Foreword (1/2)

This presentation targets developers familiar with Unix development tools (shell, make, compiler) that want to learn Autotools.

The latest version of this document can be retrieved from
`http://www.lrde.epita.fr/~adl/autotools.html`

Please mail me corrections and suggestions **about this document** at `adl@gnu.org`.

Do not send me any general question about the Autotools. Use the appropriate mailing list instead (`autoconf@gnu.org`, or `automake@gnu.org`).

Foreword (2/2)

This document was updated for the following releases of the Autotools:

GNU Autoconf	2.65	(November 2009)
GNU Automake	1.11.1	(December 2009)
GNU Libtool	2.2.6b	(November 2009)
GNU Gettext	0.17	(November 2007)

These were the last releases at the time of writing.

- The usage of these tools has improved a lot over the last years.
- Some syntaxes used here will not work with older tools.
- This a deliberate choice:
  - New users should learn today's recommended usages.
  - Make sure you have up-to-date tools and do not bother with old releases.

Using GNU Autotools

Alexandre Duret-Lutz
`adl@gnu.org`

May 16, 2010

Part I

The GNU Build System

Portable Packages

Sources of Non-Portability in C

Consider C functions...

- that do not exist everywhere (e.g., `strtod()`)
- that have different names (e.g., `strchr()` vs. `index()`)
- that have varying prototypes
  (e.g., `int setpgrp(void);` vs. `int setpgrp(int, int);`)
- that can behave differently (e.g., `malloc(0);`)
- that might require other libraries
  (is `pow()` in *libm.so* or in *libc.so*?)
- that can be defined in different headers
  (*string.h* vs. *strings.h* vs. *memory.h*)

How should a package deal with those?

Possible Solutions

- Slice the code with lots of `#if`/`#else`
- Create substitution macros
- Create substitution functions

The latter two are to be preferred.

Code Cluttered with `#if`/`#else`

Excerpt of ffcall-1.10's `alloc_trampoline()`

```
#if !defined(CODE_EXECUTABLE)
  static long pagesize = 0;
#if defined(EXECUTABLE_VIA_MMAP_DEVZERO)
  static int zero_fd;
#endif
  if (!pagesize) {
#if defined(HAVE_MACH_VM)
      pagesize = vm_page_size;
#else
      pagesize = getpagesize();
#endif
#if defined(EXECUTABLE_VIA_MMAP_DEVZERO)
      zero_fd = open("/dev/zero", O_RDONLY,0644);
      if (zero_fd < 0) {
        fprintf(stderr, "trampoline: Cannot open /dev/zero!\n");
        abort();
      }
#endif
  }
#endif
```

Substitution macros

Excerpt of coreutils-5.2.1's *system.h*

```
#if ! HAVE_FSEEKO && ! defined fseeko
# define fseeko(s, o, w) ((o) == (long) (o) \
                          ? fseek (s, o, w) \
                          : (errno = EOVERFLOW, -1))
#endif
```

Then use `fseeko()` whether it exists or not.

Substitution functions

If `strdup()` does not exist, link your program with a replacement definition such as

***strdup.c* (from the GNU C library)**

```
char *
strdup (const char *s)
{
  size_t len = strlen (s) + 1;
  void *new = malloc (len);
  if (new == NULL)
    return NULL;
  return (char *) memcpy (new, s, len);
}
```

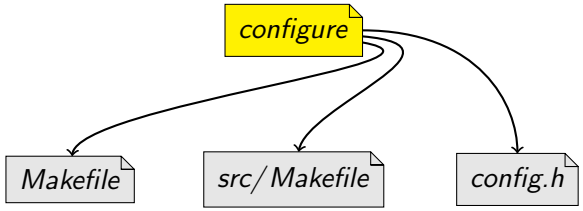

Uniform Builds

Need for Automatic Configuration

- Maintaining a collection of `#define` for each system by hand is cumbersome.
- Requiring users to add the necessary `-D`, `-I`, and `-l` compilation options to *Makefile* is burdensome.
- Complicated builds hinder the acceptance of free software.

- In 1991 people started to write shell scripts to **guess** these settings for some GNU packages.
- Since then the *configure* script is mandatory in any package of the GNU project.

configure's Purpose



- *configure* probes the systems for required functions, libraries, and tools
- then it generates a *config.h* file with all `#defines`
- as well as *Makefile*s to build the package

GNU Coding Standards

`http://www.gnu.org/prep/standards/`

Practices that packages of the GNU project should follow:

- program behavior
  - how to report errors,
  - standard command line options,
  - etc.
- coding style
- configuration
- *Makefile* conventions
- etc.

The User Point of View

Standard Installation Procedure

```
~ % tar zxf amhello-1.0.tar.gz
~ % cd amhello-1.0
~/amhello-1.0 % ./configure
...
~/amhello-1.0 % make
...
~/amhello-1.0 % make check
...
~/amhello-1.0 % su
Password:
/home/adl/amhello-1.0 # make install
...
/home/adl/amhello-1.0 # exit
~/amhello-1.0 % make installcheck
...
```

Standard Makefile Targets

'make all'	Build programs, libraries, documentation, etc. (Same as 'make'.)
'make install'	Install what needs to be installed.
'make install-strip'	Same as 'make install', then strip debugging symbols.
'make uninstall'	The opposite of 'make install'.
'make clean'	Erase what has been built (the opposite of 'make all').
'make distclean'	Additionally erase anything './configure' created.
'make check'	Run the test suite, if any.
'make installcheck'	Check the installed programs or libraries, if supported.
'make dist'	Create *PACKAGE-VERSION.tar.gz*.

Standard File System Hierarchy

Directory variable	Default value
prefix	*/usr/local*
exec-prefix	prefix
bindir	exec-prefix*/bin*
libdir	exec-prefix*/lib*
...	
includedir	prefix*/include*
datarootdir	prefix*/share*
datadir	datarootdir
mandir	datarootdir*/man*
infodir	datarootdir*/info*
...	

```
~/amhello-1.0 % ./configure --prefix ~/usr
~/amhello-1.0 % make
~/amhello-1.0 % make install
```


Standard Configuration Variables

'./configure' automatically detects many settings.
You can force some of them using configuration variables.

CC	C compiler command
CFLAGS	C compiler flags
CXX	C++ compiler command
CXXFLAGS	C++ compiler flags
LDFLAGS	linker flags
CPPFLAGS	C/C++ preprocessor flags
...	See './configure --help' for a full list.

```
~/amhello-1.0 % ./configure --prefix ~/usr CC=gcc-3 \
CPPFLAGS=-I$HOME/usr/include LDFLAGS=-L$HOME/usr/lib
```


The Power User Point of View

Overriding Default Configuration Settings with *config.site*

Recall that old command

```
~/amhello-1.0 % ./configure --prefix ~/usr CC=gcc-3 \
CPPFLAGS=-I$HOME/usr/include LDFLAGS=-L$HOME/usr/lib
```

Common configuration settings can be put in prefix/*share/config.site*

```
~/amhello-1.0 % cat ~/usr/share/config.site
test -z "$CC" && CC=gcc-3
test -z "$CPPFLAGS" && CPPFLAGS=-I$HOME/usr/include
test -z "$LDFLAGS" && LDFLAGS=-L$HOME/usr/lib
```

Reducing the command to...

```
~/amhello-1.0 % ./configure --prefix ~/usr
configure: loading site script /home/adl/usr/share/config.site
...
```


Parallel Build Trees (a.k.a. VPATH Builds)

Objects files, programs, and libraries are built where *configure* was run.

```
~ % tar zxf ~/amhello-1.0.tar.gz
~ % cd amhello-1.0
~/amhello-1.0 % mkdir build && cd build
~/amhello-1.0/build % ../configure
~/amhello-1.0/build % make
...
```

Sources files are in *~/amhello-1.0/*,
built files are all in *~/amhello-1.0/build/*.

Parallel Build Trees for Multiple Architectures

Builds for multiple architectures can share the same source tree.

Have the source on a (possibly read-only) shared directory

```
~ % cd /nfs/src
/nfs/src % tar zxf ~/amhello-1.0.tar.gz
```

Compilation on first host

```
~ % mkdir /tmp/amh && cd /tmp/amh
/tmp/amh % /nfs/src/amhello-1.0/configure
/tmp/amh % make && sudo make install
```

Compilation on second host, assuming shared data

```
~ % mkdir /tmp/amh && cd /tmp/amh
/tmp/amh % /nfs/src/amhello-1.0/configure
/tmp/amh % make && sudo make install-exec
```


Two Part Installation

'make install'	
=	
'make install-exec'	install platform-dependent files
+	
'make install-data'	install platform-independent files (can be shared among multiple machines)

Cross-Compilation

```
~/amhello-1.0 % ./configure --build i686-pc-linux-gnu \
                            --host i586-mingw32msvc
checking for a BSD-compatible install... /usr/bin/install -c
checking whether build environment is sane... yes
checking for gawk... gawk
checking whether make sets $(MAKE)... yes
checking for i586-mingw32msvc-strip... i586-mingw32msvc-strip
checking for i586-mingw32msvc-gcc... i586-mingw32msvc-gcc
checking for C compiler default output file name... a.exe
checking whether the C compiler works... yes
checking whether we are cross compiling... yes
checking for suffix of executables... .exe
checking for suffix of object files... o
checking whether we are using the GNU C compiler... yes
checking whether i586-mingw32msvc-gcc accepts -g... yes
checking for i586-mingw32msvc-gcc option to accept ANSI C...
...
```


Cross-Compilation

```
~/amhello-1.0 % ./configure --build i686-pc-linux-gnu \
                            --host i586-mingw32msvc
...
~/amhello-1.0 % make
...
~/amhello-1.0 % cd src; file hello.exe
hello.exe: MS Windows PE 32-bit Intel 80386 console executable not relocatable
```

Of course you need a cross-compiler installed first.

Cross-compilation *configure* options:

- `'--build=BUILD'` The system on which the package is built.
- `'--host=HOST'` The system where built programs & libraries will run.
- `'--target=TARGET'` Only when building compiler tools: the system for which the tools will create output.

For simple cross-compilation, only `'--host=HOST'` is needed.

Renaming Programs at Install Time

Maybe *hello* is already a command on this host?

`'--program-prefix=PREFIX'`
: prepend PREFIX to installed program names,

`'--program-suffix=SUFFIX'`
: append SUFFIX to installed program names,

`'--program-transform-name=PROGRAM'`
: run 'sed PROGRAM' on installed program names.

```
~/amhello-1.0 % ./configure --program-prefix test-
~/amhello-1.0 % make
~/amhello-1.0 % sudo make install
```

Will install *hello* as */usr/local/bin/test-hello*.

The Packager Point of View

Building Binary Packages Using DESTDIR

DESTDIR is used to relocate a package at install time.

```
~/amhello-1.0 % ./configure --prefix /usr
...
~/amhello-1.0 % make
...
~/amhello-1.0 % make DESTDIR=$HOME/inst install
...
~/amhello-1.0 % cd ~/inst
~/inst % tar zcvf ~/amhello-1.0-i686.tar.gz .
./
./usr/
./usr/bin/
./usr/bin/hello
```

... and *~/amhello-1.0-i686.tar.gz* is ready to be uncompressed in */* on many hosts.

The Maintainer Point of View

Preparing Distributions

'`make dist`' Create *PACKAGE-VERSION.tar.gz*.

'`make distcheck`' Likewise, with many sanity checks. Prefer this one!

'`make distcheck`' ensures most of the use cases presented so far work.

- It tests VPATH builds (with read-only source tree)
- It ensures '`make clean`', '`make distclean`', and '`make uninstall`' do not omit files,
- It checks that DESTDIR installations work,
- It runs the test suite (both '`make check`' and '`make installcheck`').

Releasing a package that fails '`make distcheck`' means releasing a package that will disappoint many users.

Automatic Dependency Tracking

```
~/amhello-1.0 % ./configure --prefix /usr
...
checking dependency style of gcc... gcc3
...
```

Dependency tracking is performed as a side-effect of compilation.
Several methods are supported, and checked for by *configure*.
(The gcc3 method above is the fastest.)

Dependency tracking is only needed when the source files change;
it can be safely disabled for throw-away installation builds.
Slow methods must be enabled explicitly.

'`--disable-dependency-tracking`' speed up one-time builds

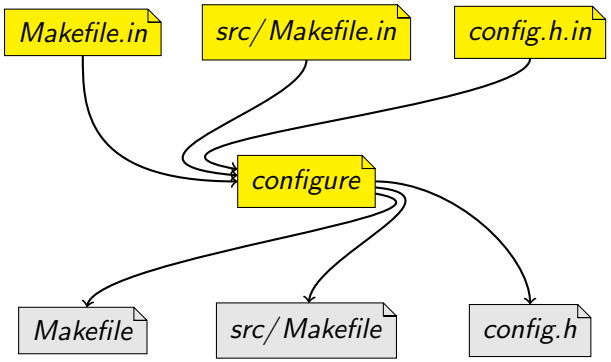
'`--enable-dependency-tracking`' do not reject slow dependency extractors

Nested Packages

- *Autoconfiscated* packages can be nested to arbitrary depth.
  - A package can distribute a third-party library it uses in a subdirectory.
  - It's possible to gather many packages this way to distribute a set of tools.
- For installers:
  - A single package to configure, build, and install.
  - '`configure`' options are passed recursively to sub-packages.
  - '`configure --help=recursive`' shows the help of all sub-packages.
- For maintainers:
  - Easier integration.
  - The sub-package is autonomous.

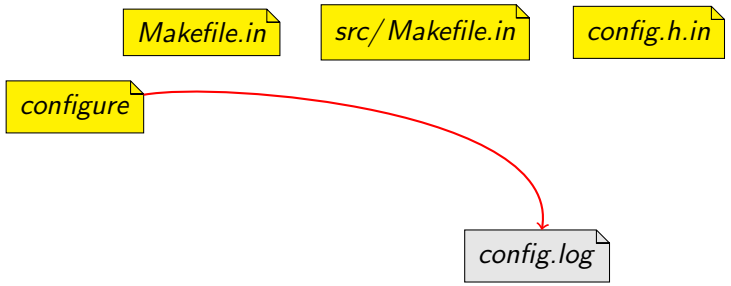
The configure Process

The (simplified) *configure* process



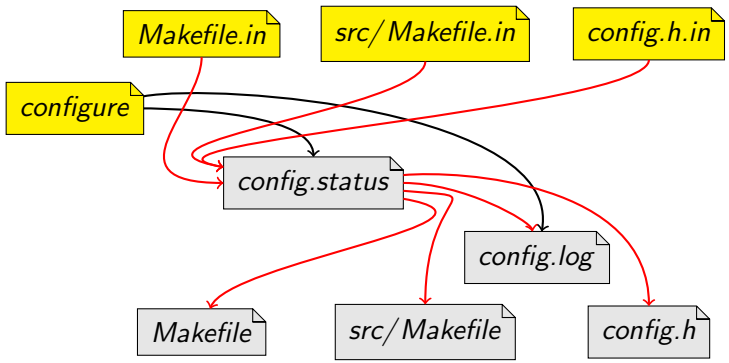
**.in* files are configuration templates
from which *configure* generates the configuration files to use for building

The (real) *configure* process



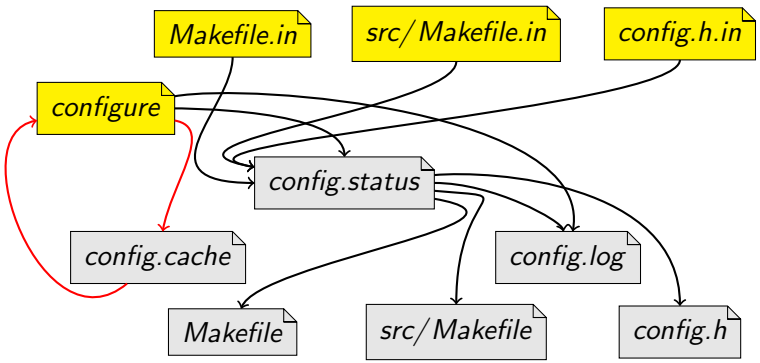
config.log contains a trace of the configuration

The (real) *configure* process



config.status will actually process the templates

The (real) *configure* process



'`configure -C`' caches results in *config.cache* to speed up reconfigurations

Why We Need Tools

Why We Need Tools

If you try to mimic this build system by hand, you'll discover that

- The GNU Build System has a lot of features.
  Some users may expect features you do not use.
- Implementing them portably is difficult, and exhausting.
  (Think portable shell scripts, portable *Makefile*s, on systems you may not have handy.)
- You will have to upgrade your setup to follow changes of the GNU Coding Standards.

GNU Autotools provide:

- Tools to create the GNU Build System from simple instructions.
- A central place where fixes and improvements are made.
  (A bug-fix for a portability issue benefits every package.)

Part II

GNU Autotools

Hello World

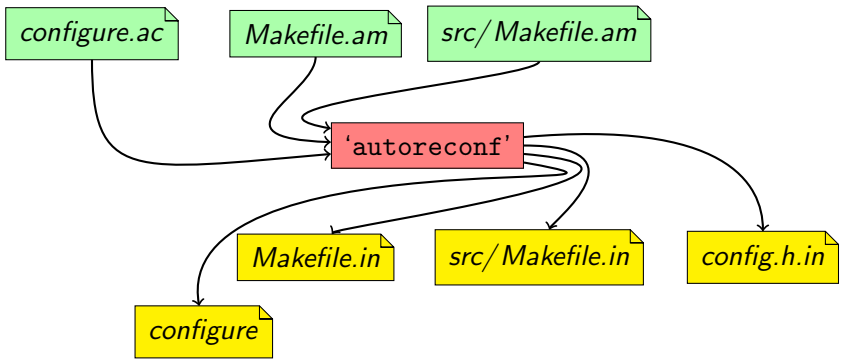
src/main.c for Hello World

src/main.c

```
#include <config.h>
#include <stdio.h>

int
main (void)
{
  puts ("Hello World!");
  puts ("This is " PACKAGE_STRING ".");
  return 0;
}
```


Generating All Template Files



Autotools Inputs

configure.ac

```
AC_INIT([amhello], [1.0], [bug-report@address])
AM_INIT_AUTOMAKE([foreign -Wall -Werror])
AC_PROG_CC
AC_CONFIG_HEADERS([config.h])
AC_CONFIG_FILES([Makefile src/Makefile])
AC_OUTPUT
```

Makefile.am

```
SUBDIRS = src
```

src/Makefile.am

```
bin_PROGRAMS = hello
hello_SOURCES = main.c
```


Preparing the Package

```
~/amhello % ls -R
.:
Makefile.am  configure.ac  src/

./src:
Makefile.am  main.c
~/amhello % autoreconf --install
configure.ac:2: installing `./install-sh'
configure.ac:2: installing `./missing'
src/Makefile.am: installing `./depcomp'
~/amhello %
```


Preparing the Package

```
~/amhello % ls -R
.:
Makefile.am      configure.ac
Makefile.in      depcomp*
aclocal.m4       install-sh*
autom4te.cache/  missing*
config.h.in      src/
configure*

./autom4te.cache:
output.0  requests  traces.1
output.1  traces.0

./src:
Makefile.am  Makefile.in  main.c
```

expected configuration templates

Preparing the Package

```
~/amhello % ls -R
.:
Makefile.am      configure.ac
Makefile.in      depcomp*
aclocal.m4       install-sh*
autom4te.cache/  missing*
config.h.in      src/
configure*

./autom4te.cache:
output.0  requests  traces.1
output.1  traces.0

./src:
Makefile.am  Makefile.in  main.c
```

definitions for third-party macros used in *configure.ac*

Preparing the Package

```
~/amhello % ls -R
.:
Makefile.am       configure.ac
Makefile.in       depcomp*
aclocal.m4        install-sh*
autom4te.cache/   missing*
config.h.in       src/
configure*

./autom4te.cache:
output.0  requests  traces.1
output.1  traces.0

./src:
Makefile.am  Makefile.in  main.c
```

auxiliary tools used during the build

Preparing the Package

```
~/amhello % ls -R
.:
Makefile.am       configure.ac
Makefile.in       depcomp*
aclocal.m4        install-sh*
autom4te.cache/   missing*
config.h.in       src/
configure*

./autom4te.cache:
output.0  requests  traces.1
output.1  traces.0

./src:
Makefile.am  Makefile.in  main.c
```

Autotools cache files

Preparing the Package

```
~/amhello % ./configure
checking for a BSD-compatible install... /usr/bin/install -c
checking whether build environment is sane... yes
checking for gawk... gawk
checking whether make sets $(MAKE)... yes
checking for gcc... gcc
...
checking dependency style of gcc... gcc3
configure: creating ./config.status
config.status: creating Makefile
config.status: creating src/Makefile
config.status: creating config.h
config.status: executing depfiles commands
~/amhello % make
...
```


Preparing the Package

```
~/amhello % src/hello
Hello World!
This is amhello 1.0.
~/amhello % make distcheck
...
========================================
amhello archives ready for distribution:
amhello-1.0.tar.gz
========================================
~/amhello %
```

Preparing the Package

```
~/amhello % tar ztf amhello-1.0.tar.gz
amhello-1.0/
amhello-1.0/Makefile.am
amhello-1.0/Makefile.in
amhello-1.0/aclocal.m4
amhello-1.0/config.h.in
amhello-1.0/configure
amhello-1.0/configure.ac
amhello-1.0/depcomp
amhello-1.0/install-sh
amhello-1.0/missing
amhello-1.0/src/
amhello-1.0/src/Makefile.am
amhello-1.0/src/Makefile.in
amhello-1.0/src/main.c
~/amhello %
```


Introducing Core Autotools

Two Core Packages

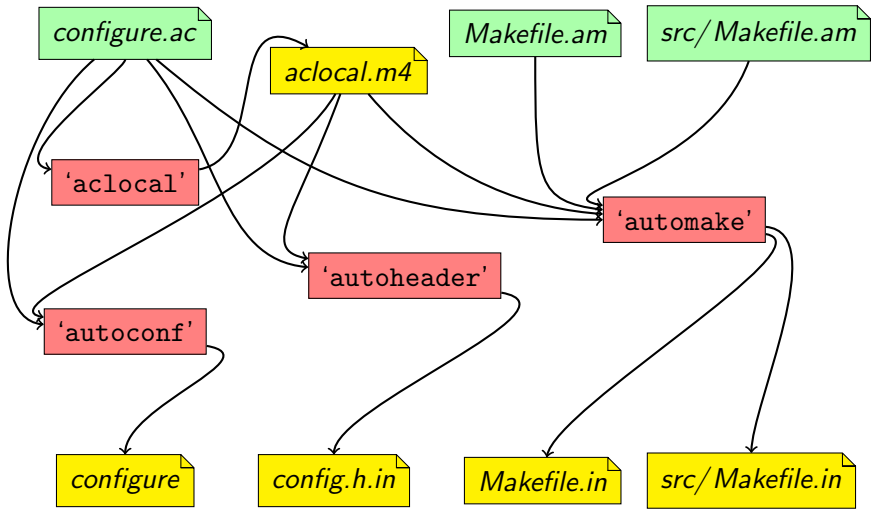
GNU Autoconf

- `'autoconf'` Create *configure* from *configure.ac*.
- `'autoheader'` Create *config.h.in* from *configure.ac*.
- `'autoreconf'` Run all tools in the right order.
- `'autoscan'` Scan sources for common portability problems, and related macros missing from *configure.ac*.
- `'autoupdate'` Update obsolete macros in *configure.ac*.
- `'ifnames'` Gather identifiers from all `#if`/`#ifdef`/... directives.
- `'autom4te'` The heart of Autoconf. It drives M4 and implements the features used by most of the above tools. Useful for creating more than just *configure* files.

GNU Automake

- `'automake'` Create *Makefile.in*s from *Makefile.am*s and *configure.ac*.
- `'aclocal'` Scan *configure.ac* for uses of third-party macros, and gather definitions in *aclocal.m4*.

Behind 'autoreconf'



'autoreconf' is Your Friend

In practice,

- You do not have to remember the interaction of all tools.
- Use 'autoreconf --install' to setup the package initially.
- Rely on the rebuild rules (output in *Makefile*s) to rerun the right autotool when you change some input file.
- You only need a rough idea of the purpose of each tool to understand errors. (What tool complains and about what?)

'autoconf' Creates *configure* from *configure.ac*.

'autoheader' Creates *config.h.in* from *configure.ac*.

'automake' Creates *Makefile.in*s from *Makefile.am*s and *configure.ac*.

'aclocal' Scans *configure.ac* for uses of third-party macros, and gather definitions in *aclocal.m4*.

'autom4te' Autoconf driver for M4. All tools that process *configure.ac* do so through 'autom4te'.

Hello World Explained

amhello's *configure.ac* explained

configure.ac

```
AC_INIT([amhello], [1.0], [bug-report@address])
AM_INIT_AUTOMAKE([foreign -Wall -Werror])
AC_PROG_CC
AC_CONFIG_HEADERS([config.h])
AC_CONFIG_FILES([Makefile src/Makefile])
AC_OUTPUT
```

- Initialize Autoconf. Specify package's name, version number, and bug-report address.
- Initialize Automake. Turn on all Automake warnings and report them as errors. This is a foreign package.
- Check for a C compiler.
- Declare *config.h* as output header.
- Declare *Makefile* and *src/Makefile* as output files.
- Actually output all declared files.

foreign Ignores some GNU Coding Standards

configure.ac

```
...
AM_INIT_AUTOMAKE([foreign -Wall -Werror])
...
```

```
~/amhello % autoreconf --install
configure.ac:2: installing './install-sh'
configure.ac:2: installing './missing'
src/Makefile.am: installing './depcomp'
```

foreign Ignores some GNU Coding Standards

configure.ac without the foreign option

```
...
AM_INIT_AUTOMAKE([ -Wall -Werror])
...
```

```
~/amhello % autoreconf --install
configure.ac:2: installing `./install-sh'
configure.ac:2: installing `./missing'
src/Makefile.am: installing `./depcomp'
Makefile.am: installing `./INSTALL'
Makefile.am: required file `./NEWS' not found
Makefile.am: required file `./README' not found
Makefile.am: required file `./AUTHORS' not found
Makefile.am: required file `./ChangeLog' not found
Makefile.am: installing `./COPYING'
autoreconf: automake failed with exit status: 1
```


amhello's *src/Makefile.am* explained

src/Makefile.am

```
bin_PROGRAMS = hello
hello_SOURCES = main.c
```

- We are building some programs.
- These programs will be installed in bindir.
- There is only one program to build: *hello*.
- To create *hello*, just compile *main.c*.

amhello's *Makefile.am* explained

Makefile.am

```
SUBDIRS = src
```

- Build recursively in *src/*.
- Nothing else is declared for the current directory.
  (The top-level *Makefile.am* is usually short.)

Using Autoconf

From *configure.ac* to *configure* and *config.h.in*

- 'autoconf' is a macro processor.
- It converts *configure.ac*, which is a shell script using macro instructions, into *configure*, a full-fledged shell script.
- Autoconf offers many macros to perform common configuration checks.
- It is not uncommon to have a *configure.ac* without shell constructs, using only macros.
- While processing *configure.ac* it is also possible to trace the occurrences of macros. This is how 'autoheader' creates *config.h.in*. It just looks for the macros that #define symbols.
- The real macro processor actually is GNU M4. Autoconf offers some infrastructure on top of that, plus the pool of macros.

Discovering M4

example.m4

```
m4_define(NAME1, Harry)↲
m4_define(NAME2, Sally)↲
m4_define(MET, $1 met $2)↲
MET(NAME1, NAME2)↲
```

```
~ % m4 -P example.m4
↲
↲
↲
Harry met Sally↲
```


M4 Quoting

- The macro's arguments are processed
- Then the macro is expanded
- Finally the output of the macro is processed too
- A string can be protected from processing using quotes.

This is a source of many mistakes for the unwary.

example.m4

```
m4_define(NAME1, ‘Harry, Jr.’)↲
m4_define(NAME2, Sally)↲
m4_define(MET, $1 met $2)↲
MET(NAME1, NAME2)↲
```

Can you guess the output of the above?

M4 Quoting Rule of the Thumb

- Quote each macro argument once.
- So it is processed only after it has been output.

example.m4

```
m4_define(‘NAME1’, ‘Harry, Jr.’)↲
m4_define(‘NAME2’, ‘Sally’)↲
m4_define(‘MET’, ‘$1 met $2’)↲
MET(‘NAME1’, ‘NAME2’)↲
```

Spacing Matters

- The parenthesis must stick to the macro name.

example.m4

```
m4_define(‘NAME1’, ‘Harry, Jr.’)↲
m4_define(‘NAME2’, ‘Sally’)↲
m4_define(‘MET’, ‘$1 met $2’)↲
MET␣(‘NAME1’, ‘NAME2’)↲
```

```
~ % m4 -P example.m4
↲
↲
↲
 met ␣(NAME1, NAME2)↲
```


Spacing Matters

- Spaces after or inside quotes are part of the arguments.

example.m4

```
m4_define(‘NAME1’, ‘Harry, Jr.’)↲
m4_define(‘NAME2’, ‘Sally’)↲
m4_define(‘MET’, ‘$1 met $2’)↲
MET(‘␣NAME1␣’␣, ‘NAME2’)↲
```

```
~ % m4 -P example.m4
↲
↲
↲
␣Harry, Jr.␣␣ met Sally↲
```


Spacing Matters

- Spaces before quotes are ignored.

example.m4

```
m4_define(‘NAME1’,␣‘Harry, Jr.’)↲
m4_define(‘NAME2’,␣‘Sally’)↲
m4_define(‘MET’,␣‘$1 met $2’)↲
MET(␣‘NAME1’,␣‘NAME2’)↲
```

```
~ % m4 -P example.m4
↲
↲
↲
Harry, Jr. met Sally↲
```


Autoconf on Top of M4

- Autoconf = M4 with more machinery, and many predefined macros.
- The quotes are `[` and `]` (instead of ‘ and ’).
- For this reason we use the `test` command instead of `[` in shell fragments:

```
if test "$x" = "$y"; then ...
```

- Macros are defined with `AC_DEFUN`.

```
AC_DEFUN([NAME1], [Harry, Jr.])
AC_DEFUN([NAME2], [Sally])
AC_DEFUN([MET], [$1 met $2])
MET([NAME1], [NAME2])
```

The Structure of a *configure.ac*

configure.ac

```
# Prelude.
AC_INIT([PACKAGE], [VERSION], [BUG-REPORT-ADDRESS])
AM_INIT_AUTOMAKE([foreign -Wall -Werror])
# Checks for programs.
AC_PROG_CC
# Checks for libraries.
# Checks for header files.
# Checks for typedefs, structures, and compiler characteristics.
# Checks for library functions.
# Output files.
AC_CONFIG_HEADERS([config.h])
AC_CONFIG_FILES([FILES])
AC_OUTPUT
```


The Structure of a *configure.ac*

configure.ac

```
# Prelude.
AC_INIT([amhello], [1.0], [bug-report@address])
AM_INIT_AUTOMAKE([foreign -Wall -Werror])
# Checks for programs.
AC_PROG_CC
# Checks for libraries.
# Checks for header files.
# Checks for typedefs, structures, and compiler characteristics.
# Checks for library functions.
# Output files.
AC_CONFIG_HEADERS([config.h])
AC_CONFIG_FILES([Makefile src/Makefile])
AC_OUTPUT
```


Useful Autoconf Macros for Prelude

`AC_INIT(PACKAGE, VERSION, BUG-REPORT-ADDRESS)`
: Mandatory Autoconf initialization.

`AC_PREREQ(VERSION)`
: Require a minimum Autoconf version. E.g. `AC_PREREQ([2.65])`

`AC_CONFIG_SRCDIR(FILE)`
: A safety check. FILE should be a distributed source file, and this makes sure that 'configure' is not run from outer space. E.g. `AC_CONFIG_SRCDIR([src/main.c])`.

`AC_CONFIG_AUX_DIR(DIRECTORY)`
: Auxiliary scripts such as *install-sh* and *depcomp* should be in DIRECTORY. E.g. `AC_CONFIG_AUX_DIR([build-aux])`.

AC_CONFIG_AUX_DIR Example

configure.ac

```
AC_INIT([amhello], [1.1], [bug-report@address])
AC_CONFIG_AUX_DIR([build-aux])
AM_INIT_AUTOMAKE([foreign -Wall -Werror])
AC_PROG_CC
AC_CONFIG_HEADERS([config.h])
AC_CONFIG_FILES([Makefile src/Makefile])
AC_OUTPUT
```

```
% autoreconf --install
configure.ac:3: installing 'build-aux/missing'
configure.ac:3: installing 'build-aux/install-sh'
src/Makefile.am: installing 'build-aux/depcomp'
```

Useful Program Checks

AC_PROG_CC, AC_PROG_CXX, AC_PROG_F77, ...
: Compiler checks. (Handle search cross-compilers if needed.)

AC_PROG_SED, AC_PROG_YACC, AC_PROG_LEX, ...
: Find good implementations and set $SED, $YACC, $LEX, etc.

AC_CHECK_PROGS(VAR, PROGS, [VAL-IF-NOT-FOUND])
: Define VAR to the first PROGS found, or to VAL-IF-NOT-FOUND otherwise.

```
AC_CHECK_PROGS([TAR], [tar gtar], [:])
if test "$TAR" = :; then
  AC_MSG_ERROR([This package needs tar.])
fi
```

... and many more

Useful Autoconf Action Macros

AC_MSG_ERROR(ERROR-DESCRIPTION, [EXIT-STATUS])
: Print ERROR-DESCRIPTION (also to *config.log*) and abort 'configure'.

AC_MSG_WARN(ERROR-DESCRIPTION)
: Likewise, but don't abort.

AC_DEFINE(VARIABLE, VALUE, DESCRIPTION)
: Output the following to *config.h*.

```
/* DESCRIPTION */
#define VARIABLE VALUE
```

AC_SUBST(VARIABLE, [VALUE])
: Define $(VARIABLE) as VALUE in *Makefile*.

```
AC_SUBST([FOO], [foo])
```

```
FOO=foo
AC_SUBST([FOO])
```

```
AC_SUBST([FOO])
FOO=foo
```

All equivalent.

Checking for Libraries

AC_CHECK_LIB(LIBRARY, FUNCT, [ACT-IF-FOUND], [ACT-IF-NOT])

Check whether LIBRARY exists and contains FUNCT.
Execute ACT-IF-FOUND if it does, ACT-IF-NOT otherwise.

```
AC_CHECK_LIB([efence], [malloc], [EFENCELIB=-lefence])
AC_SUBST([EFENCELIB])
```

... we would later use $(EFENCELIB) in the link rule.

If ACT-IF-FOUND is not set and the library is found, AC_CHECK_LIB will do LIBS="-lLIBRARY $LIBS" and #define HAVE_LIBLIBRARY. (Automake uses $LIBS for linking everything.)

Checking for Headers

AC_CHECK_HEADERS(HEADERS...)
: Check for HEADERS and #define HAVE_HEADER_H for each header found.

```
AC_CHECK_HEADERS([sys/param.h unistd.h])
AC_CHECK_HEADERS([wchar.h])
```

Might #define HAVE_SYS_PARAM_H, HAVE_UNISTD_H, and HAVE_WCHAR_H.

```
#if HAVE_UNISTD_H
# include <unistd.h>
#endif
```

AC_CHECK_HEADER(HEADER, [ACT-IF-FOUND], [ACT-IF-NOT])
: Check only one header.

Output Commands

AC_CONFIG_HEADERS(HEADERS...)

Create HEADER for all HEADER.*in*. Use only one such header unless you know what you are doing ('autoheader' creates HEADER.*in* only for the first HEADER).
HEADERS contain definitions made with AC_DEFINE.

```
AC_CONFIG_HEADERS([config.h])
```

Will create *config.h* from *config.h.in* (DJGPP supports only 1 dot).

AC_CONFIG_FILES(FILES...)

Create FILE for all FILE.*in*.
FILES contain definitions made with AC_SUBST.

```
AC_CONFIG_FILES([Makefile sub/Makefile script.sh:script.in])
```

Automake creates FILE.*in* for each FILE that has a FILE.*am*.
It's legitimate to process non-*Makefile*s too.

Output Commands

AC_CONFIG_HEADERS(HEADERS...)

Create HEADER for all HEADER.*in*. Use only one such header unless you know what you are doing ('autoheader' creates HEADER.*in* only for the first HEADER).
HEADERS contain definitions made with AC_DEFINE.

```
AC_CONFIG_HEADERS([config.h:config.hin])
```

Will create *config.h* from *config.hin* (DJGPP supports only 1 dot).

AC_CONFIG_FILES(FILES...)

Create FILE for all FILE.*in*.
FILES contain definitions made with AC_SUBST.

```
AC_CONFIG_FILES([Makefile sub/Makefile script.sh:script.in])
```

Automake creates FILE.*in* for each FILE that has a FILE.*am*.
It's legitimate to process non-*Makefile*s too.

AC_CONFIG_FILES([script.sh:script.in]) Example

script.in

```
#!/bin/sh
SED='@SED@'
TAR='@TAR@'
d=$1; shift; mkdir "$d"
for f; do
  "$SED" 's/#.*//' "$f" \
    >"$d/$f"
done
"$TAR" cf "$d.tar" "$d"
```

script.sh

```
#!/bin/sh
SED='/usr/xpg4/bin/sed'
TAR='/usr/bin/tar'
d=$1; shift; mkdir "$d"
for f; do
  "$SED" 's/#.*//' "$f" \
    >"$d/$f"
done
"$TAR" cf "$d.tar" "$d"
```

.in files are templates where @XYZ@ are placeholders for AC_SUBST([XYZ]) definitions. 'config.status' substitutes them.

*Makefile.in*s also use @XYZ@ as placeholders but Automake makes all XYZ=@XYZ@ definitions and you may simply use $(XYZ) as needed.

Using Automake

Automake Principles

- Automake helps creating portable and GNU-standard compliant *Makefile*s.
  - You may be used to other kinds of build systems. (E.g., no VPATH builds, but all objects go into *obj/*.)
  - Do not use Automake if you do not like the GNU Build System: Automake will get in your way if you don't fit the mold.
- 'automake' creates complex *Makefile.in*s from simple *Makefile.am*s.
  - Consider *Makefile.in*s as internal details.
- *Makefile.am*s follow roughly the same syntax as *Makefile*s however they usually contains only variable definitions.
  - 'automake' creates build rules from these definitions.
  - It's OK to add extra *Makefile* rules in *Makefile.am*: 'automake' will preserve them in the output.

Declaring Automake in *configure.ac*

`AM_INIT_AUTOMAKE([OPTIONS...])`

Check for tools needed by 'automake'-generated *Makefile*s.
Useful options:

`-Wall`	Turn all warnings on.
`-Werror`	Report warnings as errors.
`foreign`	Relax some GNU standard requirements.
`1.11.1`	Require a minimum version of 'automake'.
`dist-bzip2`	Also create `tar.bz2` archives during 'make dist' and 'make distcheck'.
`tar-ustar`	Create tar archives using the ustar format.

`AC_CONFIG_FILES(FILES...)`

Automake creates FILE.*in* for each FILE that has a FILE.*am*.

```
AC_CONFIG_FILES([Makefile sub/Makefile])
```

... and write *Makefile.am* and *sub/Makefile.am*.

`where_PRIMARY` Convention for Declaring Targets

Makefile.am

```
option_where_PRIMARY = targets ...
```

targets should be installed in...

`bin_`	`$(bindir)`
`lib_`	`$(libdir)`
...	
`custom_`	`$(`*`customdir`*`)` You define `customdir`.
`noinst_`	Not installed.
`check_`	Built by 'make check'.

targets should be built as...

- `_PROGRAMS`
- `_LIBRARIES`
- `_LTLIBRARIES` (Libtool libraries)
- `_HEADERS`
- `_SCRIPTS`
- `_DATA`

Optionally:
- `dist_` Distribute targets (if not the default)
- `nodist_` Don't.

Declaring Sources

Makefile.am

```
bin_PROGRAMS = foo run-me
foo_SOURCES = foo.c foo.h print.c print.h
run_me_SOURCES = run.c run.h print.c
```

- These programs will be installed in `$(bindir)`.
- The sources of each program go into program`_SOURCES`.
- Non-alphanumeric characters are mapped to '_'.
- Automake automatically computes the list of objects to build and link from these files.
- Header files are not compiled. We list them only so they get distributed (Automake does not distribute files it does not know about).
- It's OK to use the same source for two programs.
- Compiler and linker are inferred from the extensions.

(Static) Libraries

- Add AC_PROG_RANLIB to *configure.ac*.

Makefile.am

```
lib_LIBRARIES = libfoo.a libbar.a
libfoo_a_SOURCES = foo.c privfoo.h
libbar_a_SOURCES = bar.c privbar.h
include_HEADERS = foo.h bar.h
```

- These libraries will be installed in $(libdir).
- Library names must match lib*.a.
- Public headers will be installed in $(includedir).
- Private headers are not installed, like ordinary source files.

Directory Layout

- You may have one *Makefile* (hence one *Makefile.am*) per directory.
- They must all be declared in *configure.ac*.

configure.ac

```
AC_CONFIG_FILES([Makefile lib/Makefile src/Makefile
                 src/dira/Makefile src/dirb/Makefile])
```

- 'make' is run at the top-level.
- *Makefile.am*s should fix the order in which to recurse directories using the SUBDIRS variable.

Makefile.am

```
SUBDIRS = lib src
```

src/Makefile.am

```
SUBDIRS = dira dirb
```

- The current directory is implicitly built after subdirectories.
- You can put '.' where you want to override this.

Directory Layout

- You may have one *Makefile* (hence one *Makefile.am*) per directory.
- They must all be declared in *configure.ac*.

configure.ac

```
AC_CONFIG_FILES([Makefile lib/Makefile src/Makefile
                 src/dira/Makefile src/dirb/Makefile])
```

- 'make' is run at the top-level.
- *Makefile.am*s should fix the order in which to recurse directories using the SUBDIRS variable.

Makefile.am

```
SUBDIRS = lib src
```

src/Makefile.am

```
SUBDIRS = dira . dirb
```

- The current directory is implicitly built after subdirectories.
- You can put '.' where you want to override this.

$(srcdir) and VPATH Builds

- Remember VPATH builds: a source file is not necessary in the current directory.
- There are two twin trees: the **build tree**, and the **source tree**.
  - *Makefile* and objects files are in the build tree.
  - *Makefile.in*, *Makefile.am*, and source files are in the source tree.
  - If './configure' is run in the current directory, the two trees are one.
- In each *Makefile*, 'config.status' will define $(srcdir): the path to the matching source directory.
- When referring to sources files or targets in Automake variables, you do not have to worry about *source* vs. *build*, because 'make' will check both directories.
- You may need $(srcdir) when specifying flags for tools, or writing custom commands. E.g., to tell the compiler to include headers from *dir/*, you should write -I$(srcdir)/dir, not -Idir. (-Idir would fetch headers from the build tree.)

Convenience Libraries

lib/Makefile.am

```
noinst_LIBRARIES = libcompat.a
libcompat_a_SOURCES = xalloc.c xalloc.h
```

- This is a convenience library, used only when building the package.

src/Makefile.am

```
LDADD = ../lib/libcompat.a
AM_CPPFLAGS = -I$(srcdir)/../lib
bin_PROGRAMS = foo run-me
foo_SOURCES = foo.c foo.h print.c print.h
run_me_SOURCES = run.c run.h print.c
```

- LDADD is added when linking all programs.
- AM_CPPFLAGS contains additional preprocessor flags.

Convenience Libraries

lib/Makefile.am

```
noinst_LIBRARIES = libcompat.a
libcompat_a_SOURCES = xalloc.c xalloc.h
```

- This is a convenience library, used only when building the package.

src/Makefile.am

```
bin_PROGRAMS = foo run-me
foo_SOURCES = foo.c foo.h print.c print.h
run_me_SOURCES = run.c run.h print.c
run_me_LDADD = ../lib/libcompat.a
run_me_CPPFLAGS = -I$(srcdir)/../lib
```

- LDADD is added when linking all programs.
- AM_CPPFLAGS contains additional preprocessor flags.
- You can use per-target variables: they apply to a single program.

Per-Target Flags

Assuming foo is a program or library:

`foo_CFLAGS`	Additional C compiler flags
`foo_CPPFLAGS`	Additional preprocessor flags (`-Is` and `-Ds`)
`foo_LDADD`	Additional link objects, `-ls` and `-Ls` (if foo is a program)
`foo_LIBADD`	Additional link objects, `-ls` and `-Ls` (if foo is a library)
`foo_LDFLAGS`	Additional linker flags

The default value for `foo_XXXFLAGS` is `$(AM_XXXFLAGS)`.
Use plain file names to refer to libraries inside your package (keep `-ls` and `-Ls` for external libraries only).

src/Makefile.am

```
bin_PROGRAMS = foo run-me
foo_SOURCES = foo.c foo.h print.c print.h
run_me_SOURCES = run.c run.h print.c
run_me_CPPFLAGS = -I$(srcdir)/../lib
run_me_LDADD = ../lib/libcompat.a $(EFENCELIB)
```


What Gets Distributed

'`make dist`' and '`make distcheck`' create a tarball containing:

- All sources declared using ..._SOURCES
- All headers declared using ..._HEADERS
- All scripts declared with dist_..._SCRIPTS
- All data files declared with dist_..._DATA
- ...
- Common files such as *ChangeLog*, *NEWS*, etc.
  See '`automake --help`' for a list of those files.
- Extra files or directories listed into EXTRA_DIST.

Makefile.am

```
SUBDIRS = lib src
EXTRA_DIST = HACKING
```

... will additionally distribute *HACKING*.

Conditionals: Usage

- *Conditionals* allow for conditional builds and unconditional distribution.

Conditional Programs

```
bin_PROGRAMS = foo
if WANT_BAR
  bin_PROGRAMS += bar
endif
foo_SOURCES = foo.c
bar_SOURCES = bar.c
```

Conditional Sources

```
bin_PROGRAMS = foo
foo_SOURCES = foo.c
if WANT_BAR
  foo_SOURCES += bar.c
endif
```

- *bar* is built iff WANT_BAR is true.
- *bar.o* is linked in *foo* iff WANT_BAR is true.
- In all cases *foo.c* and *bar.c* are distributed regardless of WANT_BAR.
- This is portable. 'config.status' will comment rules of *Makefile.in* that must be disabled.
- WANT_BAR must be declared and valued in *configure.ac*.

Conditionals: Declaration

`AM_CONDITIONAL(NAME, CONDITION)`

Declare conditional NAME. CONDITION should be a shell instruction that succeeds iff NAME should be enabled.

configure.ac

```
AC_CHECK_HEADER([bar.h], [use_bar=yes])
AM_CONDITIONAL([WANT_BAR], [test "$use_bar" = yes])
```

Will enable WANT_BAR only if *bar.h* is present on the system.

Extending Automake Rules

- The contents of *Makefile.am* are copied almost verbatim to *Makefile.in*.
- 'automake' adds new rules and variables in *Makefile.in*, to achieve the semantics of the special variables you have defined.
- Some minor rewriting is done to handle constructs like conditionals or += portably.
- It's OK to define your own rules in *Makefile.am*.
  - Helpful maintenance targets ('make style-check')
  - Build idiosyncratic files (generate a *FAQ* from some random source)
  - ...
- It's OK to define variables that are meaningless to Automake.
  - For use in custom rules.
- Beware of conflicts: your definitions (of variables or rules) will override those of Automake.
  - -Wall will diagnose these.

Recommendations

- Use -Wall -Werror.
- Keep Your Setup Simple (KYSS!).
  - You will spend a large part of time debugging your cunning tricks if you try to automatize too much.
- Do not lie to Automake.
  - Automake can be annoying, but when you lie it gets worse!

Lost? 'autoreconf' is Still Your Friend

If 'make' fails to rebuild configuration files, run 'autoreconf' manually.

```
~/amhello % autoreconf --install
```

If this does not help, try harder.

```
~/amhello % autoreconf --install --force
```

If this still does not help, try even harder.

```
~/amhello % make -k maintainer-clean
~/amhello % autoreconf --install --force
```

Do this only when necessary. Each of these commands will cause your package to take longer to reconfigure and recompile.

Part III

More Autotools

Writing Autoconf Macros

Writing an Autoconf Macro? Why? How?

Two fundamentally different types of new macros:

- Macros that factor related tests in a single reusable entity.
  - High-level.
  - Combination of existing lower-level macros.
  - May not use shell code at all.
- Macros that implements new tests.
  - Low-level.
  - Actually code the check.
  - Need to bother with caching values.

Defining Macros

AC_DEFUN(MACRO-NAME, MACRO-BODY)
Define MACRO-NAME as MACRO-BODY.

Avoid names that may conflict. Macro name spaces:

- m4_ Original M4 macros, plus M4sugar macros.
- AS_ M4sh macros (macroized shell constructs)
- AH_ **A**uto**h**eader macros
- AC_ **A**uto**c**onf macros (written on top of the above layers)
  - AC_CHECK_ Generic checks.
  - AC_FUNC_ Specific function checks.
  - AC_HEADER_ Specific header checks.
  - AC_PROG_ Specific program checks.
  - ...
- AM_ **A**uto**m**ake macros
- AT_ **A**uto**t**est macros

mkdir() Example

- POSIX systems define mkdir() with two arguments.
- On Mingw32 (at least), mkdir() takes only one argument.
- On Win32 (at least), the name is _mkdir() with one argument.

```
#if HAVE_MKDIR
# if MKDIR_ONE_ARG
#  define mkdir(a,b) mkdir(a)
# endif
#else
# if HAVE__MKDIR
#  define mkdir(a,b) _mkdir(a)
# else
#  error "Don't know how to create a directory."
# endif
#endif
```

Let's write an Autoconf macro to define these C macros.

Writing a High-Level Macro: AX_FUNC_MKDIR

```
AC_DEFUN([AX_FUNC_MKDIR],
[AC_CHECK_FUNCS([mkdir _mkdir])
AC_CHECK_HEADERS([io.h])
AX_FUNC_MKDIR_ONE_ARG
])
```

- Suggested name space for extension macros.
- Use same convention as Autoconf for categorizing macros.
- Defines HAVE_MKDIR and HAVE__MKDIR.
- Defines HAVE_IO_H if *io.h* exists.
  (mkdir() may also be defined there, and *sys/stat.h* and *unistd.h* are always tested by AC_PROG_CC)
- Will define MKDIR_ONE_ARG... once written.

Checking mkdir()'s number of arguments

```
# _AX_FUNC_MKDIR_ONE_ARG(IF-ONE-ARG, IF-TWO-ARGS)
# -----------------------------------------------
# Execute IF-TWO-ARGS if mkdir() accepts two
# arguments; execute IF-ONE-ARG otherwise.
AC_DEFUN([_AX_FUNC_MKDIR_ONE_ARG],
[AC_COMPILE_IFELSE([AC_LANG_PROGRAM([[
#include <sys/stat.h>
#if HAVE_UNISTD_H
# include <unistd.h>
#endif
#if HAVE_IO_H
# include <io.h>
#endif
]], [[mkdir (".", 0700);]])], [$2], [$1])])
```

AC_COMPILE_IFELSE
Creates a small program and attempt to compile it. In our case it will execute one of _AX_FUNC_MKDIR_ONE_ARG's arguments depending on whether compilation succeeded.

- Wait! That's not enough for an Autoconf check: we should also add some *checking whether...* message on top of this.
- We use the _AX prefix for helper macros not meant to be used directly.

Writing a Low-Level Macro

Low-level macros need to

- print a *checking whether...* message
- do the actual check
- cache the result of the check

Most of this is achieved via the AC_CACHE_CHECK macro.

```
AC_DEFUN(MACRO-NAME,
[AC_CACHE_CHECK(WHETHER-MESSAGE,
                CACHE-VARIABLE,
                CODE-TO-SET-CACHE-VARIABLE)
CODE-USING-CACHE-VARIABLE])
```

- The CACHE-VARIABLE should match *_cv_*.
- CODE-TO-SET-CACHE-VARIABLE should contain the check. It will be skipped when the cache is used.
- CODE-USING-CACHE-VARIABLE is always executed, use AC_SUBST and AC_DEFINE here.

A Low-Level Macro: AX_FUNC_MKDIR_ONE_ARG

```
AC_DEFUN([AX_FUNC_MKDIR_ONE_ARG],
[AC_CACHE_CHECK([whether mkdir takes one argument],
                [ax_cv_mkdir_one_arg],
[_AX_FUNC_MKDIR_ONE_ARG([ax_cv_mkdir_one_arg=yes],
                        [ax_cv_mkdir_one_arg=no])])
if test x"$ax_cv_mkdir_one_arg" = xyes; then
  AC_DEFINE([MKDIR_ONE_ARG], 1,
            [Define if mkdir takes only one argument.])
fi]) # AX_FUNC_MKDIR_ONE_ARG
```

- AC_CACHE_CHECK
  - prints *checking whether mkdir...*
  - does the check (unless already done)
  - cache the result in ax_cv_mkdir_one_arg
- Keep configuration actions outside AC_CACHE_CHECK: they have to be executed whether the check is run or cached.

Recommendations for Writing Autoconf Macros

- Test for features, not for systems.
  - E.g., check whether mkdir() takes one argument, not whether you are compiling for Win32.
  - Your package will be more likely to adapt to untested systems.
- Avoid writing tests that are conditional on previous tests.
  - Have unconditional tests, with conditional actions.
  - E.g., check for _mkdir() even if mkdir() exists.
- Do not reinvent the wheel.
  - Autoconf comes with a lot of well-tested macros. Use them.
- Remember to [quote].
- Read the *Portable Shell* section of the Autoconf manual, before writing shell code.
- Test your macros on different systems.
  - Check test results in *config.log*.
  - Get accounts on foreign systems (Google for "free shell account").

Managing Custom Macros with 'aclocal'

aclocal.m4 and Third-Party Macros

- 'autoconf' knows only the macros it provides. (m4_*, AS_*, AH_*, AC_*, AT_*).
- 'autoconf' knows nothing about macro supplied by third-party tools (e.g., Automake's AM_* macros).
- 'autoconf' reads *aclocal.m4* in addition to *configure.ac*.
- *aclocal.m4* should define the extra macros required by *configure.ac*.
- 'aclocal' automates the construction of *aclocal.m4* from various sources.

'aclocal' searches macros in

- directories specified with -I options
- a system-wide directory (usually */usr/share/aclocal/*) where third-party packages may install their macros
- Automake's own private macro directory

Managing Custom Macros in Your Package

- Create a *m4/* subdirectory.
- Put your macros there.
  E.g., define AX_FUNC_MKDIR and AX_FUNC_MKDIR_ONE_ARG in *m4/mkdir.m4*.
  (The extension *must* be **.m4*)
- Add `ACLOCAL_AMFLAGS = -I m4` to the top-level *Makefile.am*.
- Add `AC_CONFIG_MACRO_DIR([m4])` to *configure.ac*.
- Use your macros in *configure.ac*.

The ACLOCAL_AMFLAGS are used by 'autoreconf' and by the *Makefile* rebuild rule when they need to run 'aclocal'.
Local macros that are used are automatically distributed. (Those that are not used are simply ignored.)
You need such a setup to use Gettext, and Libtool.

Libtool

Shared Libraries: A Portability Hell

- Almost each system has its own format of shared library
  - *libhello.so*
  - *libhello.dll*
  - *libhello.sl*
  - *libhello.dylib*
  - ...
- Building will require different flags
  - -fPIC, -shared
  - -KPIC, -G
  - -bM:SRE
  - ...
- Linking against the library may also require specific flags.
- There is no way for a developer to keep track of all these details.
  - Quiz: match each of the above example with its OS.
- Not all systems support shared libraries.

Shared Libraries: Libtool's Solution

- A new library format that abstracts all the others
  - *libhello.la* (**l**ibtool **a**rchive)
- A wrapper script for the compiler and linker
  - translates operations involving *libhello.la* into the correct operation for the current system using the real library
- In a *Makefile.am*, you simply create and link against **.la* files.
- These operations are translated appropriately.

Setting Up Libtool: Roadmap

- Libtool will require some local Autoconf macros for all the checks it has to perform. Use an *m4/* subdirectory as explained earlier.
- Call `LT_INIT` in *configure.ac*.
- Use `_LTLIBRARIES` to declare libtool archives in *Makefile.am*
- Use `_LDADD` to link against local libtool archives.

Makefile.am

```
lib_LTLIBRARIES = libfoo.la
libfoo_la_SOURCES = foo.c foo.h etc.c

bin_PROGRAMS = runme
runme_SOURCES = main.c
runme_LDADD = libfoo.la
```


Hello World Using Libtool: C Files

lib/say.c

```
#include <config.h>
#include <stdio.h>

void say_hello (void)
{
  puts ("Hello World!");
  puts ("This is " PACKAGE_STRING ".");
}
```

src/main.c

```
#include "say.h"

int main (void)
{
  say_hello ();
  return 0;
}
```

lib/say.h

```
void say_hello (void);
```


Hello World Using Libtool: *Makefile.ams*

lib/Makefile.am

```
lib_LTLIBRARIES = libhello.la
libhello_la_SOURCES = say.c say.h
```

src/Makefile.am

```
AM_CPPFLAGS = -I$(srcdir)/../lib
bin_PROGRAMS = hello
hello_SOURCES = main.c
hello_LDADD = ../lib/libhello.la
```

Makefile.am

```
SUBDIRS = lib src
ACLOCAL_AMFLAGS = -I m4
```

Hello World Using Libtool: *configure.ac*

configure.ac

```
AC_INIT([amhello], [2.0], [bug-report@address])
AC_CONFIG_AUX_DIR([build-aux])
AC_CONFIG_MACRO_DIR([m4])
AM_INIT_AUTOMAKE([foreign -Wall -Werror])
LT_INIT
AC_PROG_CC
AC_CONFIG_HEADERS([config.h])
AC_CONFIG_FILES([Makefile lib/Makefile src/Makefile])
AC_OUTPUT
```


Hello World Using Libtool: ‘autoreconf’

```
~/amhello % ls -R
.:
Makefile.am  configure.ac  lib/  src/

./lib:
Makefile.am  say.c  say.h

./src:
Makefile.am  main.c
```


Hello World Using Libtool: ‘autoreconf’

```
~/amhello % mkdir m4
~/amhello % autoreconf --install
libtoolize: putting auxiliary files in AC_CONFIG_AUX_DIR, ‘bui
libtoolize: copying file ‘build-aux/ltmain.sh’
libtoolize: putting macros in AC_CONFIG_MACRO_DIR, ‘m4’.
libtoolize: copying file ‘m4/libtool.m4’
libtoolize: copying file ‘m4/ltoptions.m4’
libtoolize: copying file ‘m4/ltsugar.m4’
libtoolize: copying file ‘m4/ltversion.m4’
libtoolize: copying file ‘m4/lt~obsolete.m4’
configure.ac:5: installing ‘build-aux/config.guess’
configure.ac:5: installing ‘build-aux/config.sub’
configure.ac:4: installing ‘build-aux/install-sh’
configure.ac:4: installing ‘build-aux/missing’
lib/Makefile.am: installing ‘build-aux/depcomp’...
~/amhello % ./configure --prefix ~/test
```


Hello World Using Libtool: ‘autoreconf’

```
~/amhello % mkdir m4
~/amhello % autoreconf --install
...
~/amhello % ./configure --prefix ~/test
...
~/amhello % make && make install
...
~/amhello % ~/test/bin/hello
Hello World!
This is amhello 2.0.
~/amhello %
```

What Was Built and Installed

```
~/amhello % ls -R ~/test
/home/adl/test:
bin/  lib/
/home/adl/test/bin:
hello*
/home/adl/test/lib:
libhello.a    libhello.so@    libhello.so.0.0.0*
libhello.la*  libhello.so.0@
~/amhello % ldd ~/test/bin/hello
libhello.so.0 => /home/adl/test/lib/libhello.so.0 (0xb7fe7000)
libc.so.6 => /lib/tls/libc.so.6 (0xb7e9c000)
lib/ld-linux.so.2 => /lib/ld-linux.so.2 (0xb7fea000)
~/amhello % ldd src/hello
not a dynamic executable
~/amhello % file src/hello
src/hello: Bourne shell script text executable
```


Building Shared or Static Libraries

- By default, both static and shared libraries are built.
- This default can be changed in a package using options passed to `LT_INIT(options...)`:
  - `disable-shared` do not build shared libraries by default
  - `disable-static` do not build static libraries by default
- The installer can override these settings using *configure* options.
  - `--enable-shared` build shared libraries
  - `--disable-shared` don't
  - `--enable-static` build static libraries
  - `--disable-static` don't
- At least one flavor is built, always.
- Some systems don't leave any choice.

The *src/hello* Wrapper Script

- *src/hello* can be a wrapper script
  - Depending on Libtool's configuration.
- The real binary has been built elsewhere
  - Libtool hides it in the build tree (don't bother about it)
- This wrapper script runs the real binary, and arranges so it finds the not-yet-installed libraries
  - This way *src/hello* can be run, for instance in a test suite

Do not debug the shell script!

```
~/amhello % gdb -q src/hello
"src/hello": not in executable format: File format not recognized
(gdb)
```

Prefix such commands with `libtool --mode=execute`

```
~/amhello % libtool --mode=execute gdb -q src/hello
```


Versioning Libtool Libraries: Interfaces

- Versioning libraries allow several versions to coexist.
- It ensures programs use the library that implements the interface they require.

Interface = public variables and functions, I/O, formats, protocols, ...

- Interfaces are identified using integers.
- A program remembers the interface numbers of the libraries it was linked against.
- A library can implement several interfaces.
  - E.g., adding new functions changes the interface, but does not break old interfaces.
- Hence libtool's versioning format encodes a range of supported interfaces.

Interface numbers are not release numbers.

Versioning Libtool Libraries: Version Triplets

CURRENT The latest interface implemented.

REVISION The implementation number of CURRENT (read: number of bugs fixed...)

AGE The number of interfaces implemented, minus one. The library supports all interfaces between CURRENT − AGE and CURRENT.

These numbers should be specified using `-version-info`.

lib/Makefile.am

```
lib_LTLIBRARIES = libhello.la
libhello_la_SOURCES = say.c say.h
libhello_la_LDFLAGS = -version-info CURRENT:REVISION:AGE
```

The default version is 0:0:0. It's also a good initial version.

Versioning Libtool Libraries: Bumping Versions

Remember to bump library versions before a release.
Suppose the old version was CURRENT:REVISION:AGE.

If you have	bump the version to
not changed the interface (bug fixes)	CURRENT : REVISION+1 : AGE
augmented the interface (new functions)	CURRENT+1 : 0 : AGE+1
broken old interface (e.g. removed functions)	CURRENT+1 : 0 : 0

Introducing Gettext

Introducing Gettext

- Internationalization = I18n
  Changing a program to support for multiple languages and cultural habits.
  - Character handling (unicode...)
  - Locale awareness (date formats, currencies, numbers, time zones, etc.)
  - Localizability
    - Isolate localizable items (messages, pictures, etc.)
    - Implement infrastructure necessary for localizing above items.

  The programmer's work.
- Localization = L10n
  Providing an internationalized package the necessary bits to support one's native language and cultural habits.
  - Translate localizable items (messages, pictures, etc.) for one language.

  The translator's work.

Gettext = complete toolset for translating messages output by programs.

Translating Messages Made Easy

```
#include <config.h>
#include <stdio.h>
#include "gettext.h"
#define _(string) gettext (string)
void say_hello (void)
{
  puts ("Hello World!");
  puts ("This is " PACKAGE_STRING ".");
}
```

- The program is written in English.
- Messages that must be translated are marked with `_(...)`.
  - 'xgettext' builds catalogs of translatable messages from such strings.
  - Translators will provide translated catalogs for their locale.
- `gettext` looks up the translation of the English message in the current locale's catalog.

Translating Messages Made Easy

```
#include <config.h>
#include <stdio.h>
#include "gettext.h"
#define _(string) gettext (string)
void say_hello (void)
{
  puts (_("Hello World!"));
  printf (_("This is %s.\n"), PACKAGE_STRING);
}
```

- The program is written in English.
- Messages that must be translated are marked with `_(...)`.
  - 'xgettext' builds catalogs of translatable messages from such strings.
  - Translators will provide translated catalogs for their locale.
- `gettext` looks up the translation of the English message in the current locale's catalog.

Internationalizing a Package, Start to Finish

Internationalizing a Package, Start to Finish

Roadmap:

1. Start with a non-internationalized Hello World.
2. Invoke `AM_GNU_GETTEXT` from *configure.ac*
3. Run 'gettextize' to provide the basic infrastructure.
4. Fill in the configuration files left by 'gettextize'.
5. Update *src/Makefile.am* to link *hello* with the necessary library.
6. Update the code:
   - Initialize Gettext in `main()`
   - Mark translatable strings.
7. Generate messages catalogs automatically.

We'll talk about localization once this is done.

Non Internationalized Hello World (1/2)

src/main.c

```
#include "say.h"

int
main (void)
{
  say_hello ();
  return 0;
}
```

src/say.h

```
#ifndef AMHELLO_SAY_H
# define AMHELLO_SAY_H
void say_hello (void);
#endif
```

src/say.c

```
#include <config.h>
#include <stdio.h>

void say_hello (void)
{
  puts ("Hello World!");
  puts ("This is " PACKAGE_STRING ".");
}
```


Non Internationalized Hello World (2/2)

configure.ac

```
AC_INIT([amhello], [3.0], [bug-report@address])
AC_CONFIG_AUX_DIR([build-aux])
AM_INIT_AUTOMAKE([foreign -Wall -Werror])
AC_PROG_CC
AC_CONFIG_HEADERS([config.h])
AC_CONFIG_FILES([Makefile src/Makefile])
AC_OUTPUT
```

Makefile.am

```
SUBDIRS = src
```

src/Makefile.am

```
bin_PROGRAMS = hello
hello_SOURCES = main.c say.c say.h
```


Update *configure.ac* for Gettext

configure.ac

```
AC_INIT([amhello], [3.0], [bug-report@address])
AC_CONFIG_AUX_DIR([build-aux])
AM_INIT_AUTOMAKE([foreign -Wall -Werror])
AM_GNU_GETTEXT_VERSION([0.17])
AM_GNU_GETTEXT([external])
AC_PROG_CC
AC_CONFIG_HEADERS([config.h])
AC_CONFIG_FILES([Makefile src/Makefile])
AC_OUTPUT
```

- `AM_GNU_GETTEXT_VERSION` = *exactly* which Gettext version to use.
- `AM_GNU_GETTEXT([external])`
  - the GNU libc or an external (= not distributed) Gettext library will be used if found
  - NLS (Native Language System) will be disabled otherwise

Running 'gettextize'

You should run 'gettextize':

- A first time, to install the Gettext infrastructure in your package.
- Each time you upgrade Gettext to a new version.

```
~/amhello % gettextize --copy --no-changelog
[...]
~/amhello % cp /usr/share/gettext/gettext.h src
```

- Install most of the Gettext infrastructure.
- Copy *gettext.h* in the source tree, it will be distributed.

Gettextize Updated Some Files

configure.ac

```
AC_INIT([amhello], [3.0], [bug-report@address])
AC_CONFIG_AUX_DIR([build-aux])
AM_GNU_GETTEXT_VERSION([0.17])
AM_GNU_GETTEXT([external])
AM_INIT_AUTOMAKE([foreign -Wall -Werror])
AC_PROG_CC
AC_CONFIG_HEADERS([config.h])
AC_CONFIG_FILES([Makefile src/Makefile po/Makefile.in])
AC_OUTPUT
```

Makefile.am

```
SUBDIRS = po src
ACLOCAL_AMFLAGS = -I m4
EXTRA_DIST = ...
```

src/Makefile.am

```
bin_PROGRAMS = hello
hello_SOURCES = main.c say.c say.h
```


po/Makevars and *po/POTFILES.in*

Fill *po/Makevars.template* and rename it as *po/Makevars*:

po/Makevars

```
DOMAIN = $(PACKAGE)
subdir = po
top_builddir = ..
XGETTEXT_OPTIONS = --keyword=_ --keyword=N_
COPYRIGHT_HOLDER = Your Name or Your Employer
MSGID_BUGS_ADDRESS = $(PACKAGE_BUGREPORT)
EXTRA_LOCALE_CATEGORIES =
```

`$(PACKAGE_BUGREPORT)` is the third argument of `AC_INIT`. Some packages use a mailing list dedicated to translation issues instead.

List sources files that (may) contain translatable strings in *POTFILES.in*.

po/POTFILES.in

```
src/main.c
src/say.c
```


What's Next?

Done:

1. Start with a non-internationalized Hello World.
2. Invoke `AM_GNU_GETTEXT` from *configure.ac*
3. Run '`gettextize`' to provide the basic infrastructure.
4. Fill in the configuration files left by '`gettextize`'.

Now, '`autoreconf --install; ./configure; make`' should work.

To do:

5. Update *src/Makefile.am* to link *hello* with the necessary library.
6. Update the code:
   - Initialize Gettext in `main()`
   - Mark translatable strings.
7. Generate messages catalogs automatically.

Updating *src/Makefile.am*

src/Makefile.am

```
AM_CPPFLAGS = -DLOCALEDIR=\"$(localedir)\"
bin_PROGRAMS = hello
hello_SOURCES = main.c say.c say.h gettext.h
LDADD = $(LIBINTL)
```

- `$(LIBINTL)` lists the libraries any internationalized program should be linked against.
- We can strip the leading `hello_` and use the global `LDADD` instead.
- Mention *gettext.h* (we will use it shortly) so it is distributed.
- `$(LOCALEDIR)` is the place where message catalogs are installed. This is needed during initialization.

Initializing Gettext

src/main.c

```
#include <config.h>
#include <locale.h>
#include "gettext.h"
#include "say.h"
int
main (void)
{
  setlocale (LC_ALL, "");
  bindtextdomain (PACKAGE,
                  LOCALEDIR);
  textdomain (PACKAGE);
  say_hello();
  return 0;
}
```

- Initialize the locale as specified in the environment.
  (E.g., the user sets `LANG=fr_FR` in the environment to get French messages.)
- Tell Gettext where to find message catalogs for this program.
  (All programs in the same package usually share the same message catalog.)

Marking Strings for Translation

src/say.c

```
#include <config.h>
#include <stdio.h>
#include "gettext.h"
#define _(string) gettext (string)
void say_hello (void)
{
  puts (_("Hello World!"));
  printf (_("This is %s.\n"), PACKAGE_STRING);
}
```

- Messages that must be translated are marked with `_(...)`.
- NLS (Native Language System) can be disabled.
  - Explicitly with '`./configure --disable-nls`'
  - Implicitly if no `gettext` implementation is installed.

Then *gettext.h* defines `gettext()`, `textdomain()`, ..., as no-ops.

Building the Whole Shebang

Our Hello World is now internationalized.

```
~/amhello % autoreconf --install
...
~/amhello % ./configure
...
~/amhello % make
...
Making all in po
make amhello.pot-update
...
```

The *po/* directory contains messages catalogs.
po/amhello.pot is the template message catalog.

Updating *po/amhello.pot* is costly and occurs only before releases (e.g., during '`make distcheck`') or if the file did not exist (our case above).
It can be updated explicitly with '`cd po; make update-po`'.

Localizing a Package

po/amhello.pot: The PO Template File

```
# ... COMMENTS ...
#, fuzzy
msgid ""
msgstr ""
"Project-Id-Version: PACKAGE VERSION\n"
"Report-Msgid-Bugs-To: bug-report@address\n"
"POT-Creation-Date: 2005-03-05 00:27+0100\n"
"PO-Revision-Date: YEAR-MO-DA HO:MI+ZONE\n"
"Last-Translator: FULL NAME <EMAIL@ADDRESS>\n"
"Language-Team: LANGUAGE <LL@li.org>\n"
"MIME-Version: 1.0\n"
"Content-Type: text/plain; charset=CHARSET\n"
"Content-Transfer-Encoding: 8bit\n"

#: src/say.c:9
msgid "Hello World!"
msgstr ""

#: src/say.c:10
#, c-format
msgid "This is %s.\n"
msgstr ""
```


po/amhello.pot: List of Messages

```
#: src/say.c:9
msgid "Hello World!"
msgstr ""

#: src/say.c:10
#, c-format
msgid "This is %s.\n"
msgstr ""
```

- `msgid`s identify all strings in the package
- empty `msgstr`s are placeholders for translations
- the location of each string is shown,
  so the translator can check the context if needed
- additional flags can be used

po/amhello.pot: The Header Entry

```
msgid ""
msgstr ""
"Project-Id-Version: PACKAGE VERSION\n"
"Report-Msgid-Bugs-To: bug-report@address\n"
"POT-Creation-Date: 2005-03-05 00:27+0100\n"
"PO-Revision-Date: YEAR-MO-DA HO:MI+ZONE\n"
"Last-Translator: FULL NAME <EMAIL@ADDRESS>\n"
"Language-Team: LANGUAGE <LL@li.org>\n"
"MIME-Version: 1.0\n"
"Content-Type: text/plain; charset=CHARSET\n"
"Content-Transfer-Encoding: 8bit\n"
```

The translation of the empty string is a special entry that will be filled with administrative information.

How to Add a New Language?

1. Initialize *po/LL.po* or *po/LL_CC.po* from *po/amhello.pot*, using '`msginit`'.
   LL is your language code, and CC is your country code
   pt is Portuguese
   pt_BR is Brazilian Portuguese
   (The annexes of the Gettext manual show lists of LLs and CCs.)
2. Fill in *po/LL.po* (or *po/LL_CC.po*)
3. List the new translation in *po/LINGUAS*

Let's add a French translation for `amhello`.

Preparing *po/fr.po*

```
~/amhello % cd po
~/amhello/po % msginit -l fr
...
~/amhello/po % emacs fr.po &
```

The PO mode of 'emacs' (⟨M-x⟩ po-mode):

- The buffer is modified only indirectly.
- ⟨Enter⟩ on a message will open a buffer to edit the translation.
- Use ⟨C-c⟩⟨C-c⟩ after you have completed the translation, to get back to the updated *amhello.pot* buffer.
- Once all strings are translated, use ⟨V⟩ to save and check the file.
- Use ⟨Tab⟩ to remove fuzzy attributes.

po/fr.po: Message Translations

```
#: src/say.c:9
msgid "Hello World!"
msgstr "Bonjour Monde !"

#: src/say.c:10
#, c-format
msgid "This is %s.\n"
msgstr "Ceci est %s.\n"
```


po/fr.po: Header

```
msgid ""
msgstr ""
"Project-Id-Version: amhello 3.0\n"
"Report-Msgid-Bugs-To: bug-report@address\n"
"POT-Creation-Date: 2005-03-05 00:27+0100\n"
"PO-Revision-Date: 2005-03-15 20:54+0100\n"
"Last-Translator: Alexandre Duret-Lutz <adl@gnu.org>\n"
"Language-Team: French\n"
"MIME-Version: 1.0\n"
"Content-Type: text/plain; charset=iso-8859-1\n"
"Content-Transfer-Encoding: 8bit\n"
"Plural-Forms: nplurals=2; plural=(n > 1);\n"
```

- 'msginit' filled these fields.
- You may have to customize it a bit.
- The revision date will also be updated on save.

po/fr.po: Validation and Addition

Once *po/fr.po* is completed, hit ⟨V⟩. This will:

1. Update the revision date
2. Save the file
3. Run 'msgfmt --statistics --check' on *po/fr.po*, to validate it.

We can now register the language.

```
~/amhello/po % echo fr >> LINGUAS
```

`hello` now Speaks French!

```
~/amhello % ./configure --prefix ~/test
~/amhello % make
~/amhello % cd po
~/amhello/po % make update-po
~/amhello/po % cd ..
~/amhello % make install
~/amhello % ~/test/bin/hello
Hello World!
This is amhello 3.0.
~/amhello % LANG=fr_FR ~/test/bin/hello
Bonjour Monde !
Ceci est amhello 3.0.
```

update-po

This step is needed because we just created *fr.po*, and it has to be compiled. This happens automatically during '`make dist`'.

Updating Message Catalogs

Because maintainers can change the strings marked for translation, the messages catalogs are varying, and are not always up-to-date.

Varying messages. `update-po` modify **.po* file:

- New messages are added with a blank translation.
- Obsolete translations, not used anymore, are commented.
- Messages with tiny changes keep their translation, but are marked `fuzzy`.
  Translators remove `fuzzy` attributes (⟨Tab⟩) after verification.

Not up-to-date. `gettext` copes with incomplete translations as follows.

- Untranslated messages are output untranslated.
- Fuzzy messages are also output untranslated. (Better output the original sentence, rather than an inappropriate translation.)

Good practice: the string freeze. Two weeks before a release, run '`make update-po`' and send the **.pot* file to translators. Don't change or add strings from this point on. Let translators send you updated **.po* files.

Language Teams & The Translation Project

`http://www.iro.umontreal.ca/translation/`

The Translation Project provides an infrastructure for package maintainers and translators to exchange messages catalogs.

- Translators gather in *Language Teams* (consider joining the team of your own language) to discuss issues.
- Maintainer submit **.pot* files and are notified when **.po* files are updated.
- Pages in The Translation Project will show where work is needed (consider adopting an orphan **.po* file.)

This is only one way of getting a project translated. A lot of packages have dedicated translators and deal with them directly.

Nested Packages

Nested Packages

- *Autoconfiscated* packages can be nested to arbitrary depth.
  - A package can distribute a third-party library it uses in a subdirectory.
  - It's possible to gather many packages this way to distribute a set of tools.
- For installers:
  - A single package to configure, build, and install.
  - 'configure' options are passed recursively to sub-packages.
  - 'configure --help=recursive' shows the help of all sub-packages.
- For maintainers:
  - Easier integration.
  - The sub-package is autonomous.

Setting Up Nested Packages

- A sub-package should appear as an ordinary directory.
- In *Makefile.am*, this directory must appear in SUBDIRS so 'make' recurses into it.
- *configure.ac* should also declare this directory

```
AC_CONFIG_SUBDIRS([subdir])
```

so 'configure' calls *subdir/configure* recursively.

Nested Packages Example

The *arm* program links with an *hand* library, a nested package in *hand/*.

arm's *configure.ac*

```
AC_INIT([arm], [1.0])
AM_INIT_AUTOMAKE([foreign -Wall -Werror])
AC_PROG_CC
AC_CONFIG_FILES([Makefile src/Makefile])
AC_CONFIG_SUBDIRS([hand])
AC_OUTPUT
```

arm's *Makefile.am*

```
SUBDIRS = hand src
```

arm's *src/Makefile.am*

```
AM_CPPFLAGS = -I$(top_srcdir)/hand
bin_PROGRAMS = arm
arm_SOURCES = arm.c
arm_LDADD = ../hand/libhand.a
```


The End

Where to go Now?

- Locate the reference manuals in your preferred format.
  - Autoconf, Automake, Libtool, and Gettext all install reference manuals in the Info format. (Try '`info Autoconf`', '`info Automake`', etc.)
  - The web pages of these tools also have *.html* or *.pdf* versions.
  - These manuals may not be easy introductions to the tools, but they make good and up-to-date references.
- Subscribe to these tools' mailing lists, to see other people's uses of the tools.
- Pick a package that uses these tools and dissect its setup.
  - Try picking something written by somebody who isn't just another neophyte!
  - I recommend looking at *GNU Coreutils*.