Week 3
Compiling and Makefiles

23 January 2019

CS 35L Lab 4

Jeremy Rotman

Announcements

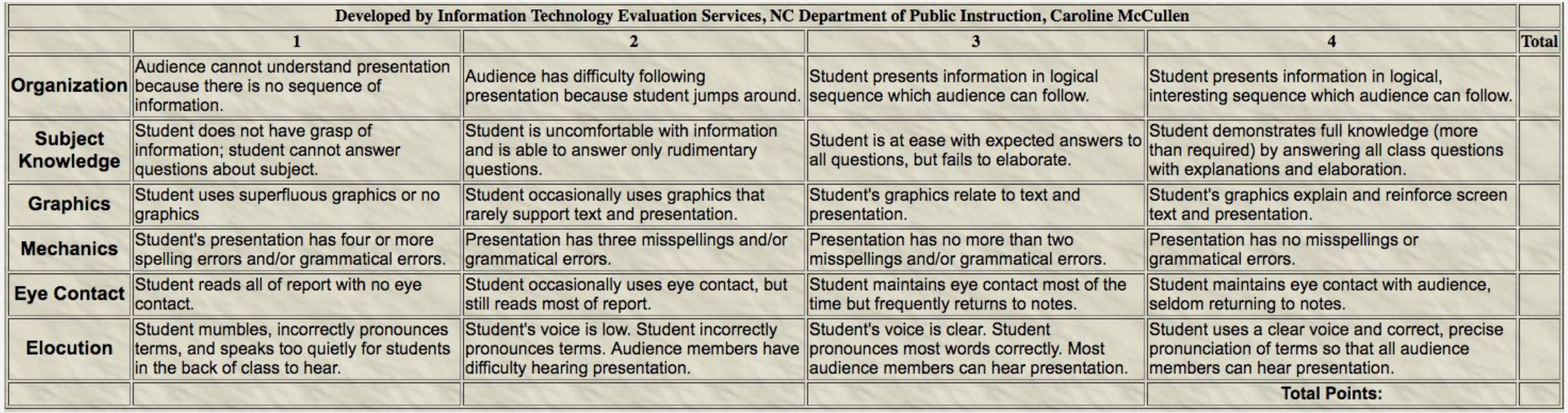
- Assignment #2 is TODAY by 11:55pm
- For Assignment #10
  - Feel free to begin choosing stories
  - Email me to tell me what you are choosing
- Reminder: Buy your BeagleBone!
  - Also, if you have a beagle, buy your beagle a bone, he/she probably deserves it

Questions?

Outline

- ➔ Assignment #10 info
- ➔ Interpreted and Compiled Languages
- ➔ Makefiles

Example Presentation Rubric



Developed by Information Technology Evaluation Services, NC Department of Public Instruction, Caroline McCullen					
	1	2	3	4	Total
Organization	Audience cannot understand presentation because there is no sequence of information.	Audience has difficulty following presentation because student jumps around.	Student presents information in logical sequence which audience can follow.	Student presents information in logical, interesting sequence which audience can follow.	
Subject Knowledge	Student does not have grasp of information; student cannot answer questions about subject.	Student is uncomfortable with information and is able to answer only rudimentary questions.	Student is at ease with expected answers to all questions, but fails to elaborate.	Student demonstrates full knowledge (more than required) by answering all class questions with explanations and elaboration.	
Graphics	Student uses superfluous graphics or no graphics	Student occasionally uses graphics that rarely support text and presentation.	Student's graphics relate to text and presentation.	Student's graphics explain and reinforce screen text and presentation.	
Mechanics	Student's presentation has four or more spelling errors and/or grammatical errors.	Presentation has three misspellings and/or grammatical errors.	Presentation has no more than two misspellings and/or grammatical errors.	Presentation has no misspellings or grammatical errors.	
Eye Contact	Student reads all of report with no eye contact.	Student occasionally uses eye contact, but still reads most of report.	Student maintains eye contact most of the time but frequently returns to notes.	Student maintains eye contact with audience, seldom returning to notes.	
Elocution	Student mumbles, incorrectly pronounces terms, and speaks too quietly for students in the back of class to hear.	Student's voice is low. Student incorrectly pronounces terms. Audience members have difficulty hearing presentation.	Student's voice is clear. Student pronounces most words correctly. Most audience members can hear presentation.	Student uses a clear voice and correct, precise pronunciation of terms so that all audience members can hear presentation.	
				Total Points:	

Presentation Specifics

- Presentation must be within 5-10 minutes
- Be prepared to answer questions from the audience
  - Students who ask questions may receive some extra points
- I am expecting you to be well prepared for the topic you are presenting on
  - I.e. just reading the article is not enough
- Here is the link to signup for a timeslot to present
  - Up to 3 presenters per day
  - Note: Read the date you sign up for, the sheet skips the holiday

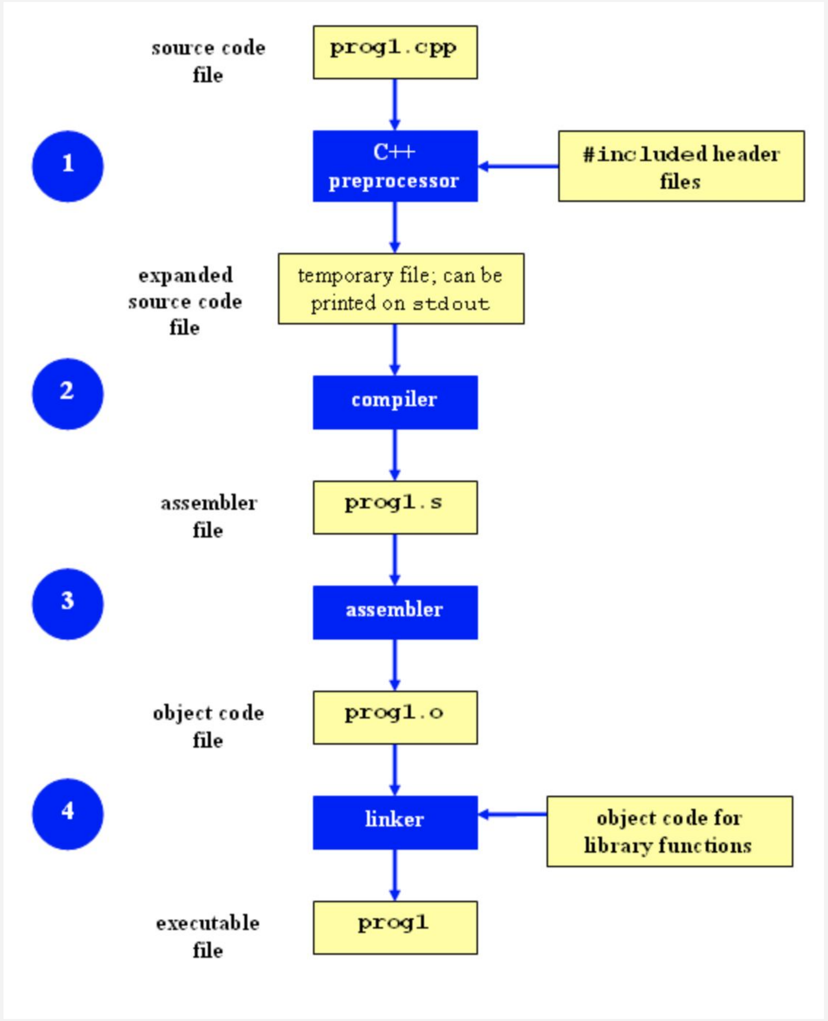
Compilation

- ➔ What does it mean when we compile code?

C++ compilation process

```
g++ -Wall -o prog1 prog1.cpp
```

1. C++ preprocessor copies header, generates macro, and checks defined constants
2. Source code compiled to assembly
3. Assembler code is assembled into object code
4. Object code is linked with object code files for library functions, producing an executable



Compilation

- ➔ What does it mean when we compile code?
  - ◆ Essentially translating code that we can read to code that a computer can quickly read and execute
- ➔ So what do other languages do?

Interpreters

- ➔ How does an interpreter differ from a compiler?

Interpreters

- ➔ How does an interpreter differ from a compiler?
  - ◆ Rather than one compile step to translate the entire source, an interpreter reads the source and translates line by line
- ➔ Advantages and Disadvantages

Interpreters

- ➔ How does an interpreter differ from a compiler?
  - ◆ Rather than one compile step to translate the entire source, an interpreter reads the source and translates line by line
- ➔ Advantages and Disadvantages
  - ◆ Programs run slower
    - ● But you can usually run them sooner
  - ◆ Simpler, and probably more intuitive
  - ◆ More portable
  - ◆ But it really is much slower
    - ● There is a lot of overhead
    - ● Added to extra time to actually call variables

What do we use in the real world?

- C++ is clearly a compiled language
- Python is generally seen as more of an interpreted language
  - Though even Python is a bit of a hybrid
- Java is…both?
  - Java <u>compiles</u> code into Java Bytecode
  - Java Bytecode is <u>interpreted</u> by a Java Virtual Machine
  - You can argue it is compiled into a language that runs faster
    - But it also still requires software to interpret the compiled file

Back to compilers

What happens if we want to compile many files?

- ➔ bookstore.cpp
  - ◆ Includes book.h and coffee.h
- ➔ book.cpp
  - ◆ Includes book.h
- ➔ coffee.cpp
  - ◆ Includes coffee.h

Back to compilers

What happens if we want to compile many files?

- bookstore.cpp
  - Includes book.h and coffee.h
- book.cpp
  - Includes book.h
- coffee.cpp
  - Includes coffee.h

```
g++ -Wall coffee.cpp book.cpp bookstore.cpp -o BandN
```

Compiling

- What happens every time we change something?
  - We have to recompile everything?
  - Do we have to recompile things that weren’t affected?
  - How do we keep track of which files are dependent on other files?

Compiling

- What happens every time we change something?
  - We have to recompile everything?
  - Do we have to recompile things that weren’t affected?
  - How do we keep track of which files are dependent on other files?
- Make
  - Allows us to use a file (makefile) to manage compilation on large projects
  - Can efficiently compile so that it only compiles what needs to be recompiled, or compiled in the first place

Makefile

- ➔ The makefile is the file that defines the ways to compile many files
- ➔ Additionally, it can define default flags to be used and which files are dependent on others

Makefile Example

```
CC=g++
CFLAGS=-Wall -g
all: BandN
BandN: coffee.o book.o bookstore.o
	$(CC) $(CFLAGS) -o BandN coffee.o book.o bookstore.o
coffee.o: coffee.cpp coffee.h
	$(CC) $(CFLAGS) -c coffee.cpp
book.o: book.cpp book.h
	$(CC) $(CFLAGS) -c book.cpp
bookstore.o: bookstore.cpp book.h coffee.h
	$(CC) $(CFLAGS) -c bookstore.cpp
clean:
	rm -f coffee.o book.o bookstore.o BandN
```

Make commands

- ./configure --prefix=./
  - The configure script checks to make sure the machine is ready to install software
  - Checks for proper dependencies
  - Usually generates the Makefile from an unfinished file
- make
  - Requires a Makefile
  - Builds the software
- make install
  - Runs the install label in the Makefile
  - Copies the built software to the system’s directories

The beginnings of Assignment #3

- ➔ I don’t want to go too far into it since Assignment #2 is still the priority
- ➔ However, we will go over a few small things

Unpacking a tarball

- The most common compressed file in linux is a tarball (.tar) or (.tar.gz) if also gzipped
  - This is especially common for distributing software
- However, once downloaded, you need to decompress it
- `tar -xzvf` *`filename`*`.tar.gz`
  - -x to e<u>x</u>tract
  - -z for g<u>z</u>ip
  - -v for <u>v</u>erbose
  - -f for <u>f</u>ile

Packing a tarball

- In the case that you ever need to compress a file or directory, you can still use tar
- `tar -czvf` *`archivename`*`.tar.gz` *`path_to_file`*
  - -c for create
  - The rest mean the same things

Python2 vs Python3

- ➔ They are different!
  - ◆ There are a number of key differences between python2 and python3, so it is important that you use the python that your code is written for
    - (Hint: the python in the assignment is Python2)
- ➔ Why are they both used?
  - ◆ When Python3 was released, they attempted to fix things they saw as problems in Python2
  - ◆ This meant that many things that were already written would no longer work in Python3
  - ◆ As a result the transition has taken almost 10 years, and is still kind of a work in progress