



## Fun with Build Systems

Brian Barrett

## The Next 60 Minutes

- Did we all do our homework?
- Subversion is your friend, really
  - Open MPI repositories
  - Simple Subversion commands
- The Autotools: The good, the bad, the ugly
  - Autogen.sh
  - Configure
  - Make

## Subversion

- “a better CVS”
  - Multi-directory transactional commits
  - Symlinks and directories versioned
  - More commands local - less traffic
- All communication over https (open port 443 on your firewalls)
- Very well documented - online book:  
<http://svnbook.red-bean.com/>

## Open MPI Repositories

- <https://svn.open-mpi.org/svn/mpi>
  - The Open MPI source code
  - Directory structure:
    - /trunk development head
    - /branches/v1.0 Open MPI 1.0 release branch
    - /branches/v1.1 Open MPI 1.1 release branch
    - /tags/v1.0 Open MPI 1.0
    - /tags/v1.0.1 Open MPI 1.0.1
    - /tmp/.... Volatile development area

## Open MPI Repositories

- <https://svn.open-mpi.org/svn/mpi-tests>
  - Rich set of MPI test suites
  - Many **not** licensed for redistribution
- <https://svn.open-mpi.org/svn/mpi-docs>
  - Papers, presentations, etc. given about Open MPI
  - Some of these are not yet released / accepted and should not be distributed or referenced without authors' permission

## Using Subversion

- Checking out source code  
svn co <https://svn...org/svn/mpi/trunk>
- Updating source (in checkout directory)
  - svn up [-rNUMBER]
- Seeing what you've changed
  - svn diff [file/directory]

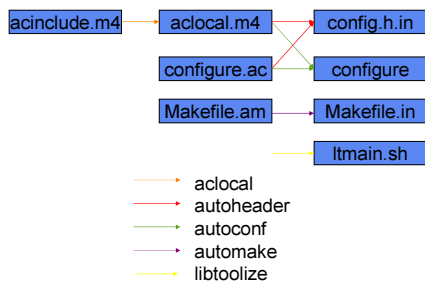
## Autogen.sh

- Helper script for building the build system
  - Used *only* on subversion checkouts
  - Basic sanity checks
  - Generates Open MPI specific data
    - List of active projects
    - List of frameworks / components
  - Runs the GNU Autotools in proper order
    - `aclocal` ; `autoheader` ; `autoconf` ; `libtoolize` ; `automake`
- Good time to check your e-mail...

## GNU Autotools

- Refers to three projects
  - `Autoconf`: Portable testing of Unix features
  - `Automake`: Greatly simplify Makefile creation
  - `Libtool`: Portable building of libraries
- Reduces porting work
  - Unix-like platforms work well
  - Windows causes some issues
- Documentation available from project websites

## The Big Picture



## Autoconf

- Portable package configuration
- Test for features, not operating systems
  - Not always possible
- Input: m4 macros of shell code
- Output: sh-compatible shell code
- Rich set of tests already available
  - Hundreds come with Autotools
  - We have large set of OMPI-specific macros

## Autoconf Example

- Testing for header `foobar.h`
  - In `configure.ac`

```
AC_CHECK_HEADERS([foobar.h])
```
  - In source code

```
#ifdef HAVE_FOOBAR_H
#include <foobar.h>
#endif
```
- Do not rely on headers as proof of functionality

## Automake

- Drastically simplify creating Makefiles
- Automates dependency information
  - Build system (regenerate `configure`, etc.)
  - Source code (rebuild correct files/libraries)
- Simplifies building source code releases
- Input: `Makefile.am`
- Output: `Makefile.in`
  - Running `configure` creates `Makefile`

## Automake Example

- Building the opal\_wrapper compiler script

```
bin_PROGRAMS = opal_wrapper

opal_wrapper_SOURCES = opal_wrapper.c

opal_wrapper_LDADD = \
$(top_builddir)/opal/libopal.la
opal_wrapper_DEPENDENCIES = \
$(top_builddir)/opal/libopal.la
```

## Automake Example 2

- ompi/request/Makefile.am

```
headers += \
    request/grequest.h \
    request/request.h

libmpi_la_SOURCES += \
    request/grequest.c \
    request/request.c \
    request/req_test.c \
    request/req_wait.c
```

## A Word on Libtool

- Most interaction through Libtool
  - LTLIBRARIES targets instead of LIBRARIES
  - Library names end in .la
- Configure-time choice of static, shared, or both for libraries
- Also portably handles dlopen() compatibility with ltdl package

## Configuring Open MPI

- First thing user does
- Sets up the Open MPI build
  - Tests for Operating System / Platform features and issues
  - Tests component for availability
  - Compiler flags / options
  - Library build method (static / shared / both)
- Shorter than autogen.sh, but not by much...

## Specifying Compiler

- Compilers / compiler flags set by environment variables
  - `./configure CC=cc ...`
  - `export CC=cc ; ./configure ...`
- Available variables:
  - `CC, CPP, CXX, CXXCPP, CCAS, F77, FC, OBJC`
  - `CFLAGS, CPPFLAGS, CXXFLAGS, CCASFLAGS, FFLAGS, FCFLAGS, OBJCFLAGS, LDFLAGS`

## Popular Configure Options

<code>--prefix</code>	Installation prefix (\$HOME/local)
<code>--enable-shared</code>	Enable shared libraries
<code>--enable-static</code>	Enable static libraries
<code>--disable-mpi-{f77,f90}</code>	Disable MPI (F77 / F90) bindings
<code>--enable-mpi-threads</code>	Build MPI threading support
<code>--enable-dependency-tracking</code>	Always enable dependency tracking (useful for Solaris)

## More Configure Options

- Interesting networks:  
`--with-{mvapi,openib,gm,mx}=DIR`
- Batch systems  
`--with-{bproc,tm,slurm,xcpu}=DIR`
- Too many other options to list here  
`./configure --help`
- Options that *don't work!*  
`--program-{prefix,suffix,transform-name}`

## Debugging Configure

- Configure is a pain to debug
  - Start by looking at output
  - Then look at config.log
- Usually, if piece of Open MPI missing, it's because something happened here
  - Libraries not found
  - Compilers doing weird things
- Step through some simple examples...

## Component Output

- Compiled with `--with-gm=/dev/null`

```
--- MCA component btl:gm (m4 configuration macro)
checking for MCA component btl:gm compile mode... dso
checking gm.h usability... no
checking gm.h presence... no
checking for gm.h... no
configure: error: GM support requested but not found.
Aborting
```

## Component Output (VAPI)

- Another example - 64 bit VAPI installed, but want to compile 32 bit

```
looking for header without includes
checking vapi.h usability... yes
checking vapi.h presence... yes
checking for vapi.h... yes
looking for library without search path
checking for VAPI_open_hca in -lvapi... no
looking for library in lib
checking for VAPI_open_hca in -lvapi... no
looking for library in lib64
checking for VAPI_open_hca in -lvapi... no
configure: error: MVAPI support requested but not
found. Aborting
```

## VVAPI config.log Contents

```
configure:98317: result: looking for library in lib
configure:98319: checking for VAPI_open_hca in -lvapi
configure:98349: gcc -o conftest -m32 -g3 -Wall -Wundef -
Wno-long-long -Wsign-compare -Wmissing-prototypes -
Wstrict-prototypes -Wcomment -pedantic -Werror-implicit-
function-declaration -fno-strict-aliasing -pthread -
L/usr/lib conftest.c -lvapi -lmosal -lnsl -lutil -lm
>&5
conftest.c:320: warning: function declaration isn't a
prototype
conftest.c:323: warning: function declaration isn't a
prototype
/usr/bin/ld: skipping incompatible /usr/lib64/libvapi.so
when searching for -lvapi
/usr/bin/ld: cannot find -lvapi
collect2: ld returned 1 exit status
```

## Known Issues

- A bit complex....
- Multilib support when linking external libtool libraries
- Static / Shared warnings
- Fortran 90 bindings static only
- Windows support "interesting"