COMP 2400 UNIX Tools

Christian Grothoff
christian@grothoff.org

http://grothoff.org/christian/

When to ship?

Do **not** assume that you can produce a bug-free product, rather:

- Decide which bugs are acceptable and which ones are not
- Ship with known bugs rather than untested bugfixes!

Bug Assessment

- How severe is the bug?
- How frequent does it happen?
- How much does it cost to fix it?
- How risky is fixing it (in terms of introducing new problems)

Testing!

- Unit testing: do individual components work?
- Integration testing: do components work together?
- Functional / system testing: software does what it is supposed to do?
- Acceptance / beta testing: "real" users happy?

Get ready!

- Mailinglist?
- Webpage, on-line discussion forum?
- End-user documentation?
- Bugtracking system?

What to ship?

- Consider target audience
- Consider software license(s)
- Consider dependencies (include Java libraries? Java VM? Entire operating system?)
- Always include a version number

Build System Considerations

- Dependency management (internal and external)
- Portability (build for Win32, Linux, OS X, etc.)
- Configurability (build with GUI, without sound, etc.)
- Usability (for developer, end-user)
- Diagnostics

Free Build Systems for Java

- jar
- izPack
- autotools

`jar` overview

- Commonly used to create archive with `class` files
- Command options similar to `tar`, but produces `zip` file
- `jar` files can include arbitrary files
- `jar` files can contain metadata about the contents

$ java -jar myprogram.jar # if metadata specifies main class

Using jar

$ jar -cvf myprogram.jar ‘find * -name ’*.class‘

$ jar -cvfm myprogram.jar manifest *.class image.png

```
ClassLoader cl = getClass().getClassLoader();
URL url = cl.getResource("image.png");
InputStream is = url.openConnection()
                    .getInputStream();
```

Manifest Example

```
Manifest-Version: 1.0
Main-Class: edu.edu.application.Test
Sealed: true
```

Use `keytool` and `jarsigner` to create cryptographically signed `jar` archives.

izPack overview

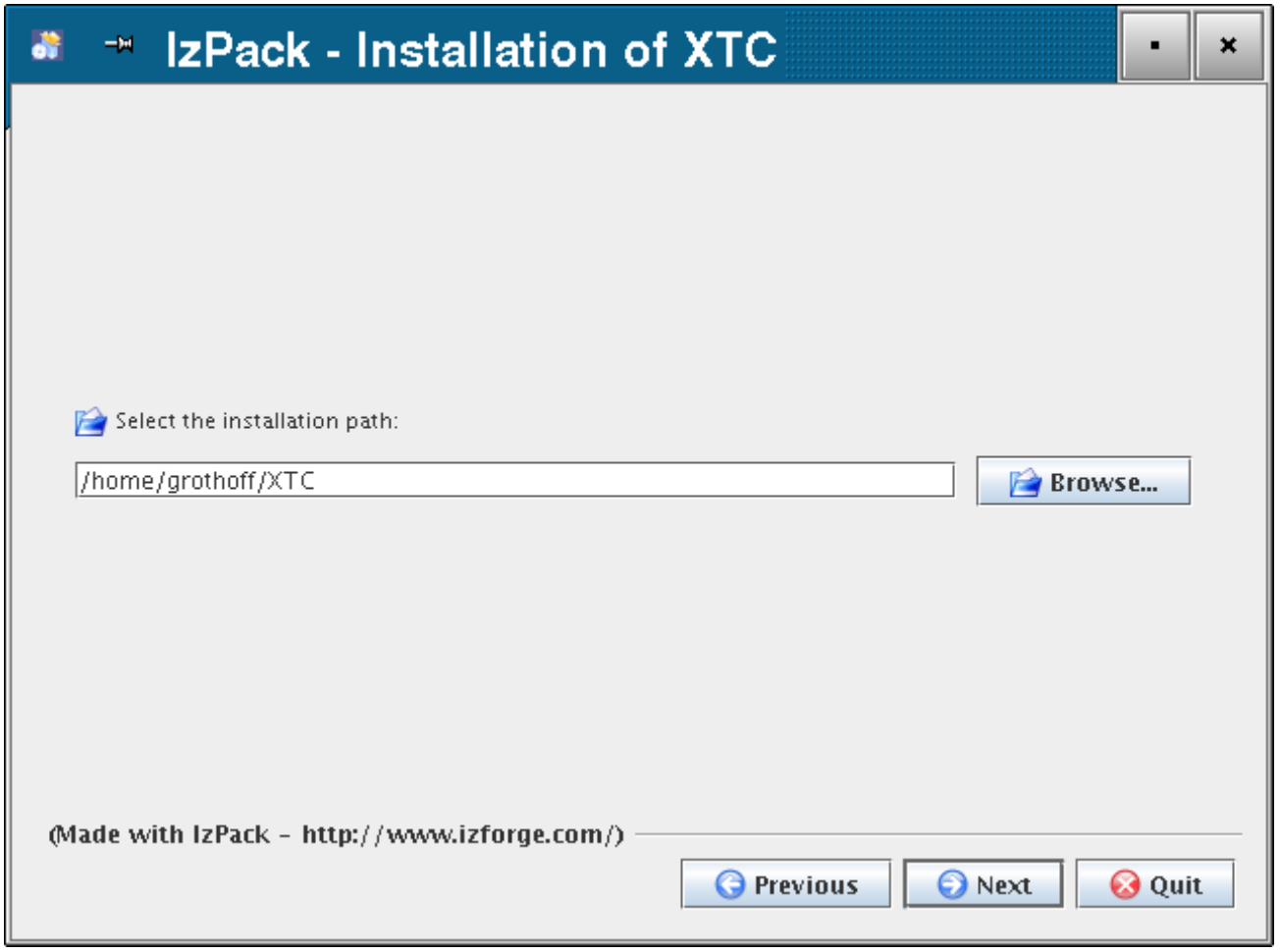
Inputs:

- `jar` file(s) with code from build system
- Configuration file describing installation

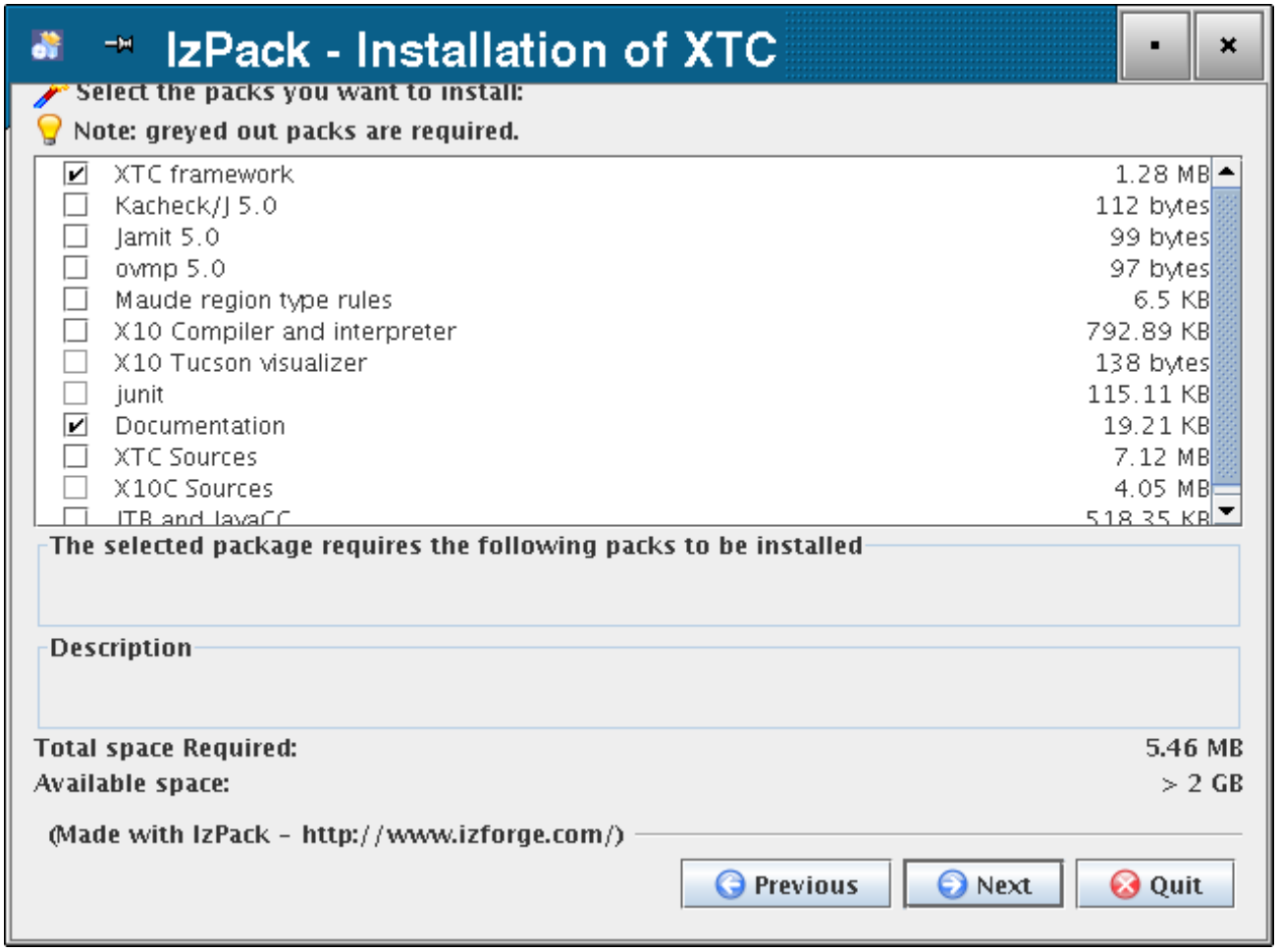
Output:

- `jar` file(s) with graphical installer

izPack graphical installer (1/2)



izPack graphical installer (2/2)



Configuration Example

```
<?xml version="1.0" encoding="iso-8859-1" standalone="yes"
<installation version="1.0">
<info>
 <appname>MyApplication</appname>
 <appversion>1.0</appversion>
 <authors>
  <author name="Christian" email="c@example.com"/>
 </authors>
 <url>http://myapp.org/</url>
 <javaversion>1.5</javaversion>
</info>
<guiprefs width="640" height="480" resizable="no"/>
```

Configuration Example

```
<locale><langpack iso3="eng"/></locale>
<resources>
 <res id="LicencePanel.licence" src="COPYING"/>
 <res id="InfoPanel.info" src="README"/>
</resources>
<variables>
 <variable name="JDKPathPanel.minVersion" value="1.5" />
 <variable name="JDKPathPanel.maxVersion" value="1.5.99" /
 <variable name="JDKPathPanel.skipIfValid" value="yes" />
</variables>
```

Configuration Example

```
<panels>
 <panel classname="HelloPanel"/>
 <panel classname="InfoPanel"/>
 <panel classname="LicencePanel"/>
 <panel classname="TargetPanel"/>
 <panel classname="JDKPathPanel"/>
 <panel classname="PacksPanel"/>
 <panel classname="InstallPanel"/>
 <panel classname="FinishPanel"/>
</panels>
```

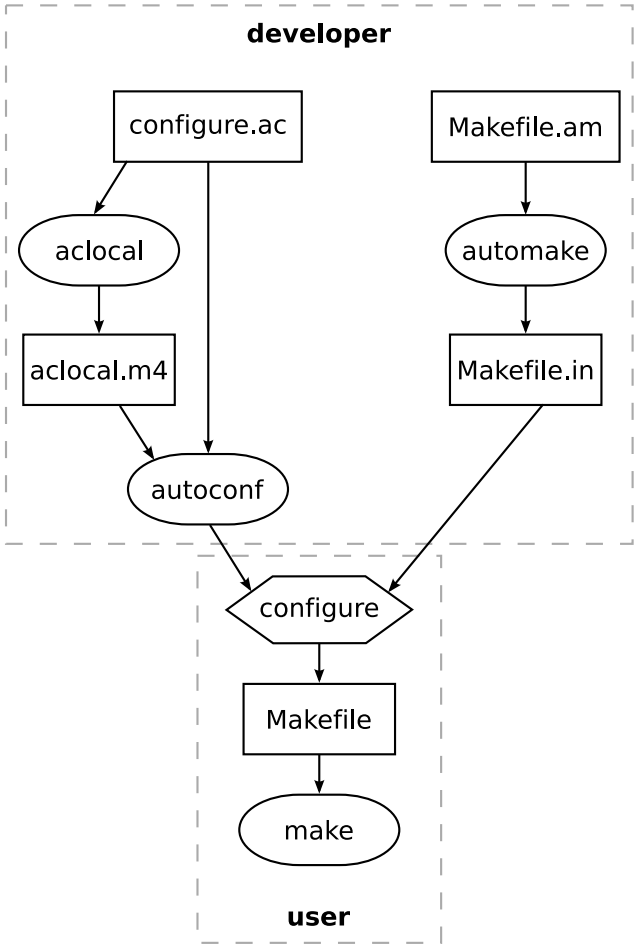
Configuration Example

```
<packs>
 <pack name="Main application" preselected="yes"
                               required="yes">
   <description>Cool program!</description>
   <file src="app.jar" targetdir="$INSTALL_PATH/lib"/>
 </pack>
</packs>
</installation>
```

autotools

- Intended for building and distributing C and C++ source code
- Can be made to work with other languages
- Generates shell skript(s) which generate Makefiles
- Uses shell scripting, M4 macro processor and make
- Main commands: make, m4, automake, autoconf

autotools


developer
configure.ac
Makefile.am
aclocal
automake
aclocal.m4
Makefile.in
autoconf
configure
Makefile
make
user

autotools for Java

- Packages source code for distribution
- Detects presence of JVM and Java compiler
- Compiles Java code, creates and installs `jar` archive
- Can be combined with compilation of C code (useful for JNI)
- Sample code is at https://gnunet.org/svn/Extractor-java/

Questions

?

Task

For each of your projects, use `jar` to create an archive with the Java classes of your main application. Create a second archive with the test suite.

Use `izPack` to create an installer, including selection of installation path, accepting the license and selection of features to install (main application, testcases).

Have izPack generate executable shell scripts to start the application and test suites.