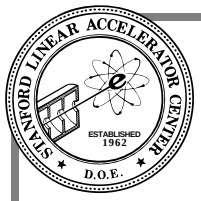


Building with Automake

Paul Kunz

SLAC

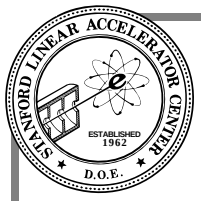
Overview

Examples

Expose the downsides

Benefits

Risks and Costs



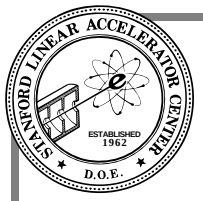
Overview

Primary targets

- build in developer's working directory
- an installation and un-install in shared file system
- a gzipped tar file for distribution or RPM input

Some points

- not everything in working directory gets installed or put into the tar file
- the tar file is self contained and doesn't require external tools except *make* and a compiler
- versioning is handled by libtool
- configure scripts tests the system/environment for the required features, libraries, etc. before building makefiles
- designed for portability
- does not depend on environment variables



The components

Automake

- reads `Makefile.am` to generate `Makefile.in`
- generates “all” the standard boiler plate

Libtool

- tool to build shared and/or static libraries

Autoconf

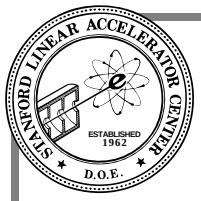
- reads `configure.in` to create `configure` script

configure script

- reads `Makefile.in` to create `Makefile`
- does the machine/environment specific configuration
- non-developers only need to run this script

Am2msdev (written by me)

- tool to create MS Dev Studio project files
- the only tool needed by NT developers/users



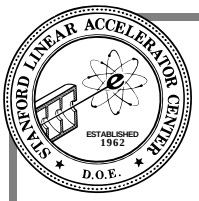
Library Example

In `xml/src/Makefile.am`

```
# $Id: Makefile.am,v 1.4 2000/01/17 02:30:56 pfkeb Exp $
#
# Author: Paul_Kunz@slac.stanford.edu
#

# The following set,  else would follow GNU conventions
AUTOMAKE_OPTIONS = foreign

noinst_LTLIBRARIES = libxml.la

libxml_la_SOURCES = \
    Document.cxx     \
    Element.cxx      \
    IFile.cxx        \
    IFileManager.cxx \
    Persistence.cxx  \
    XML.cxx          \
    XMLini.cxx

INCLUDES = \
    -I$(top_srcdir) \
    -I$(top_srcdir)/../facilities \
    -I$(glast_prefix)/include
```

builds a non-installed shared library

`glast_prefix` is configure-able shared file system path



Library Example continued

In `xml/Makefile.am`

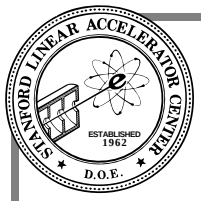
```
## subdirectories to build first
SUBDIRS = xml expat src

## library to be installed
lib_LTLIBRARIES = libxml.la

## source code for the library
libxml_la_SOURCES = version.cxx

## library version info
libxml_la_LDFLAGS = -version-info 1

## contents of the library
libxml_la_LIBADD = \
    src/libxml.la     \
    expat/libexpat.la
```

combines objects compiled in subdirectories into top level library

`version.cxx` is needed because a library must have at least one source file

`version-info` is for the shared library version

clearly subdirectories are supported



Library Headers Example

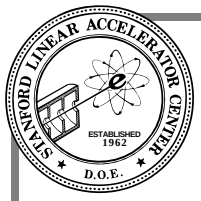
In `xml/xml/Makefile.am`

```
xmldir = $(includedir)/xml

xml_HEADERS =   \
    Document.h     \
    Element.h      \
    IFile.h        \
    IFileManager.h \
    Persistence.h  \
    XMLini.h

noinst_HEADERS = \
    XML.h
```

`xml_HEADERS` are files that must be installed in shared file system

`noinst_HEADERS` are files that will not be installed in shared file system, but must be part of `tar.gz` file

`includedir` == `$(prefix)/include`

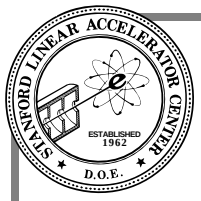


Executable Example

In `merit/src/Makefile.am`

```
bin_PROGRAMS = merit

## The headers are listed so they become part
## of the distribution
merit_SOURCES =                                    \
       AnalysisList.cxx AnalysisList.h             \
        Analyze.cxx Analyze.h                      \
        BackgroundCuts.cxx BackgroundCuts.h\
        Cut.cxxCut.h                               \
        EnergyAnalysis.cxx EnergyAnalysis.h\
        FigureOfMerit.cxx FigureOfMerit.h\
        LayerGroup.cxx LayerGroup.h\
        Level1.cxx Level1.h                        \
        MultiPSF.cxx MultiPSF.h                    \
        PSFanalysis.cxx PSFanalysis.h\
        PSFtailCuts.cxx PSFtailCuts.h\
        Statistic.cxx Statistic.h\
        main.cxx

INCLUDES = \
    -I$(top_srcdir)/../analysis \
    -I$(glast_prefix)/include

merit_LDADD = \
    $(top_builddir)/../analysis/libanalysis.la
```

builds merit by linking against `libanalysis`



automake Generated Targets

`all:` standard default. Will also run automake, autoconf, and configure if either input file has changed.

`clean:` standard

`distclean:` clean plus remove configure generated files

`install:` installs appropriate files in shared file system

`check:` runs the test suite

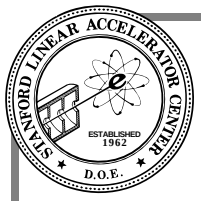
`dist:` builds the tar.gz file

`distcheck:` builds the `tar.gz` file, unzips it, unpacks it to separate directory, does a build, runs the test suite, removes all temporary directories, and leaves the tested `.tar.gz` file for you.

There are others. They follow the GNU Coding Standard document

These are defacto standards in the UNIX world

Can build outside source directory tree



automake Summary

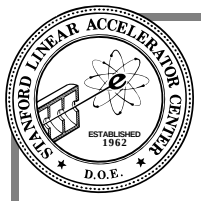
IMHO, automake files are easy to write and modify

Documentation is in texinfo format

- readable by emacs, and Linux help browsers
- convertible to tex, thus to dvi and PostScript and also HTML
- not great, but can learn from plenty of examples from other projects via webCVS

automake is used by GNOME, KDE, and many other Open Source projects

automake is ***not*** needed by NT developers, but can be used under CygWin



Libtool

libtool does not depend on automake/autoconf (could be used by CMT)

However, automake will use libtool when you request shared libraries

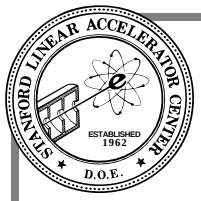
Knows how to build shared libraries on most flavors of UNIX (but not NT)

Supports versioning and version dependencies

- version is M.m.n where
  M is incompatible interface version
  m is compatible interface version
  n is release number, identical interface
- dependent package can specify range of versions it can link with

Uses `-rpath` when linking executable (i.e. does not use `LD_LIBRARY_PATH`

Installing new version in shared file system will not break installed executable



Autoconf

Even more of a defacto standard then automake

A `configure.in` file is mixture of mostly M4 macros and maybe a few bourne shell statements

The autoconf command expands it to configure script, which is designed to be portable

The configure script creates `Makefile` from `Makefile.in` making appropriate substitutions from what it found from its tests.

Example

in `geomrep/configure.in`

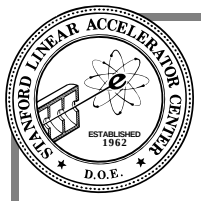
```
AC_PATH_X
```

in `geomrep/test/Makefile.am`

```
X_LIBS   = @X_LIBS@ -lXm -lXt -lX11
```

in `geomrep/test/Makefile`

```
X_LIBS =  -L/usr/X11R6/lib -lXm -lXt -lX11
```



Configurations

What can you configure?

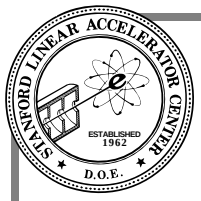
Examples…

- which compiler
- where to install
- where is shared file system
- enable/disable shared or static libraries

Examples

```
../sr_program/configure --prefix=/usr/local/glast
../sr_program/configure --disable-shared
configure --with-glast=/afs/slac/g/glast/prod

setenv CXX /home/pfkeb/gcc-test/bin/c++
../sr_program/configure
```

Options are remembered



GLAST M4 macros

`glast_CHECK_ROOT` -- check that GLAST software is installed in default place, or in path specified by `--with-glast flag`

Use of this macro will ensure that code will build against version you expect

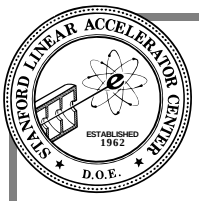
```
glast_CHECK_LIB(LIB_ANALYSIS,
libanalysis.la, ../analysis)
```

check path to specified library,
first for checkout version in working directory area,
second for installed shared file system

If neither is found, the user can not do a build

These are the only two extensions to autoconf I needed

You don’t have to know M4 language to write these



Miscellaneous

In spirit of making a checkout self contained, i.e. not depending on developers having installed anything (but the tools in their standard places), I've added a few extra files to each top level package directory

autogen

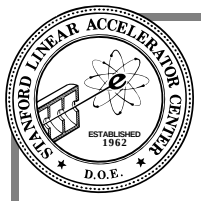
- runs the autoconf, automake, commands in the correct order
- this file is identical in all packages

acinclude.m4

- contains the GLAST M4 macros
- aclocal command builds aclocal.m4 with only the required macros

acconfig.h (if needed)

- contains GLAST specific items to be included in the `config.h` file
- only defines, or not `CLHEP_MAX_MIN_DEFINED`
- autoheader command builds standard `config.h`



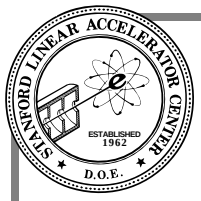
CVS Inteface

One can do standard CVS checkouts

A CVS module, `sr_program`, will build the entire source tree for you

It also will drive make into each subdirectory

But each of our packages can be checked out and built stand-a-lone



Example session

What a developer does to get started

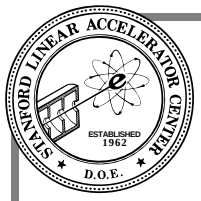
```
libra> cvs co sr_program
(lots of cvs messages deleted)
libra> cd sr_program
libra> autogen
(lots of messages deleted)
libra> cd ..
libra> mkdir sr_program-build
libra> cd sr_program-build
libra> ../sr_program/configure --prefix=/usr/glast
libra> make
```

Its just like building GNU software but with the extra autogen step

Can also build in source directory tree and skip some of the above steps

What a user does to build his own copy

```
libra> tar xzvf sr_program-1.0.tar.gz
(lots of tar messages deleted)
libra> cd sr_program-1.0
libra> configure --prefix=/home/hrm/glast
libra> make
```



UNIX Status

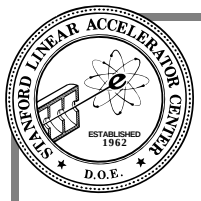
The repackaged off line simulation/reconstruction program and merit have been built with automake since early January

It is ready for developers

SLAC Solaris glitch: autoconf in /usr/local is two years old and I used newer version available under Linux

The repackaged testbeam version can be checked out and almost builds with automake

Missing is some incorrectly tagged files (Sawyer and Marios can sort this out)



am2msdev

All of these things work for UNIX and CygWin, but absolutely zero support for MS Dev Studio

am2msdev creates MS Dev Studio project files, "`.dsp`" and "`.dsw`" from `Makefile.am` files

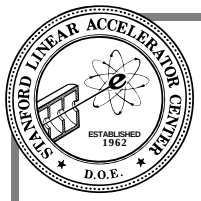
It does ***not*** depend on automake, autoconf, etc. or CygWin.

It is written in C++ because I don't know Perl

If re-written as a Perl module, it could become part of the automake distribution.

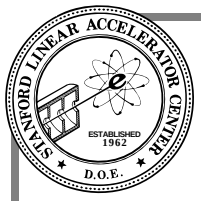
It builds its own project files, and it compiles cleanly under NT (but not yesterday for Toby :-( )

I think it will be a robust program



Status of am2msdev

- It built its own project files after its own directory structure was “packagized” last week
- Early this week, it built our analysis package, but not tested under NT
- it undoubtedly has some bugs
- I may not have made all the right decisions on what the NT directory tree should look like
- it is close to building the packaged version of our code; maybe a week away (but don’t trust time estimates from software people)



Risks and Costs

automake, autoconf, and libtool, are solidly supported on UNIX platforms as Open Source projects and are GNU projects -- very low risk

Will the libtool versioning of shared libraries work for us? -- some risk

Our costs are…

- training our people how to use it -- low cost
- UNIX developers have to install them -- low cost

am2msdev

- our own responsibility
- little interest from the automake mailing list
- no other experiments use it (yet)
- training our people to use it -- low cost
- NT developers have to install it --low cost