- @@ -1,5 +1,6 @@
- #include <stdio.h>
-
- -int main() {
- +int main(int argc, char *argv[]) {
- printf("Hello World\n");
- + return 0;
- }

Some Help for Assignment 3

For the lab

- Read the README.txt from the package (./configure & make)

- Since you might not have permission to install, execute (./) ls in /src to see wrong format, and copy another directory to try the patch)

- Since a path for file to update is specified, you need to use –p0 option

Some Help for the Assignment 3

For HW

- Need to have correct $PATH to run python3
- Two ways to execute

  chmod +x comm.py & ./comm.py

  python comm.py
- First handle the sorted case
- Make the output exactly as comm

Some Help for the Assignment 3

- Support options -1, 2, 3 -u
- No need to support other options
- Handle options similar to readlines.py
- But do need to set support environmental variables -> set LC_COLLATE to ‘c’ (import locale) for correct sorting order
- Follow the clarification post on Piazza

Some Help for the Assignment 3

- Many ways to implement unsorted case
- An example approach for unsorted case:

1. Create three columns to output
2. Run through the first file, search each row in second file to determine duplicate or unique and put to the corresponding row in the correct output column
3. Remove the duplicate from second file, not replacing with empty line
4. Print all remaining lines in second file in middle column at the end