Build Managers

Steven J Zeil

February 21, 2013

Outline

Outline I

Build Managers

A *build manager* is a tool for scripting the automated steps required to produce a software artifact.

What Should a Build Manager Do?

A good build manager should be

- easy to use

What Should a Build Manager Do?

A good build manager should be

- easy to use
- easy to set up for a given project

What Should a Build Manager Do?

A good build manager should be

- easy to use
- easy to set up for a given project
- efficient in performing the build

What Should a Build Manager Do?

A good build manager should be

- easy to use
- easy to set up for a given project
- efficient in performing the build
    - avoid redundant/unnecessary actions

What Should a Build Manager Do?

A good build manager should be

- easy to use
- easy to set up for a given project
- efficient in performing the build
    - avoid redundant/unnecessary actions
    - detect and abort bad builds in progress

What Should a Build Manager Do?

A good build manager should be

- easy to use
- easy to set up for a given project
- efficient in performing the build
    - avoid redundant/unnecessary actions
    - detect and abort bad builds in progress
- incremental

What Should a Build Manager Do?

A good build manager should be

- easy to use
- easy to set up for a given project
- efficient in performing the build
    - avoid redundant/unnecessary actions
    - detect and abort bad builds in progress
- incremental
    - allow focused/partial builds

What Should a Build Manager Do?

A good build manager should be

- easy to use
- easy to set up for a given project
- efficient in performing the build
    - avoid redundant/unnecessary actions
    - detect and abort bad builds in progress
- incremental
    - allow focused/partial builds
- flexible

What Should a Build Manager Do?

A good build manager should be

- easy to use
- easy to set up for a given project
- efficient in performing the build
  - avoid redundant/unnecessary actions
  - detect and abort bad builds in progress
- incremental
  - allow focused/partial builds
- flexible
  - allow for a variety of build actions

What Should a Build Manager Do?

A good build manager should be

- easy to use
- easy to set up for a given project
- efficient in performing the build
    - avoid redundant/unnecessary actions
    - detect and abort bad builds in progress
- incremental
    - allow focused/partial builds
- flexible
    - allow for a variety of build actions
    - on a variety of platforms

What Should a Build Manager Do?

A good build manager should be

- easy to use
- easy to set up for a given project
- efficient in performing the build
    - avoid redundant/unnecessary actions
    - detect and abort bad builds in progress
- incremental
    - allow focused/partial builds
- flexible
    - allow for a variety of build actions
    - on a variety of platforms
- configurable

What Should a Build Manager Do?

A good build manager should be

- easy to use
- easy to set up for a given project
- efficient in performing the build
    - avoid redundant/unnecessary actions
    - detect and abort bad builds in progress
- incremental
    - allow focused/partial builds
- flexible
    - allow for a variety of build actions
    - on a variety of platforms
- configurable
    - permit the management of multiple artifact configurations

Outline I

Some Sample Project Builds

Here are some of the project builds I have had to automate in the opening weeks of this semester (Spring 2013).

Student Programming Assignment

Set up to allow students to easily compile code for an assignment.

- Build each missing or out-of-date .o file by compiling a corresponding .cpp file.
    - Record which .cpp files and .h files were used during the compilation so that future builds can determine what would future source code changes would make this .o file outdated.
- Link all .o files to produce an executable

Code Annotation Tool

The code annotation tool is a program I use to convert C++ and Java code with optional markup comments like this

```
#include <iostream>

using namespace std; /**co1*/

int main (int argc, char** argv)
{
  // Let's be friendly
  cout << "Hello world!" << endl;
  /*+*/return 0;/*-*/
}
```

Code Annotation Tool Output

. . . into this or this:

```
#include <iostream>

using namespace std; ❶

int main (int argc, char** argv)
{
  // Let's be friendly
  cout << "Hello world!" << endl;
  return 0;
}
```

Building the Code Annotation Tool

The steps involved in building this tool are:

1. Run the program **jflex** on each file in `src/main/jflex`, generating a pair of .java files that get placed in `src/main/java`
2. Compile the Java files in `src/main/java`, placing the results in `target/classes`
3. Compile the Java files in `src/test/java` (using the `target/classes` compilation results, placing the results in `target/test-classes`.
4. Run the JUnit tests in `target/test-classes`.
5. If all tests pass, package the compiled classes in `target/classes` into a .jar file.

Building the Code Annotation Tool

The steps involved in building this tool are:

1. Run the program **jflex** on each file in `src/main/jflex`, generating a pair of .java files that get placed in `src/main/java`
2. Compile the Java files in `src/main/java`, placing the results in `target/classes`
3. Compile the Java files in `src/test/java` (using the `target/classes` compilation results, placing the results in `target/test-classes`.
4. Run the JUnit tests in `target/test-classes`.
5. If all tests pass, package the compiled classes in `target/classes` into a .jar file.

It's worth noting how many of the steps in this project build are *not* simply compile and link steps.

Class Assignment Setup I

In preparing to release a programming assignment to a class, the steps are

1. Setup:
    1. Copy all of the files that I will provide to students from a `Public` directory into a `Work` directory.
    2. Copy all of the files from my `Solution` directory into that `Work` directory
2. Build solution
    1. Compile any .cpp files in the `Work` directory
    2. Link the resulting .o files.
3. Run the executable produced in the last step on each `test*.dat` in the `Tests` directory, capturing the output as a corresponding `.out` file.

Class Assignment Setup II

4. Copy all source code from the `Work` directory into a `winWork` directory.
5. Use a cross-compiler to compile and link the .cpp files in `winWork` into a Windows executable
6. Install:
   1. Copy the two executables and the contents of the `Public` directory into a release area accessible to students.
   2. Set the permissions on the copied files so that they can be accessed.
   3. Copy any .html and graphics files for the assignment to the course website.

Posting Slides and Lecture Notes I

The lectures notes for this course are prepared through a process:

1. Setup
    1. If the directory has a DocBook document and no corresponding TeX file, and if we are on a machine where **db2latex** is installed, run **db2latex** to create a TeX file.
    2. Convert all graphics to PNG or PDF:
        1. For each desired document format, copy a corresponding template into this directory, substituting for various course properties (e.g., course name, website URL), saving this as a `.ltx` file.
        2. For each GIF file in the directory with no corresponding PNG file, run **convert** to produce a PNG.
        3. For each FIG file in the directory with no corresponding EPS file, run **fig2dev** to produce an EPS.
        4. For each Dia file in the directory with no corresponding EPS file, run **dia** to export as EPS.

Posting Slides and Lecture Notes II

    5. For each EPS file in the directory with no corresponding PDF file, run **epstopdf** to create a PDF.
  3. Annotate source code:
    1. For each C++ or Java file with no corresponding HTML file, use the code annotation toll to generate an HTML file.
    2. For each C++ or Java file with no corresponding TeX file, use the code annotation toll to generate an TeX file.
2. For each desired document format, run **latexmk** to produce a PDF for that format.
3. Deployment:
    1. Synchronize this directory with the corresponding directory of the website, or
    2. Prepare a zip file with the contents of this directory that can be uploaded to a remote webserver (e.g., Blackboard).

Outline I

Why Not Just Write a Script?

We could simply write a simple script to perform each of the steps in sequence. . .

Scripting

But how does this fare according to our earlier build manager goals?

- easy to use?
- easy to set up for a given project?
- efficient in performing the build?
    - avoid redundant/unnecessary actions
    - detect and abort bad builds in progress
- incremental?
    - allow focused/partial builds
- flexible?
    - allow for a variety of build actions
    - on a variety of platforms
- configurable?
    - permit the management of multiple artifact configurations

Scripting

But how does this fare according to our earlier build manager goals?

- easy to use? ✓
- easy to set up for a given project? ✗
- efficient in performing the build?
  - avoid redundant/unnecessary actions
  - detect and abort bad builds in progress
- incremental?
  - allow focused/partial builds
- flexible?
  - allow for a variety of build actions
  - on a variety of platforms
- configurable?
  - permit the management of multiple artifact configurations

Scripting

But how does this fare according to our earlier build manager goals?

- easy to use? ✓
- easy to set up for a given project? ✗
- efficient in performing the build?
  - avoid redundant/unnecessary actions ✗
  - detect and abort bad builds in progress ?
- incremental?
  - allow focused/partial builds
- flexible?
  - allow for a variety of build actions
  - on a variety of platforms
- configurable?
  - permit the management of multiple artifact configurations

Scripting

But how does this fare according to our earlier build manager goals?

- easy to use? ✓
- easy to set up for a given project? ✗
- efficient in performing the build?
  - avoid redundant/unnecessary actions ✗
  - detect and abort bad builds in progress ?
- incremental?
  - allow focused/partial builds ?
- flexible?
  - allow for a variety of build actions
  - on a variety of platforms
- configurable?
  - permit the management of multiple artifact configurations

Scripting

But how does this fare according to our earlier build manager goals?

- easy to use? ✓
- easy to set up for a given project? ✗
- efficient in performing the build?
  - avoid redundant/unnecessary actions ✗
  - detect and abort bad builds in progress ?
- incremental?
  - allow focused/partial builds ?
- flexible?
  - allow for a variety of build actions ✗
  - on a variety of platforms ✗
- configurable?
  - permit the management of multiple artifact configurations

Scripting

But how does this fare according to our earlier build manager goals?

- easy to use? ✓
- easy to set up for a given project? ✗
- efficient in performing the build?
  - avoid redundant/unnecessary actions ✗
  - detect and abort bad builds in progress ?
- incremental?
  - allow focused/partial builds ?
- flexible?
  - allow for a variety of build actions ✗
  - on a variety of platforms ✗
- configurable?
  - permit the management of multiple artifact configurations ?

IDE project managers

Most IDEs come with a built-in project manager.

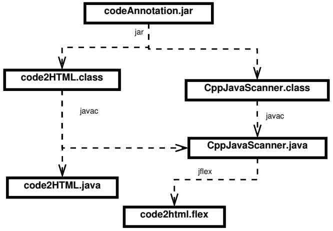
- typically limited to compoling and linking
- maybe some support for packaging

Compare to our sample projects.

Dependency-Based Managers

Some build managers are based on the idea of a *dependency graph*:

- Boxes are files.
- Arrows denote dependencies. "A depends on B" means that if B is missing or changed, then A must be (re)generated.
- Labels on arrows indicate the program used to generate the file at the base of the arrow.



Analysis of such a graph facilitates

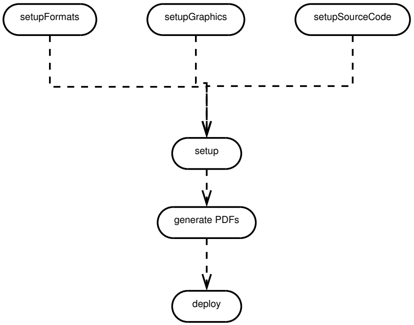
- efficiency - easy to tell what needs to be rebuilt after a change
- incrementality - can determine required build step for any file, not just the "final" one

make is the canonical example of a build manager of this type.

Task-Based Managers

Other managers are based on the idea of interdependent tasks.

- Ellipses are tasks (activities). Each task can involve multiple steps.
- Arrows denote success dependencies. "A depends on B" means that A will be run after B and only if task B finished successfully.



This approach facilitates

- easy to set up: usually less detailed than a full file-based dependency graph
- incrementality - can request any intermediate step