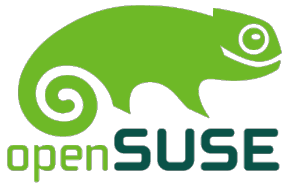


Buildsystems and what the heck for we actually use the autotools



Tomáš Chvátal
SUSE Packagers team

2013/07/19

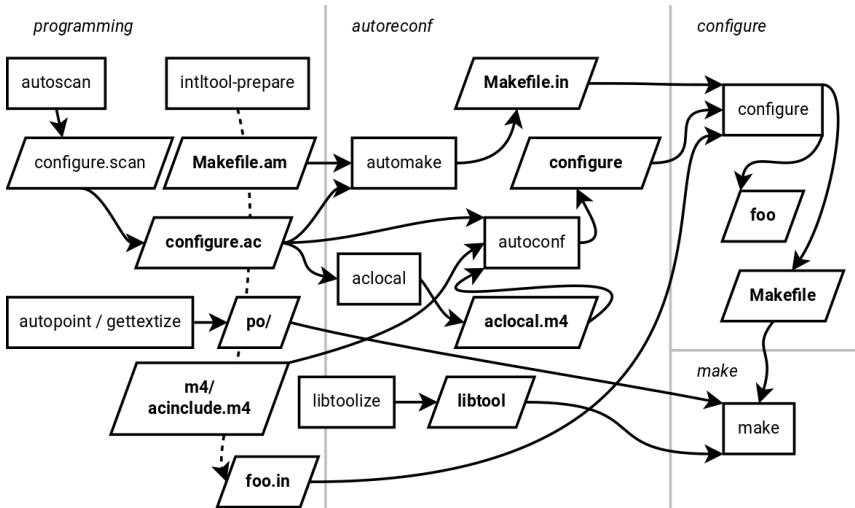
Introduction

Who the hell is Tomáš Chvátal

- SUSE Employee since 2011 - Team lead of packagers team
- Packager of Libreoffice and various other stuff for openSUSE
- openSUSE promoter and volunteer
- Gentoo developer since fall 2008

Autotools process

Complete autotools process



Make

Why not just a sh script?

Always recompiling everything is a waste of time and CPU power

Plain makefile example

```
CC      ?= @CC@
CFLAGS  ?= @CFLAGS@
PROGRAM = examplebinary
OBJ      = main.o parser.o output.o
$(PROGRAM): $(OBJ)
          $(CC) $(LDFLAGS) -o $@ $^
```

```
main.o: main.c common.h
parser.o: parser.c common.h
output.o: output.c common.h setup.h
```

```
install: $(PROGRAM)
# You have to use tabs here
          $(INSTALL) $(PROGRAM) $(BINDIR)
clean:
          $(RM) $(OBJ)
```


Variables in Makefiles

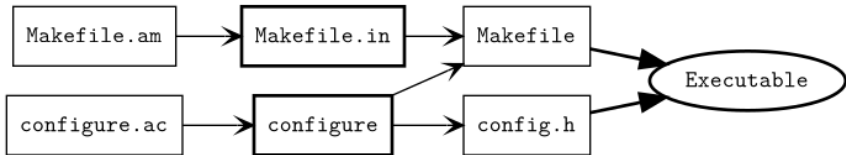
- Variables expanded using `$()`, ie `$(VAR)`
- Variables are assigned like in sh, ie `VAR=value`
- `$@` current target
- `$<` the first dependent file
- `^` all dependent files

Well nice, but why autotools then

- Makefiles can get complex fast (really unreadable)
- Lots of details to keep in mind when writing, small mistakes happen fast
- Does not make dependencies between targets really easier
- Automake gives you automatic tarball creation (make distcheck)

Autotools

Simplified autotools process



Autoconf/configure sample

```
AC_INIT(example, 0.1, bugs@example.com)
AC_CONFIG_HEADER([config.h])
```

```
AC_PROG_C
AC_PROG_CPP
AC_PROG_INSTALL
```

```
AC_HEADER_STDC
AC_CHECK_HEADERS([string.h unistd.h limits.h])
```

```
AC_CONFIG_FILES([Makefile
                  doc/Makefile
                  src/Makefile])
```

```
AC_OUTPUT
```

Autoconf syntax

- The M4 syntax is quite weird on the first read
- It is not interpreted, it is text substitution machine
- Lots of quoting is needed, if in doubt add more []
- Everything that does or might contain whitespace or commas has to be quoted
- Custom autoconf M4 macros are almost unreadable

Automake

```
bin_PROGRAMS = examplebinary
```

```
examplebinary_SOURCES = \  
    src/main.c \  
    src/parser.c \  
    src/output.c \  
    src/setup.c
```

```
noinst_HEADERS = src/common.h src/setup.h
```

Basic rules

- Always use just one Makefile.am in root folder
- All files that are to be distributed must be added to relevant parts or EXTRA_DIST
- Always run make distcheck to verify your package really works
- Use check_BINARIES/etc. . . to have test phase

Variables for automake - SUFFIXES

- `_PROGRAMS`
- `_LIBRARIES DO_NOT_USE` go for `_LTLIBRARIES`
- `_SCRIPTS`
- `_SOURCES`
- `_HEADERS`
- `_OBJECTS`
- `_DATA`
- `_LDADD`

Variables for automake - PREFIXES

- `bin_` will be installed to `bindir`
- `sbin_` will be installed to `sbindir`
- `lib_` will be installed to `libdir`
- `noinst_` will not be installed
- `EXTRA_` will be packaged upon `make dist`
- `check_` used only for `make check`

Libtool

Libtool versioning

- Start with version information of '0:0:0' for each libtool library
- If the library source code has changed at all since the last update, then increment revision ('c:r:a' becomes 'c:r+1:a')
- If any interfaces have been added, removed, or changed since the last update, increment current, and set revision to 0
- If any interfaces have been added since the last public release, then increment age
- If any interfaces have been removed or changed since the last public release, then set age to 0

configure.ac changes

```
LT_VERSION=m4_esyscmd([./version.sh -v])  
LT_INIT([disable-static pic-only])  
AC_PROG_LIBTOOL
```

Makefile.am changes

```
lib_LTLIBRARIES = libexample.la
libexample_la_SOURCES = \
    src/something.c \
    src/somethingelse.c \
    src/whatever.c
libexample_la_CFLAGS = \
    $(MYEXTERNALPACKAGE_CFLAGS)
libexample_la_LDFLAGS = \
    $(MYEXTERNALPACKAGE_LIBS) \
    -version-info $(LT_VERSION) \
    -export-symbols-regex '^foo_'
```

Autotools and windows

Initial thoughts

- Well for multiplatform support you can count on autotools on any UNIX-ish system
- On windows you have to use cygwin/mingw
- Per above you will spent bit of time getting that running
- You have to write yourself the .rc or rc.in file to be processed by cmake (see libvenge/etc.)

Changes for configure.ac

```
AC_MSG_CHECKING([for native Win32])
AS_CASE([ $host ],
  [ *-*-mingw* ], [
    native_win32=yes
    BINARY_WIN32_RESOURCE=binary-win32res.lo
    AC_CHECK_TOOL(WINDRES, windres)
  ], [
    native_win32=no
    BINARY_WIN32_RESOURCE=
  ]
)
# Ensure compat with MSVC
AS_IF([test "x$native_win32" = "xyes"], [
  AC_CHECK_TOOL(WINDRES, windres)
  AS_IF([test "x"$GCC = "xyes"], [
    AC_MSG_CHECKING([how to get MSVC-compatible struct packing])
    AS_IF([test -z "$ac_cv_prog_CC"], [
      our_gcc="$CC"
    ], [
      our_gcc="$ac_cv_prog_CC"
    ])
    AS_IF([ $our_gcc -v --help 2>/dev/null | grep ms-bitfields >/dev/null ], [
      msnative_struct="mms-bitfields"
      CFLAGS="$CFLAGS $msnative_struct"
      CXXFLAGS="$CXXFLAGS $msnative_struct"
      AC_MSG_RESULT([${msnative_struct}])
    ], [
      AC_MSG_RESULT([no way])
      AC_MSG_WARN([produced libraries might be incompatible with MSVC-c
```

Changes for Makefile.am

```
bin_PROGRAMS = examplebinary
```

```
examplebinary_SOURCES = \  
    src/main.c \  
    src/parser.c \  
    src/output.c \  
    src/setup.c
```

```
examplebinary_LDADD = \  
    $(OTHER_LIBS) \  
    @BINARY_WIN32_RESOURCE@
```

```
noinst_HEADERS = src/common.h src/setup.h
```

```
if OS_WIN32
```

```
@BINARY_WIN32_RESOURCE@ : examplebinary.rc $(examplebinary_OBJECTS)  
    chmod +x $(top_srcdir)/build/*compile-resource && \  
    WINDRES=@WINDRES@ $(top_srcdir)/build/lt-compile-resource examplebinary.rc @BINARY
```

```
endif
```

Additional points for Makefile.am

- Always pass `-avoid-version` to `libtool`
- Remember to add the resource file to `_DEPENDENCIES`
- Script to compile the `.lo` files
<https://github.com/AbiWord/enchant/blob/master/lt-compile-resource>

Autotools usability

- Not hard as people are led to believe - \bar{c} you can deploy it unless your files are too messy
- It, because of mingw, produces slower binaries than MSVC
- Most people are fine with it, but if not use Visual Studio project file and be done
- For .rc files you usually have to use some shellscript as libtool has no clue

CMake

What are the benefits?

- No libtool!
- Multiplatform generator for free Mac/Win/Linux...
- Can swap make for ninja

Any disadvantages?

- FindBLA.cmake are sometimes pretty crappy
- If you rely on just .pc files you loose multiplatformity
- Can get unreadable fast
- Conflicting guides online, fine when you have someone to ask
- Distribution archive generator using CPack confuse many people

CMake example

```
cmake_minimum_required(VERSION 2.8)
project(example C)
set(Example_VERSION_MAJOR 0)
set(Example_VERSION_MINOR 1)

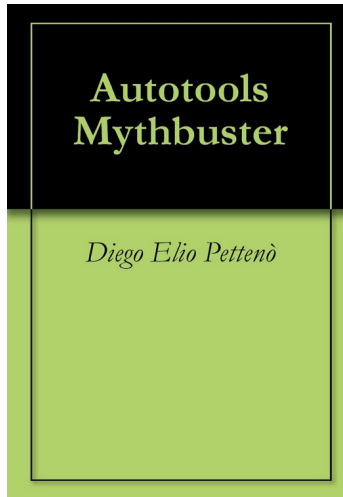
set(src_EXAMPLE
    src/main.c
    src/parser.c
    src/output.c
    src/setup.c
    src/common.h
    src/setup.h
)
add_executable(examplebinary ${src_EXAMPLE})
install(TARGETS examplebinary DESTINATION bin)
```


CPack example

```
include (InstallRequiredSystemLibraries)
set (CPACK_RESOURCE_FILE_LICENSE
    "${CMAKE_CURRENT_SOURCE_DIR}/LICENSE")
set (CPACK_PACKAGE_VERSION_MAJOR "${Tutorial_VERSION_MAJOR}")
set (CPACK_PACKAGE_VERSION_MINOR "${Tutorial_VERSION_MINOR}")
include (CPack)
```

Reading

Reading



Endnote

Thanks

Thank you for your attention.