Autoconf macros (m4 files)
2020-06-04
Tuomo Turunen

Terminology

- GNU Autoconf – extensible package of M4 macros for producing shell scripts to configure source code
- GNU Automake – tool for automatically generating Makefile.in
- GNU Libtool – generic library support script
- The above three tools together are called **Autotools**
- GNU M4 – Unix macro processor
- In this slide set ”*Autoconf macro*” refers to ”`*.m4`” file used by GNU Autoconf

Building Autotools components

- SW components using Autotools as build system can be delivered in two different *ways*
  - As source code repositories in version control system (git, subversion, etc.)
    - RCP repositories
  - As archive files (”*.tar.gz”, ”*.zip”, etc.)
    - Common way to download open source SW
    - WadersOS source codes
- The archive file is created by running ”`make dist`”
- The tools and steps needed for building Autotools components depend on the selected delivery method
  - See the next slides

Building from git

1) Install Autotools packages
```
dnf install autoconf automake libtool
```
2) Install build tools and development packages
```
dnf install make gcc gcc-c++ …
```
3) Clone the git repository
```
git clone ...
```
4) Build
```
./autogen.sh
./configure
make all
make check
make install
```

Building from archive file

1) Install build tools and development packages
```
dnf install make gcc gcc-c++ …
```
2) Extract the archive file
```
tar xzf ...tar.gz
```
3) Build
```
./configure
make all
make check
make install
```
Note the differences:
No need to install Autotools packages
No ”`./autogen.sh`”

Autoconf macros

- Autoconf macros extend the Autotools build system
- Autoconf macros can be divided to three categories based on their origin
  1) Automatically generated by "`./autogen.sh`"
  2) Manually downloaded from GNU Autoconf Archive
  3) Proprietary implementation
- All Autoconf macros needed by a SW component are automatically included into the archive file created with "`make dist`"
- See the next slides

Automatically generated

- The following Autoconf macros are automatically generated when running "`./autogen.sh`"

```
aclocal.m4
m4/libtool.m4
m4/ltoptions.m4
m4/ltsugar.m4
m4/ltversion.m4
m4/lt~obsolete.m4
```

- These **must not** be committed to git

Manually downloaded 1/2

- Autoconf macros can be manually downloaded from GNU Autoconf Archive

```
ax_prog_doxygen.m4
ax_code_coverage.m4
ax_cxx_compile_stdcxx.m4
...
```

- These are downloaded to ”m4” directory and committed to git
- There are three **rules** for downloading and committing these macros to git
  1) The files must be added in a **separate git commit**, which does not contain any Nokia code
  2) The files must be added added **as-is** without any modifications
  3) The **commit message** must explain where the files were downloaded from
- Example git commit: Add basic M4 macros
- **Never copy-paste** Autoconf macros from another repository

Manually downloaded 2/2

- Sounds very difficult, is there an easier option?
- Yes! There is much easier option:
  - Do not download any macros and do not add any macros to git
  - Instead, install ”`autoconf-archive`” package and let ”`./autogen.sh`” do the work for you:
    - Fedora, RedHat, WadersOS:
      `dnf install autoconf-archive`
    - Debian, Ubuntu:
      `apt install autoconf-archive`
    - rpm spec file:
      `BuildRequires: autoconf-archive`
- This solves all your problems, as long as you have been using the original GNU Autoconf Archive macros without any Nokia modifications

Proprietary implementation

- Sometimes open source components using Autotools have their own Autoconf macros
- Very few RCP components have proprietary Autoconf macros
- If there is need for proprietary Autoconf macro and there are *m4 wizards* willing to implement Autoconf macros, then the best approaches are:
  1) Push the macro to GNU Autoconf Archive
  2) Make ”`rcp-autoconf-macros`” package and insert the macro there

Special case: ax_code_coverage.m4

- Several RCP components are using proprietary version of ”`ax_code_coverage.m4`” macro
  - The version is based on serial 24 dating back to 2017-04-25
  - Difference to the original is replacing ”`enable_code_coverage=no`” with ”`enable_code_coverage=yes`”
  - Apparently this proprietary version has been copy-pasted from component to component
- Problems:
  - Updating the macro is difficult
  - Replacing the macro with ”`autoconf-archive`” usage is difficult
  - Enabling code coverage by default is wrong way to use the macro
  - (License issues?)
- See the next slide for correct usage

Correct usage of ax_code_coverage.m4

- The macro usage is explained in the macro itself
- Short summary:
  - configure.ac:
    - Add `AX_CODE_COVERAGE`
  - Makefile.am:
    - Add `$(CODE_COVERAGE_CPPFLAGS)` to all `CPPFLAGS`
    - Add `$(CODE_COVERAGE_CFLAGS)` to all `CFLAGS`
    - Add `$(CODE_COVERAGE_CXXFLAGS)` to all `CXXFLAGS`
    - Add "`clean-local: code-coverage-clean`"
    - Add "`distclean-local: code-coverage-dist-clean`"
  - When you want to have code coverage, run:
    - `./configure --enable-code-coverage`
    - `make check-code-coverage`
- Example integration with serial 34: Take ax_code_coverage.m4 into use by Jukka Vainio
- Conde supports generating code coverage report from components using the above instructions

Summary

- Do not commit generated files to git
- There is no need to manually download Autoconf macros from GNU Autoconf Archive
- If you do download Autoconf macros from GNU Autoconf Archive then don’t edit them and commit them as separate commits
- Do not copy-paste Autoconf macros from other repositories
- Read Autoconf macros instructions before using them
  - Especially the ”`ax_code_coverage.m4`”

Thanks!