Class 12: Tools: Project Management with Make

Held: Thursday, March 4, 2010

Summary: While our primary focus today is supposedly `make`, we will consider a wide variety of topics.

Related Pages:
- EBoard.

Notes:
- Reading for Tuesday: BC 32 (Code in Motion).
- It looks like we won’t have time for everything I have planned today. Expect the Make stuff to turn into an assignment that we will discuss on Tuesday.
- EC for the StatsGames talk today.
- EC for CS Table tomorrow.

Overview:
- Use Cases: iSimGrinL.
- Beautiful Code: Bentley.
- Beautiful Code: Subversion’s Delta Trees.
- About Make.
- Hands-on Make.

Project: Use Cases
- You’ll narrate your use cases.
- I’ll discuss some concepts that I have.
- We’ll try to tie it all together.

Bentley’s “The Most Beautiful Code I Never Wrote”
- Let’s go over the steps in the analysis together.
- What is confusing?
- What is beautiful?
- What is ugly?
Subversion’s Delta Trees

- What is confusing?
- What is beautiful?
- What is ugly?

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

Hands on Make

Part one: Building a simple C program

- Create array-utils.{h,c}, which contain the initial parts of a library of array utilities. Your library need only include the procedure str_swap(char *A[], int i, int j). Sample code appears below.
- Create quicksort.{h,c}, which contain an implementation of the Quicksort routine. You library need only contain str_quicksort(char *A[], int lb, int ub). which sorts a subarray of an array of strings.
- Create qsort.c, which sorts the strings given on the command line (that is, argv)
- Create qstest.c, a simple unit test for str_quicksort. (You can use C’s assert procedure for writing your unit tests.)
- Write a Makefile that allows you to build the qsort application when the make command is invoked with no parameters.
- Extend your Makefile to build and run qstest when the make command is invoked with test.

Part two: Making a library

Ideally, the library code we write would eventually end up in a library that we could link.
Part three: A different task

It is often useful to convert html documents to more useful formats. Write a generic makefile for such conversion.

- Use html2ps to convert HTML files to Postscript.
- Use ps2pdf to convert PostScript files to PDF.

array-utils.h

```c
#ifndef __ARRAY_UTILS_H__
#define __ARRAY_UTILS_H__

/**
 * Swap the elements in positions i and j of A.
 * Pre:
 * 0 <= i, j < length of A
 * A[i] = I
 * A[j] = J
 * Post
 * A[i] = J
 * A[j] = I
 * For all x, x != i and x != j, A[x] is unchanged.
 */
void str_swap (char *A[], int i, int j);
#endif /* __ARRAY_UTILS_H__ */
```

array-utils.c

```c
#include "array-utils.h"
...
```
Commons, 543 Howard Street, 5th Floor, San Francisco, California, 94105, USA.