This presentation...

- Basic “how to” for the autotools, so:
  - Can use autotools in simple cases
  - Can handle common error messages
  - Other info will make sense

- Intended for software developers who know how to use command line on a Unix-like system
  - Including shell (sh) and make
What are the autotools? What do they do?

- Autotools = Common name for autoconf + automake + libtool + ...

- Used by software developers (esp. C/C++) to create/distribute *automatically buildable* source code for various Unix-like (POSIX-like) systems

- Autotools make it easy to support:
  - Portability: Source code packages “just build”
  - Common build facilities (that users depend on!)
    - “make install”, select tools (e.g., CC=...), select destinations (e.g., prefix=...), DESTDIR (*vital* for packaging), VPATH builds (read-only source dirs), cross-compilation, config.site, ...
  - Auto-dependency generation for C/C++
Autotools (simplified view)

What users do:

```
./configure ...
make ...
[sudo] ... make install
```
Autotools (simplified view)

What developers create:
- configure.ac
- Makefile.am

What users do:
- ./configure ...
- Makefile.in
- Makefile
- make ...
- [sudo] ... make install

autotools
(Invoke using autoreconf)
Trivial example

1. In file “configure.ac” (for autoconf):
   ```
   AC_INIT([hello], [0.01])
   AC_OUTPUT
   ```

2. Create “configure” by running (on the command line):
   ```
   autoreconf -i
   ```

   The “-i” means “create/install any support files needed”

3. This “configure” doesn’t do much, but you can run it:
   ```
   ./configure
   ```
configure.ac language (for autoconf)

• configure.ac written in special language
  ◊ Really Bourne shell script processed by ‘m4’ macro processor, but usually use pre-created definitions
  ◊ “#” is comment character

• Key style rules (due to m4’s use):
  ◊ Bracket parameters with [ ] – needed in surprising places, best to always do it (integers don’t need it)
  ◊ Whitespace matters!
    - Whitespace before parameter ok (ignored outside [])
    - No whitespace before macro invocation’s (“”
    - No whitespace after parameters (before “,” or “”)”)
Better configure.ac (for autoconf)

AC_INIT([hello], [0.01], [x@example.com],
        [hello], [http://www.dwheeler.com/])

AC_PREREQ([2.68])

autoconf version ≥ 2.68

Safety: File must exist

AC_CONFIG_SRCDIR([hello.c])

Make config header

AC_CONFIG_HEADERS([config.h])

Auxiliary files go here

AC_CONFIG_AUX_DIR([build-aux])

Init automake

AM_INIT_AUTOMAKE([1.11 -Wall -Werror])

“configure” creates Makefile

AC_CONFIG_FILES([Makefile])

Find & probe C compiler

AC_PROG_CC

# Put various checks and such here.

AC_OUTPUT
Other configure.ac possibilities

• There are lots of predefined autoconf macros that probe for common circumstances, e.g.:
  ◊ AC_PROG_CXX: Find a C++ compiler
  ◊ AC_PROG_LEX: Find flex/lex
  ◊ AC_PROG_YACC: Find bison/yacc

• See autoconf manual for more

• See also: “GNU autoconf archive” = predefined autoconf macros
Trivial “hello.c” example

#include <stdio.h>

int
main()
{
    printf ("Hello, world!\n");
    return 0;
}

We need a program, so here's a trivial program for demonstration purposes.
Initial Makefile.am (for automake)

Lists programs to be installed in “bin” directory

\texttt{bin\_PROGRAMS = hello}

List of “targets”

Lists source files needed to generate target “hello”

\texttt{hello\_SOURCES = hello.c}

Makefile.am is a makefile... but assignments to variables with certain name patterns also generate code.
Using the example

- Create configure.ac & Makefile.am; can now generate configure, Makefile.in, etc.:
  
  autoreconf -i      # autoreconf > autoconf
  
  Required: README, etc. Create & check in.

- Build the program:
  
  ./configure
  make    # After this, rerun make for changes

- Try out some of the auto-generated capabilities:
  
  DESTDIR="$t" make install
  DESTDIR="$t" make uninstall
  make dist    # Create distribution tarball
  make distcheck # CHECK BEFORE RELEASE
Modifying/adding source files

- E.g., add: #include "config.h"
- Modify Makefile.am to note new SOURCEs
  - Be sure all non-generated source (e.g., .c and .h) is listed as a SOURCE
  - Add new files to your SCM (e.g., git’s add, commit)
- Run “make” (whenever) to remake everything
  - Automatic dependency calculation
- Run “make distcheck” to detect some errors
Makefile.am: 
{WHERE}_{PRIMARY} variables

- {WHERE}_{PRIMARY} = targets...
  - Create target types {PRIMARY} & put in {WHERE}
- {WHERE}: a makefile variable ending in “dir”
  - “bin” = $(bindir) for executables, default $(prefix)/bin
  - “lib” = $(libdir) for libraries, default $(prefix)/lib
  - “noinst” = not installed, “check” = for “make check”
- {PRIMARY}: the type of file
  - PROGRAMS= executable (binary) file, _SCRIPTS= executable scripts, _DATA= data, ...
- Example: bin_PROGRAMS
Makefile.am: Default settings

• Some AM_.... variable names in Makefile.am define automake-wide values, e.g.:
  ◦ AM_CCPPFLAGS: Default C preprocessor flags
  ◦ AM_CFLAGS: Default C compiler flags
  ◦ AM_CXXFLAGS: Default C++ compiler flags

• Do not set CPPFLAGS, CFLAGS, CXXFLAGS, and similar in Makefile.am
  ◦ Leave them be, so users can set them
Makefile.am: Target-specific variables

- Form: `{TARGET}_{SPECIFICS} = files...`
  - Sets target-specific info (overrides defaults, if any)
  - Variable’s TARGET name = original name but “_” replaces chars neither ASCII alphanumeric nor @

- `{TARGET}_SOURCES`: this target’s sources
  - Example: hello_SOURCES

- `{TARGET}_LDADD`: Extra objects for program

- `{TARGET}_CPPFLAGS`: this target’s C preprocessor flags
  - hello_CPPFLAGS = -DDEBUG
Common autotools error messages... and what to do

- In our example, adding to Makefile.am hello_CPPFLAGS=-DDEBUG will report errors
- Error “<Action> requires AM_... in configure.ac”
  ◦ Capabilities’ prerequisite not included
  ◦ Modify configure.ac as instructed. Same for AC_...
- Error “required file... not found... automake – add-missing can install...”
  ◦ Missing auxiliary files. Run “autoreconf -i”
- Now you know what to do!
Using PKG_CHECK_MODULES

- “pkg-config” system = easy way to use libraries
  ◦ Ensure pkg-config installed
  ◦ Not all libraries use it

- In configure.ac add:
  ```
  PKG_CHECK_MODULES([DEPS], [list-of-libs])
  ```
  - List-of-libs is space-separated list of library names
  - Library name may add “>= version number”
  - Sets makefile DEPS_CFLAGS & DEPS_LIBS

- In Makefile.am use those variables, e.g.:
  ```
  AM_CFLAGS = $(DEPS_CFLAGS)
  AM_LIBS = $(DEPS_LIBS)
  ```
Recap: Autotools (simplified view)

What developers create:
- configure.ac
- Makefile.am

What users do:
- ./configure ...
- make ...
- [sudo] ... make install

Invoke using autoreconf

autotools

autoconf

automake
Those are the basics...

But Wait!

There's More!

icanhasc一直都是最受欢迎的网站之一。
Creating “m4” subdirectory

- Modern convention: Use “m4” subdirectory for internal “m4” files (not default aclocal.m4):
  - Make the m4 directory
    `rm aclocal.m4 ; mkdir m4`
  - Tell autoconf to use it, by adding to configure.ac:
    `AC_CONFIG_MACRO_DIR([m4])`
  - Also add to Makefile.am:
    `ACLOCAL_AMFLAGS = -I m4 --install`
- But “autoreconf -i” fails (bug in autoconf version 2.68) & some SCMs (e.g., git) won’t store empty directories
Solution for “m4” subdirectory

• Do as above, but also
  ◊ Create dummy file in m4:
    touch m4/NOTES
    git add m4/NOTES # Put in SCM
  ◊ Add to Makefile.am, to force redistribution:
    EXTRA_DIST = m4/NOTES
  ◊ Then “autoreconf -i” to repair internals

• From then on, “m4” subdirectory works
Recursive make: Supported, but don’t do it

- Can organize source in subdirectories
- Traditionally built with “recursive make”
  - “make” called on each subdirectory
  - Autotools supports this: add “SUBDIRS =” in top Makefile.am, create Makefile.am in each dir, use AC_CONFIG_FILES in configure.ac to list them
- Recursive make is traditional, but a bad idea
  - Often wrong & harder to maintain & slower
  - “Recursive make considered harmful” (Peter Miller)
  - Just use one big Makefile.am & non-recursive make
Non-recursive make with autotools

- Still put your files in subdirectories, e.g.:
  
  ```
  mkdir src ; mv *.c *.h src/
  ```

- Modify configure.ac:
  
  ◊ Modify AM_INIT_AUTOMAKE to add option “subdir-objects” (so objects are placed in subdirs)
  
  ◊ Modify AC_CONFIG_SRCDIR so that it says “src/hello.c” instead of “hello.c” (since it moved)

- Change Makefile.am to use new locations, e.g.:
  
  ```
  bin_PROGRAMS = hello
  hello_SOURCES = src/hello.c src/whine.c src/whine.h
  ```
Libtool: Handling libraries

- Initialize libtool: Add “LT_INIT” to configure.ac
- Sample Makefile.am (should also make .pc file):

```makefile
ACLOCAL_AMFLAGS = -I m4 --install
lib_LTLIBRARIES = libwhine-1.0.la
libwhine_1_0_la_SOURCES = src/whine.c src/whine.h
libwhine_1_0_la_LDFLAGS = -version-info 0:0:0
include_HEADERS = src/whine.h
bin_PROGRAMS = hello
hello_SOURCES = src/hello.c
hello_LDADD = $(lib_LTLIBRARIES)
```

In lib, install library
In include, install header
Program uses library
Autoconfiscation

- Autoconfiscation = Changing a program’s build system to use autotools

- Use “autoscan” program when starting autoconfiscation
  - Reads existing files...
  - Creates “configure.scan” (a draft configure.ac)

- Running it on this demo points out useful things to add to configure.ac, e.g.:
  - AC_PROG_INSTALL – find an install, $(INSTALL)
  - AC_PROG_AWK – find an “awk”, $(AWK)
Advanced configure.ac style rules

- Use `AS_IF & AS_CASE`, not `if...fi & case...esac`
  - These tell autoconf what’s conditional
- To pass a literal parameter (including something with `[…]`), surround with a second `[…]` pair
  - `AC_MSG_ERROR([[These are [square brackets]!!!]])`
  - `AC_MSG_ERROR([[These [are [square brackets]!!!]])`
- Use `AC_LANG_SOURCE` for code snippets
  - `AC_LANG_SOURCE([[int main() { return 0; }]])`
- Prefer “test”, not `[…]`, for conditions
Beware of obsolete information

- Beware of obsolete info – much has changed
- Do not start with “GNU Autoconf, Automake and Libtool” (Gary Vaughan et all, 2000)
  - “Goat book” http://sourceware.org/autobook/
  - It was great in 2000, but now bad place to start
- Documentation is probably obsolete if:
  - Creates “configure.in” instead of “configure.ac”
  - Invokes tools by hand (aclocal, autoheader, …) instead of just running “autoreconf”
  - Uses just “aclocal.m4” instead of “m4/”
  - Written before ~2006
More information

- Openismus’ autotools info, e.g.,
  - “Building C/C++ libraries with Automake and Autoconf”
- “Adventures in Autoconfiscation” (Jez Higgins)
- *Autotools mythbuster* (Diego Elio “Flameeyes” Pettenò)
- *Using GNU Autotools* slides (Alexandre Duret-Lutz)
- GNU’s (reference) manuals: autoconf, automake, libtool
Thanks for watching!

- See my website: http://www.dwheeler.com
- Produced, written & directed by David A. Wheeler
- Presentation developed using OpenOffice.org, LibreOffice, gtk-recordmydesktop & Fedora

Music credit: paniq – Beyond Good and Evil - Discordian Nights