Fixing autotools-related build issues

Debian QA Meeting
Darmstadt, 9-11 September 2005

Sam Hocevar <sam@zoy.org>

What are the autotools?

- They provide an easy way to do this:
  - ./bootstrap
  - ./configure
  - make
- But it also becomes easy to do this:
  - configure.ac: error: possibly undefined macr
  - Makefile.am: TRUE does not appear in AM_COND
  - libtool: link: `0:1:2' is not valid version

How do they work?

- Upstream writes **`configure.ac`**
  - Contains checks for libraries, headers, compiler and platform features
  - Is used to generate **`configure`**
- Upstream writes **`Makefile.am`**
  - Contains build rules for all project targets
  - Is used to generate **`Makefile.in`**
  - Which will be used to generate **`Makefile`**
- Upstream bootstraps project and distributes this bootstrapped version

configure.ac example

```
AC_INIT(main.c)
AC_CONFIG_AUX_DIR(autotools)
AM_INIT_AUTOMAKE(myproject, 1.0)
AM_CONFIG_HEADER(config.h)
AC_CHECK_FUNCS(getopt_long)
AC_CHECK_HEADERS(sys/soundcard.h)
AC_CHECK_LIB(resolv, inet_pton)
AC_TRY_COMPILE(
    [asm volatile(“vperm 0,1,2,3”);])
AC_CONFIG_FILES([Makefile])
AC_OUTPUT
```

Makefile.am example

```
# Define our targets
bin_PROGRAMS = myprog

# Configure our targets
myprog_SOURCES = main.c main.h defines.h
myprog_CFLAGS = -O6 -funroll-loops\
                -faggressive-gentoo-optims \
                -fpimp-my-bike
myprog_LDFLAGS = -lm -lcrypt -lpthread

# That’s all!
```

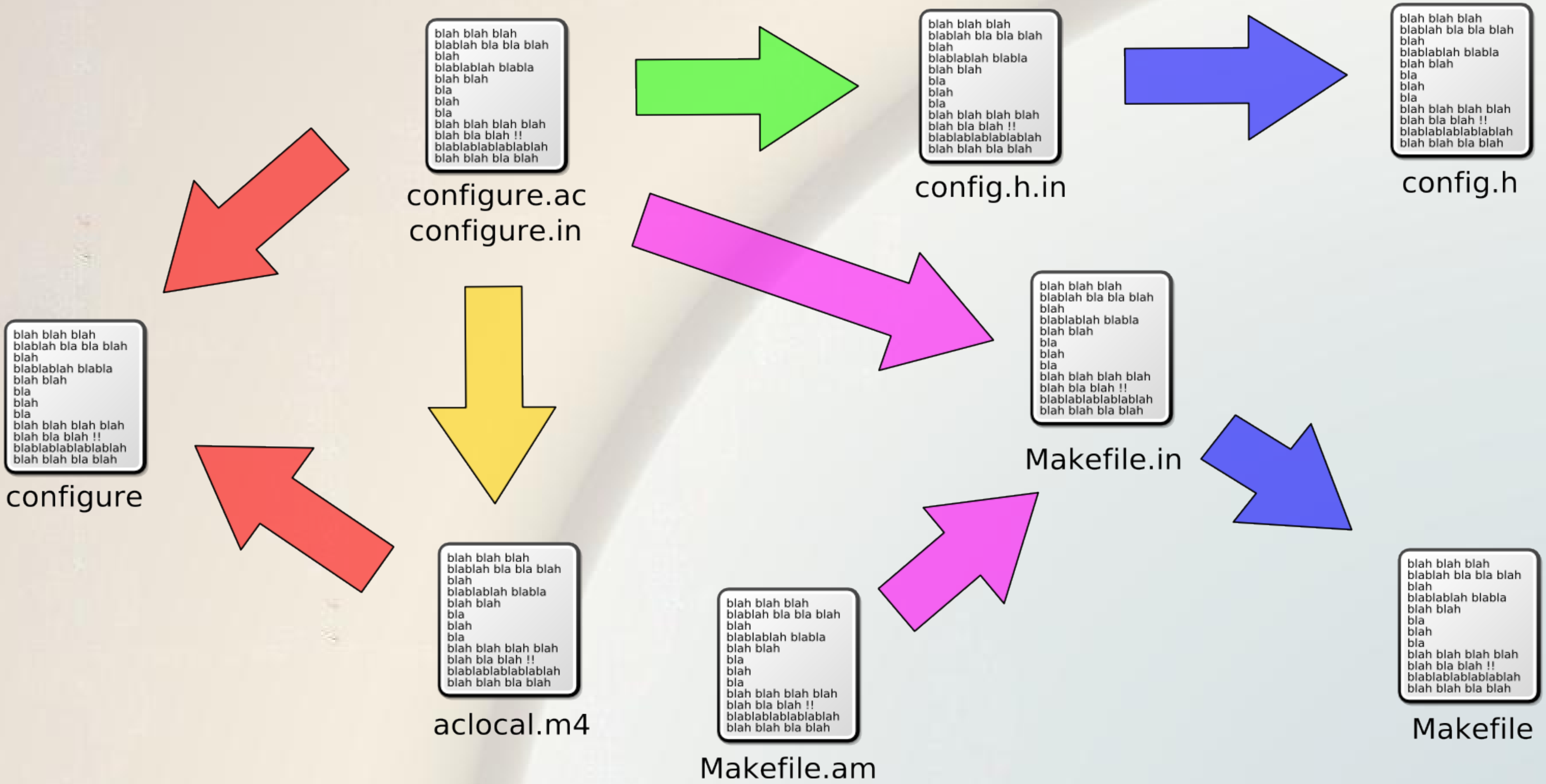
`main.c` example

```
#include “config.h”

#ifdef HAVE_SOUNDCARD_H
#include <sys/soundcard.h>
#endif
#ifndef HAVE_LIBRESOLV
int inet_pton(void) { return 2; };
#endif

int main(void) { return 0x2A; }
```

The bootstrap process



configure.ac
configure.in

config.h.in

config.h

configure

Makefile.in

aclocal.m4

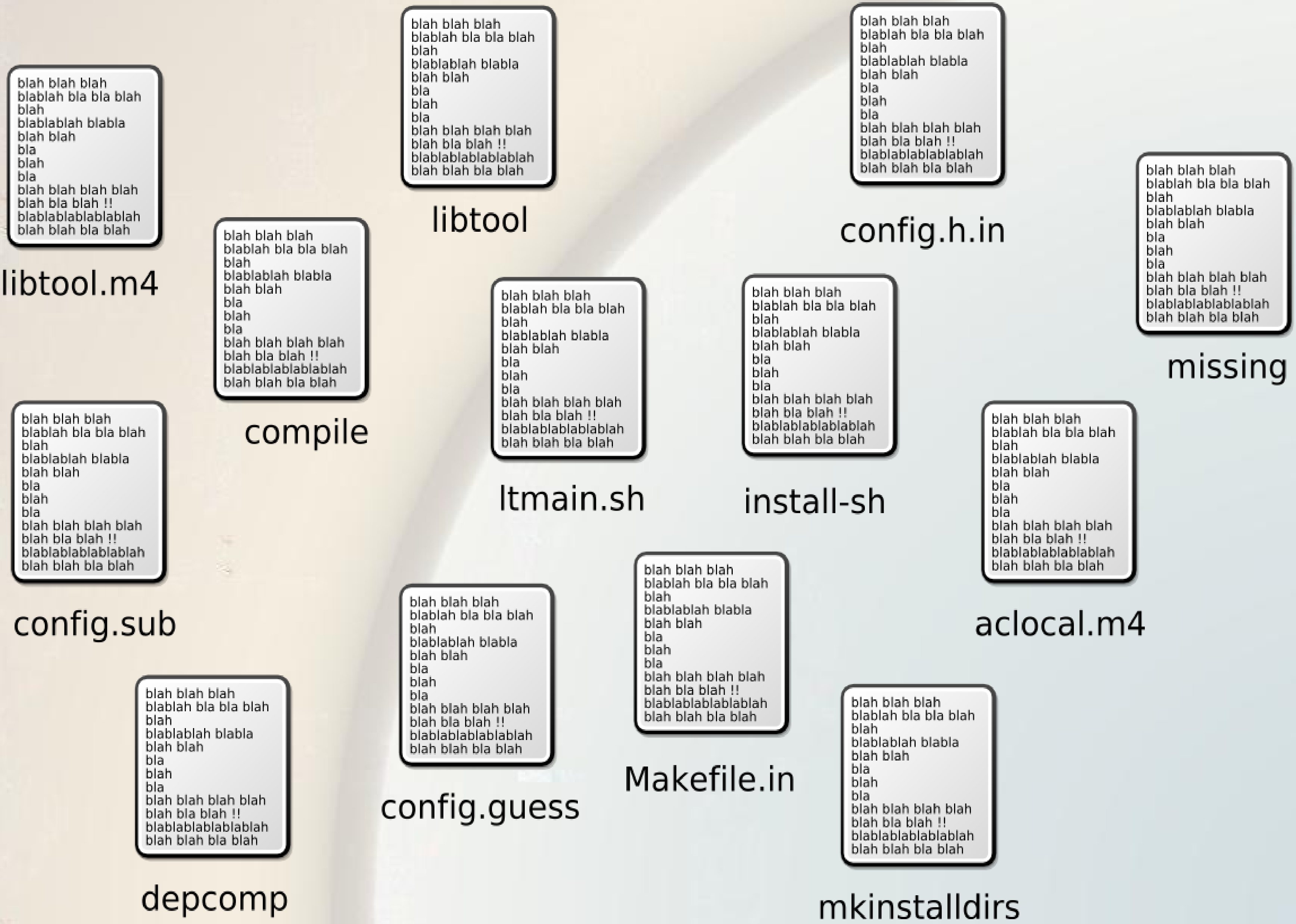
Makefile.am

Makefile

aclocal autoconf autoheader automake

./configure

The bootstrap cruft



libtool

config.h.in

libtool.m4

missing

compile

ltmain.sh

install-sh

config.sub

aclocal.m4

Makefile.in

config.guess

depcomp

mkinstalldirs

But…

- Aren’t the autotools…
  - Slow?
  - Bloated?
  - Constantly causing FTBFS errors?
  - Easily replaced with more powerful tools?
  - Totally outdated 20th century technology?
  - Unmaintainable because they use Perl?
  - The #1 cause for hair loss amongst Debian developers?

Yes, probably!

- But let’s troll about it later, shall we?

Debian packaging integration

- Totally straightforward
  - Upstream tarball includes all files resulting from the bootstrap process (**`config.guess`**, **`config.sub`**, **`install-sh`**, **`ltmain.sh`**...)
  - **`debian/rules`** calls **`./configure`**
  - **`debian/rules`** calls **`make`**
- You don’t have anything to do!
- ... or do you?

Something that does not cause build errors but may give you bad surprises in the future

- Packages that call `./configure` without any arguments
- This is very likely a bug in the package
  - Often prevents the package from building if you do not use `dpkg-buildpackage`
  - Cause of headache when cross-compiling
- Read `autotools-dev`’S `README.Debian.gz`
  - `“Calling GNU configure properly”`

How you should do it

```
export DEB_HOST_GNU_TYPE ?= \
       $(shell dpkg-architecture -qDEB_HOST_GNU_TYPE)
export DEB_BUILD_GNU_TYPE ?= \
       $(shell dpkg-architecture -qDEB_BUILD_GNU_TYPE)

ifeq ($(DEB_BUILD_GNU_TYPE), $(DEB_HOST_GNU_TYPE))
  confflags += --build $(DEB_HOST_GNU_TYPE)
else
  confflags += --build $(DEB_BUILD_GNU_TYPE) \
               --host $(DEB_HOST_GNU_TYPE)
endif

./configure $(confflags) --blahblah --otherflags
```

All is not so nice

- Even a flawlessly autotool’ed package can cause problems
  - Can have been bootstrapped with a buggy version of libtool
  - A package bootstrapped in 2001 can not know about an architecture from 2005
- Sometimes the maintainer is at fault
  - **`debian/rules`** runs a bootstrap process
  - **`debian/rules`** or **`diff.gz`** blindly patch random autotools files

Common build problems (1)

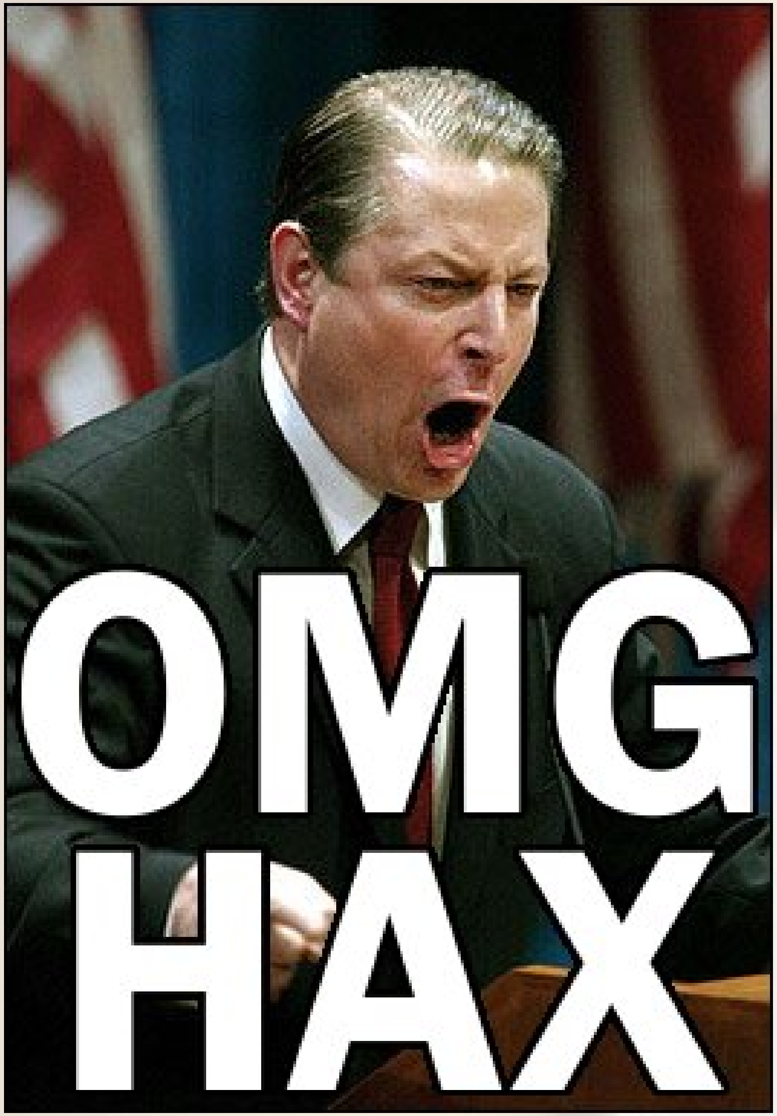
- **`./configure`** fails to detect architecture
- Likely reason:
  - **`config.guess`** and **`config.sub`** are outdated
- Possible ways to solve:
  - Run the bootstrap process and rebuild
  - Run the bootstrap process in **`debian/rules`**
  - Copy the build system’s versions of **`config.guess`** and **`config.sub`** at configure time

Common build problems (2)

- Libraries are not linked with the right libraries, with the right linker (**`gcc/g++`**) or have an incorrect **`rpath`**

- Likely reason:
  - Package was generated with a buggy libtool
- Possible ways to solve:
  - Run the bootstrap process and rebuild
  - Run the bootstrap process in **`debian/rules`**

Bootstrapping in `debian/rules`



- It is ugly. Don’t do it.
  - You will need to build-depend on the proper version of each tool
  - The clean rule will be a nightmare to write
  - It creates useless build-dependencies
  - Unless you really know what you are doing, you will simply get it wrong

Bootstrap and ship in `diff.gz`

- Has a few drawbacks
  - Often creates a huge diff
  - You will need to take care of timestamps
  - Bugs in the autotools are not automatically fixed
- But it’s probably the less ugly solution
  - Does not require insane build-depends
  - You know exactly what is in the autotools files whatever the build environment

How to properly bootstrap

- Order matters, get it wrong and regret it
  - `libtoolize`
  - `aclocal`
  - `autoconf, autoheader`
  - `automake`
- Versions matter, too
  - `aclocal-1.4, automake-1.4, automake-1.9...`
  - `autoconf` is tricky
- Remove the cruft before!

Common build problems (3)

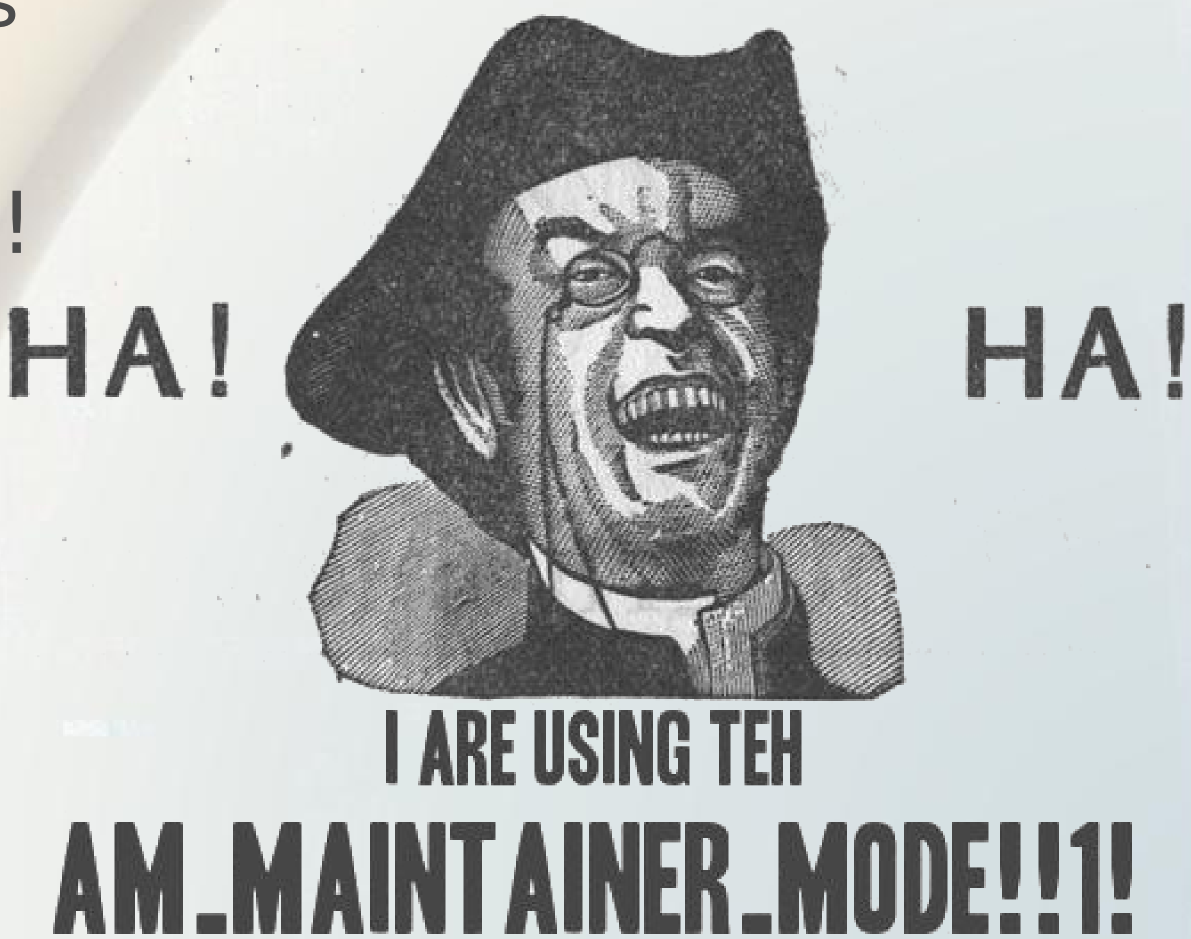
- Build calls **`automake`** or **`autoconf`** despite **`debian/rules`** making no direct call to it
- Likely reason:
  - **`diff.gz`** (or **`dpatch`** et al.) fiddled with autotools files and changed their timestamps
- Possible ways to solve:
  - Fix timestamps in **`debian/rules`**
  - Build-depend on **`autoconf`**, **`automake`**... **NO!**
  - **`AM_MAINTAINER_MODE`**

Easy fix for the timestamp issue

```
# Fix timestamps in autotools files
    touch configure.ac \
    && touch aclocal.m4 \
    && touch configure \
    && touch config.h.in \
    && find . -name Makefile.in \
              -exec touch '{}' ';'
# Configure project
    ./configure $(confflags) ...
```

AM_MAINTAINER_MODE

- Just add this line to `configure.ac`
  - It tells the autotools not to try to regenerate temporary files
- Bootstrap and be done with it!



It is ***your*** job to tell upstream

- The sooner you educate upstream about it, the sooner you can get rid of the ugliness
  - Tell them about AM_MAINTAINER_MODE
  - Tell them to use recent autotools
  - Tell them the Debian versions make better, more portable packages

Any questions?