2.2 Packaging, Make, Distribution

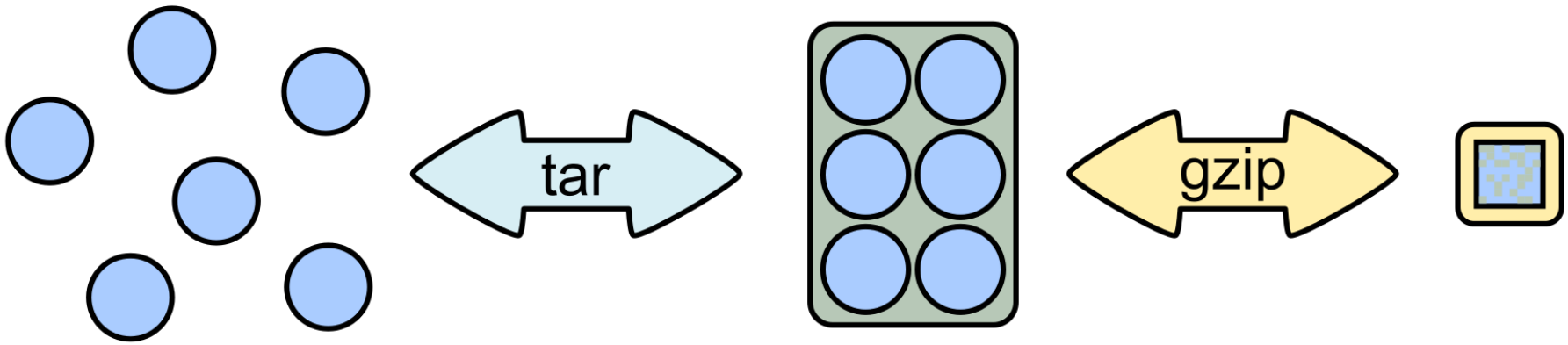
Uwe Altermann

30. April 2013

Content

What is a tarball/tar.gz package?

- "tar" name of packaging program and the packages created with
- Tar files are collections of many files for distribution or archiving
- For distribution often additionally compressed with gzip
- **pack:** tar cvf dist.tar <file-list >or <directory >→ dist.tar
  compress: gzip dist.tar → dist.tar.gz



(http://en.wikipedia.org/wiki/File:Targzip.svg)

What does architecture dependent/indipendent mean?

- Architecture dependent mean, a program that can only run on a specific architecture (after compiling)
- Architecture indipendent mean, a programm that can run on many architectures (depending on used compiler)
  - main() { printf(“hello world”); }
    is architecture indipendent, it can be compiled for each architecture

What does make do?

- Utility used to build or recompile executable programs and/or libraries from source code
- Recompiles only modified files $\rightarrow$ avoids unnecessary builds
- Architecture independent $\rightarrow$ can by used for each programming language whose compiler can be run in a shell
- Depends on a specification file called “makefile”

makefile

- Describes all relationships between files of a program
- Contents: explicit-/implicit rules, directives, variables, comments, macros
  - **rules:** describes targets and its dependencies (files, targets)
    gives a recipes to create or update a target
    Syntax: targets : prerequisites
    <tap!> recipe
    <tap!> ...
    Example prog: foo.c defs.h
    <tap!>gcc -o prog foo.c
  - **directives:** instructions for make to do something special while reading the makefile
    - ★ reading another makefile
    - ★ deciding whether to use or ignore parts of the makefile
  - **variables:** specification of a value to a variable used later in the file
  - **comments:** lines starting with #

Can “make” make use of multiple cores/CPUs for parallel builds?

- Yes
- GNU-make (Linux):
  - can be run with the -jN option
  - N specifies the number of jobs (recipes) executed at once
  - is N not given, there is are no limit of jobs
- CMake (architecture independent):
  - generates makefiles that can be used with different compiler environments
  - e.g. Linux: with GNU-make or ninja
    Mac OSX : the same as Linux
    Windows: GNU-make, ninja, nmake, jom, Visual Studio IDE...

What are standard make targets and what do they do?

- Make is invoked with a list of targets e.g. make install
- Is no target given, make uses the first target in the makefile
- Some standard targets:
  - all: compile the entire program (should be the default, first target)
  - install: compile the program and copy all necessary files to dir for use
  - uninstall: delete all installed files and copies
  - clean: delete all files in current dir created by building the program
  - info: generate the info files
  - dist: create a distribution tar file
  - installcheck: perform installation tests
- Many more standard targets available

What is a staged installation?

- A installation to a user defined directory
- Dependents on “DESTDIR” variable, which should...
  - ... be prepended to each installed target file
    - ★ e.g: binfoo $(DESTDIR)$(bindir)/binfoo
      libfoo $(DESTDIR)$(libdir)/libfoo.a
  - ... support only install and uninstall targets
  - ... be defined by the user on the make command line
    - ★ e.g: make DESTDIR=/tmp/stage install
      normally installed to...
      /usr/local/bin/binfoo
      /usr/local/lib/libfoo.a
      ...will be installed to...
      /tmp/stage/usr/local/bin/binfoo
      /tmp/stage/usr/local/lib/libfoo.a

How to prepare software for distribution? I

- C/C++:
  - Compile the programm to create a executable
    gcc -o prog source1.c, source2.c, ... or *.c
    g++ -o prog source1.cpp, source2.cpp, ... or *.cpp
  - Create a dir containing the executable and all dependencies
  - Create a package for distribution (.tar, *.zip,...)
- Java:
  - Compile the programm
    javac source1.java, source2.java, ... or *.java
  - Build jar-file with all java.class-files
    jar prog.jar source1.class, source2.class, ... or *.class
  - Create a dir containing the jar(s) and all dependencies
  - Create a package for distribution (.tar, *.zip,...)
- In huge project it is better to use “make” for compiling

How to prepare software for distribution? II

- Perl:
  - Module::Build or ExtUtils::MakeMaker can be used
  - Module::Build:
    - ★ System for building, testing and installing perl modules
    - ★ A Build.PL file has to be created
    - ★ perl Build.PL
      ./Build
      ./Build test
      ./Build install
      ./Build dist
  - ExtUtils::MakeMaker:
    - ★ Utility to write a makefile from a Makefile.PL
    - ★ Is based on Makefile.SH
    - ★ perl Makefile.PL
      make
      make test
      make install
      make tardis

How to prepare software for distribution? III

- Python
  - Distutils is used to distribute modules
    1. Write a setup script:
       e.g:
       from distutils.core import setup
       setup(name='foo', version='1.0', py_modules=['foo'])
    2. Write a configuration file (optional)
    3. Create a source distribution:
       e.g:
       python setup.py sdist
       sdist will create a archive file (*.tar, *.zip)
    4. End user can install it with:
       python setup.py install

What is Plain Old Documentation (POD) format?

- Simple markup language
- Used to write documentation for Perl (programs, modules)
- Many converter available (text, xml, html, man pages...)
- Easy to read in source-code-format
- Consists of three paragraphs:
  - **ordinary:** normal text imbedded into blank lines
  - **verbatim:** used for presenting codeblocks each line starts with a space or tab
  - **command:** used for special treatment of text(blocks) each command starts with '=' followed by an identifyer

    e.g: =head1(2,3,4) defining header
    =item * a item
    =beging <format>
    =end <format>(regions of special format e.g. html)

References I

- http://en.wikipedia.org/wiki/Tar_(file_format)
- http://www.opussoftware.com/tutorial/TutMakefile.htm
- http://en.wikipedia.org/wiki/Make_(software)
- http://www.gnu.org/prep/standards/html_node/Makefile-Conventions.html#Makefile-Conventions
- http://www.gnu.org/software/make/manual/make.html#Makefiles
- http://www.gnu.org/software/make/manual/make.html#Rules
- http://www.gnu.org/software/make/manual/html_node/Parallel.html#P
- http://www.kitware.com/blog/home/post/434
- http://www.cmake.org
- http://martine.github.io/ninja
- http://www.gnu.org/prep/standards/html_node/DESTDIR.html#DEST
- http://www.gnu.org/prep/standards/html_node/Standard-Targets.html#Standard-Targets

References II

- http://www.c-howto.de/tutorial-makefiles.html
- http://search.cpan.org/ mschwern/ExtUtils-MakeMaker-6.62/lib/ExtUtils/MakeMaker.pm
- http://www.tu-chemnitz.de/docs/perldoc-html/ExtUtils/MakeMaker/Tutorial.html
- http://www.vromans.org/johan/articles/makemaker.html
- http://search.cpan.org/ leont/Module-Build-0.40/lib/Module/Build.pm
- http://www.lemoda.net/perl/module-build
- http://search.cpan.org/ leont/Module-Build-0.40/lib/Module/Build/Cookbook.pm
- http://en.wikipedia.org/wiki/Plain_Old_Documentation
- http://search.cpan.org/dist/perl/pod/perlpod.pod