CSC161 2010F *Imperative Problem Solving*

Class 30: Make, Revisited

Held: Monday, 25 October 2010

Summary: We consider more details of `make`, a useful tool for automating lots of stuff.

Related Pages:

- EBoard.
- Reading: *Managing Projects with GNU Make*, Chapters 1 and 2..

Notes:

- I hope you had a great break.
- Returned: HW5, Exam 1. Sorry for the delay.
- I got more responses for today's reading. Thanks. Some of you asked enough questions that I probably did not need everyone's.
- Reading for Tuesday: K&R 5.1 and 5.2.
- Are there questions on Assignment 6?

Overview:

- About Make.
- Your Questions.
- A Collaborative Makefile.

Make

- Purpose: Make it easier to build projects, particularly complex multi-part projects.
- Model: A collection of "targets"
  - Each target is something that we might want to build (or a placeholder to help us build stuff)
  - There are instructions for building each target.
- Typical targets:
  - `default`: The default thing or things to build (e.g., the application or library)
  - `test` or `check`: Instructions for testing the main thing. (Generally predicated on building `default` first.)
  - `install`: Install the things we've just built.
  - `clean`: Remove intermediate files (such as .o files).
  - `package`: Put everything together into a tarball.
- Variables: Making it easier. Four kinds:
  - Your variables
  - Standard variables you set
  - Variables from the standard rules

  - Automatic variables
- Your variables

  ```
  NAME = VALUES
  ```

- Standard variables you set
  - CFLAGS
  - LDFLAGS
  - LDLIBS
- Variables from the standard rules
  - `$(CC)`
  - `$(COMPILE.c)`
  - ...
- Automatic variables
  - `$@` - The target
  - `$<` - The first prereq
  - `$?` - Newer prereqs
  - `$^` - All prereqs
  - `$*` - The stem of the target

Your Questions

- Is make basically a new language we need to learn?

- I need some review of what .o files are for.
- I don't know what -I does, exactly.
- Generally, I don't understand most of this stuff about libraries. Why are they necessary?
- Can we talk a bit more about CFLAGS and similar things?

- What is a 'jar'?
- Is a lexer a program that is assumed to be built for the implementation of this program, or is it some automatic file that is generated, or what?
- What does flex do?
- What does this mean: "First, we setup our experiment by creating an empty yacc source file and registering with RCSusing ci (that is, we want a version-controlled yacc source file)"
- What is bison and what does it do?
- Does the suffix .y signify anything specific?
- What is an 'awk' filter?

- I still don't quite understand $(...)
- I noticed that the text used a "rm" command in the makefile - does this mean we can use any valid terminal command in our makefile rules?
- I'm a little bit confused about phony targets, though. Because they don't represent files, are phony targets kind of like functions?
- I was wondering if you might be able to show us how to use wildcards in our makefiles.

- How does this work?

```
prog: *.c
        $(CC) -o $@ $^
```

Exercise

- We’ll work together to build a nice Makefile for homework 5.