Autotools: towards a standard approach for building and sharing codes

Alex Pletzer [Tech-X]

pletzer@txcorp.com

March 30 2007

What are autotools?

- Autotools refer to a collection of GNU utility programs for configuring, building, and packaging software:
  - autoconf/autoreconf will generate a configure script.
  - automake/aclocal will generate Makefiles
  - libtool can be used to generate shared objects.

Why should I (developer) use autotools?

- Autotools is about inhibiting obstacles that would prevent the acceptance of your software by others
  - Facilitates collaboration.
  - Reduces amount of documentation.
  - Empowers users. Users are in command.
  - Widely available. Installed by default on many systems.
  - Help make your code more portable.
  - Most open source packages rely on autotools (OpenMPI, MPICH, HDF5, NetCDF, FFTW, ...)

Why I (developer) should *not* use autotools

- I don't like Makefiles
- I want to write my own Makefiles
- I want to support Windows (VC++, Borland, ...). Consider using CMAKE.
- I want a configuation script that automatically downloads software. Look at PETSC's configure.py script.
- For the two reasons above. May want to look at SCons http://www.scons.org

From a user's perspective

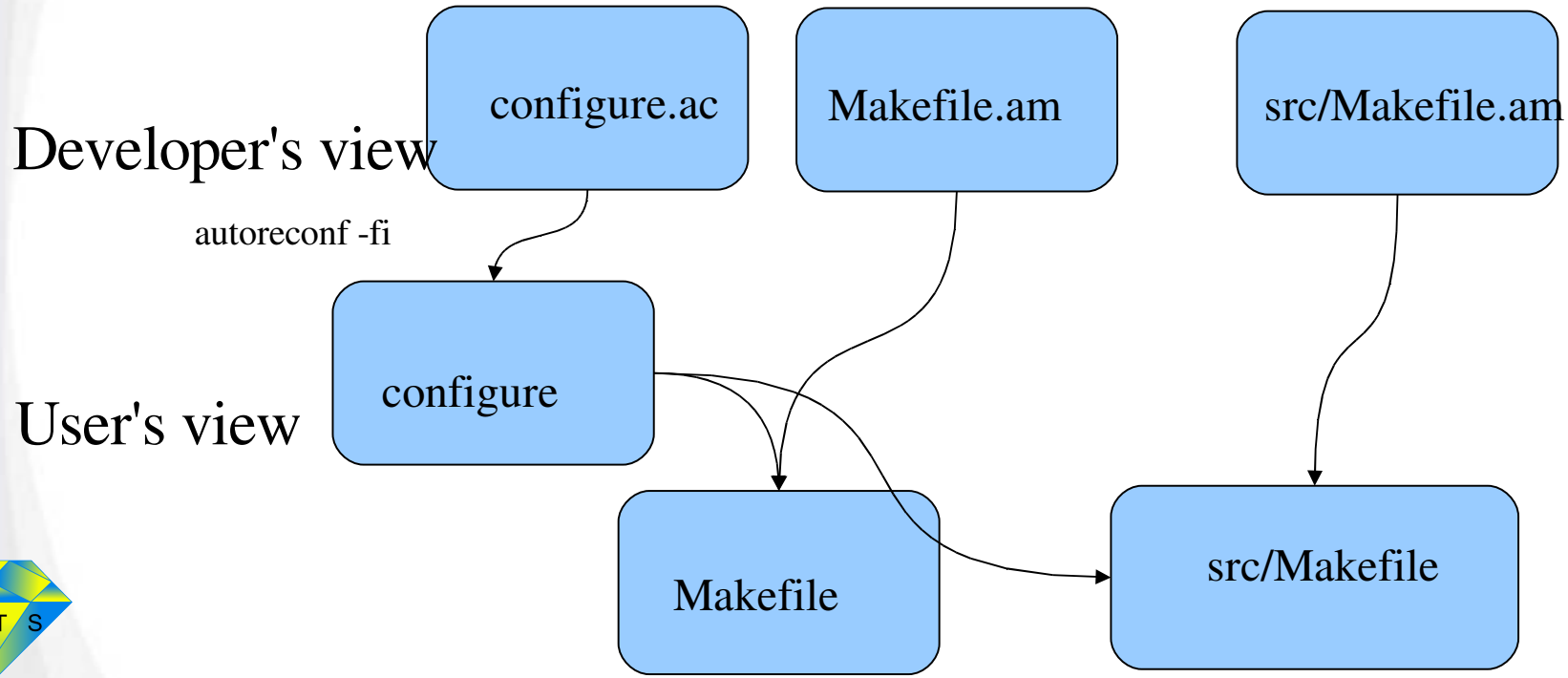
- Building code (typically) involves:
  - wget <http://super.site.org/greatsoft-1.0.tgz>
  - tar xvfz greatsoft-1.0.tgz
  - mkdir mybuild
  - cd mybuild
  - ../greatsoft-1.0/configure [options]
  - make install

Different make's

- make all/clean: build programs, libraries, documentation/erase what was built
- make install/uninstall: build and install/remove installation
- make distclean: like make clean but also removes configure file
- make check: run test suite if any
- make dist: create tar ball

From a developer's perspective

- Developer writes
  - configure.ac: check for tools, headers, libraries, etc.
  - Makefile.am (usually one per directory): determines what to build, dependencies, etc.



Autotools primer

- Consider package “foo”:
  - bar1.f90 will go into a library libfoo.a
  - main.f90 will link with libfoo.a to produce executable foo

```
foo/
`-- src
    |-- bar1.f90
    `-- main.f90
```

Foo package (2)

- Create configure.ac

```
AC_INIT(foo,0.1,pletzer@txcorp.com)
AC_CONFIG_AUX_DIR(config)
AM_INIT_AUTOMAKE
AC_CONFIG_HEADERS([config.h])
AC_PROG_CC
AC_PROG_CXX
AC_PROG_FC
AC_PROG_RANLIB
AC_PROG_INSTALL
AC_CONFIG_FILES([Makefile] [src/Makefile])
AC_OUTPUT
```

Foo (3)

Makefile.am

```
SUBDIRS = src
```

src/Makefile.am

```
foocompdir = $(libdir)/foodir
foocomp_LIBRARIES = libfoo.a
libfoo_a_SOURCES = bar1.f90
bin_PROGRAMS = foo
foo_SOURCES = main.f90
LDADD = libfoo.a
```

Foo (4) ...generating configure

```
[pletzer@localhost foo]$ autoreconf -fi
configure.ac: installing `config/install-sh'
configure.ac: installing `config/missing'
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
[pletzer@localhost foo]$ touch NEWS README AUTHORS ChangeLog
[pletzer@localhost foo]$ autoreconf -fi
```

Foo (5) ...running ./configure

```
[pletzer@localhost foo]$ ./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
...
```

Foo (6) ... building the code

```
[pletzer@localhost foo]$ make
make  all-recursive
make[1]: Entering directory `/home/pletzer/autotools/foo'
Making all in src
make[2]: Entering directory `/home/pletzer/autotools/foo/src'
gfortran  -g -O2 -c -o bar1.o bar1.f90
rm -f libfoo.a
ar cru libfoo.a bar1.o
ranlib libfoo.a
gfortran  -g -O2 -c -o main.o main.f90
gfortran  -g -O2   -o foo  main.o libfoo.a
make[2]: Leaving directory `/home/pletzer/autotools/foo/src'
make[2]: Entering directory `/home/pletzer/autotools/foo'
make[2]: Leaving directory `/home/pletzer/autotools/foo'
make[1]: Leaving directory `/home/pletzer/autotools/foo'
```

How to select another compiler

```
[pletzer@localhost foo]$ ./configure --help

...
 Some influential environment variables:
   CC          C compiler command
   CFLAGS      C compiler flags
   LDFLAGS     linker flags, e.g. -L<lib dir> if you have libraries in a
               nonstandard directory <lib dir>
   LIBS        libraries to pass to the linker, e.g. -l<library>
   CPPFLAGS    C/C++/Objective C preprocessor flags, e.g. -I<include dir> if
               you have headers in a nonstandard directory <include dir>
   CXX         C++ compiler command
   CXXFLAGS    C++ compiler flags
   FC          Fortran compiler command
   FCFLAGS     Fortran compiler flags
```

Choosing a different compiler/compiler flags

```
[pletzer@localhost foo]$ ./configure FC=ifort FCFLAGS="-g -O3 -fno-alias"
...
checking for a BSD-compatible install... /usr/bin/install -c
checking whether we are using the GNU Fortran compiler... no
checking whether ifort accepts -g... yes
```

```
[pletzer@localhost foo]$ make
...
ifort  -g -O3 -fno-alias -c -o main.o main.f90
ifort  -g -O3 -fno-alias   -o foo  main.o libfoo.a
```

Building out of source

- Want a single source for multiple platforms (Linux, Solaris, ...)
- Want to test code with different compilers (ifort, lf95,...)
- Want to run under different configuration settings (-g vs -O3)

```
[pletzer@localhost foo]$ cd ../mybuild/
[pletzer@localhost mybuild]$ ../foo/configure
[pletzer@localhost mybuild]$ make
```

Checking for external library dependencies

- Say you need libnetcdf.a with fortran support
  - In configure.ac, add line

```
AC_CHECK_LIB([netcdf],[nf_open_],
  AC_MSG_NOTICE(netcdf found!),AC_MSG_ERROR(netcdf not found))
```

```
[pletzer@localhost ex1]$ ./configure
...
checking for nf_open_ in -lnetcdf... no
configure: error: netcdf not found
...
[pletzer@localhost ex1]$ ./configure LDFLAGS="-L/usr/local/gfortran/lib"
```

Automiracle?

- Autoconf can help find you compile mixed C++/Fortran code
  - In configure.ac, add line

```
AC_FC_LIBRARY_LDFLAGS
```

```
[pletzer@localhost ex2]$ ./configure FC=ifort
...
checking for Fortran libraries of ifort...  -L/opt/intel/fc/9.1.041/lib -L/usr/lib/gcc/i386-redhat-linux/4.1.1/ -L
/usr/lib/gcc/i386-redhat-linux/4.1.1/../../../ -lifport -lifcore -limf -lm -lipgo -lirc -lirc_s -ldl
...
```

Fortran module dependencies

- Automake will not handle Fortran module dependencies automatically for you...
- ...but you can add dependency information to Makefile.am
- In the example below, joe.o depends on mary.o. Each time mary.o is build, joe.o is rebuilt.

```
AM_FCFLAGS = -I $(top_builddir)/mary
joe.o: $(top_builddir)/mary/mary.o
```

Summary

- Simple configure.ac and Makefile.am can go a long way toward automating builds
- Showed how to:
  - Build executables and libraries
  - Check for compilers
  - Check for libraries
  - Add module dependency
  - Find Fortran link libraries
- Autotools' support for Fortran (90) is improving but not perfect