Grand Valley State University
www.gvsu.edu

Software Engineering
System Building

Erik Fredericks // frederer@gvsu.edu

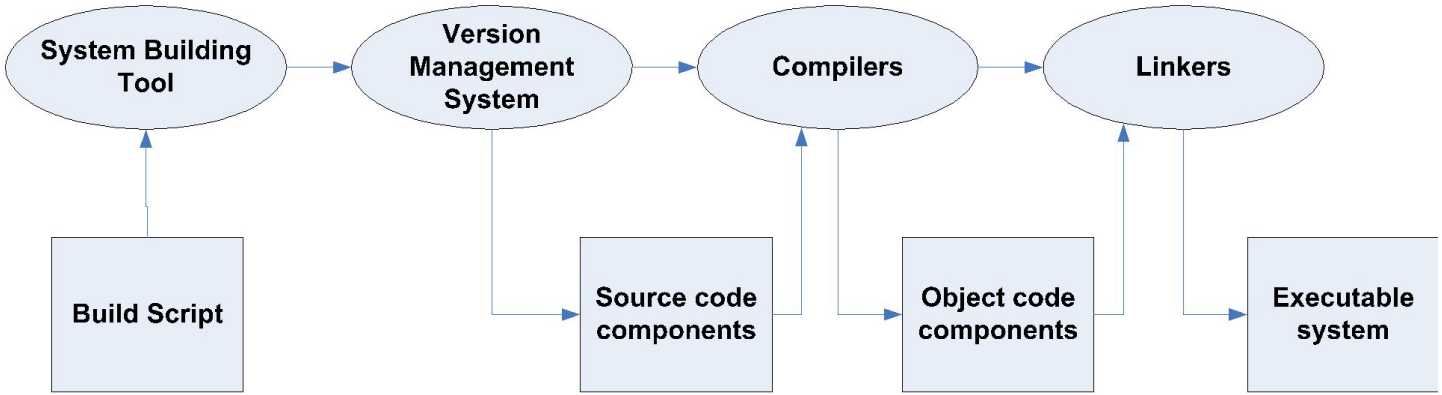
Adapted from materials provided by Byron DeVries, Jagadeesh Nandigam

Outline

What is System Building?
System Building Tools
Apache ANT
Built-in Tasks in Ant

What … *is* … system building?

System building is the process of assembling software components into a program that executes on a particular target configuration.



System building tools

A number of system building automation tools are available:

- make, gnumake, nmake
- Apache Ant, Nant
- Apache Maven
- Gradle
- MSBuild
- Rake
- CruiseControl

...

What is 🐜?

Another **N**eat **T**ool

Like `make` without `make`'s wrinkles.

Open source:

- Apache Ant project: http://ant.apache.org

Implemented in Java

Used to build many open source projects, including:

- Tomcat,
- Axis,
- JDOM,
- ...

Ant vs. make

Ant is more portable

Ant only requires a Java VM

make relies on OS specific commands

Ant scripts are XML-based

- make scripts are cryptic
- make relies on proper use of tabs in scripts

Ant is better for some Java-specific tasks

How does Ant work?

Each **project** using Ant will have a build file:

- Typically called build.xml

A build file is composed of targets

- Targets correspond to common activities like:
  - Compiling
  - Running
  - Deploying

Each target is composed of tasks

- Tasks are commands that are executed when the target is executed
- Ant scripts use <property> elements to define global name/value pairs available for use in the script.

Ant tasks

Ant tasks are implemented by Java classes

- Many useful tasks are built-in
- Other tasks come in optional jar files
- Custom tasks can always be created
    - Ant is an extensible tool

Simple Ant

```
<?xml version="1.0"?>
<project name="test" default="compile" basedir=".">

    <property name="src" value="."/>
    <property name="build" value="build"/>

    <target name="init">
        <mkdir dir="${build}"/>
    </target>

    <target name="compile" depends="init">
        <javac srcdir="${src}" destdir="${build}"/>
    </target>
</project>
```

Running Ant script

`ant -h`:	command prints help page
`ant –p`:	prints the targets and the default target (if defined) for the current project
`ant [target]`:	runs the ant script with the target
`ant`:	runs ant with default target

Built-in tasks in Ant

Archive Tasks
Audit/Coverage Tasks
Compile Tasks
Deployment Tasks
Documentation Tasks
EJB Tasks
Execution Tasks
File Tasks
Java2 Extensions Tasks
Logging Tasks
Mail Tasks
Miscellaneous Tasks
Pre-process Tasks
Property Tasks
Remote Tasks
SCM Tasks
Testing Tasks

What other build systems are you familiar with?

gradle ex (building for C++) (in WSL2) (((~˘▾˘)~))

Install gradle

```
$ gradle init
$ ./gradlew build
$ ./app/build/exe/main/debug/app
$ ./gradlew build --scan
```

https://docs.gradle.org/current/samples/sample_building_cpp_applications.html

cmake example

Note: I am _not_ good with cmake

https://github.com/ttroy50/cmake-examples/

Typical:

```
$ mkdir build
$ cd build
$ cmake ..
$ make
```

http://www.gimpel-online.com/OnlineTesting.html

In-Class Work

Describe:

1) How would you "package" your term projects (i.e., to deploy)

2) How would you "sell" your term projects (i.e., convince others to use/integrate/etc.)