LibreOffice
The Document Foundation

ROME
CONFERENCE

gbuild Field Guide

Michael Stahl, Red Hat, Inc.

mstahl@redhat.com

ROME | 2017-10-12

Agenda

- Quick Overview of gbuild
- GNU make
- How to add a new Module
- How to add a new Package
- How to add a new Library

Quick Overview: what is gbuild?

- Pseudo object oriented build system
- Implemented in GNU make
  - https://www.gnu.org/software/make/manual/make.html
- Try this: `$ make help`
- $(eval $(call gb_Class_method,instance,argument,…))
- ctags finds definitions:
  - `$ make tags`
  - Vim: set tags=$SRCDIR/tags, Ctrl+]
  - Emacs: (setq path-to-ctags "$SRCDIR/tags")

gbuild Naming Conventions (1)

- Variable name-spacing prefixes:
  - gb_ for core gbuild variables
  - Module name, e.g. sal_
- gb_Class_method
  - public
- gb_Class__method
  - private
- gb_Class_Class
  - Every user Makefile begins with one call to constructor

gbuild Naming Conventions (2)

- **gb_*_add_*** appends something, potentially with dependency
  - gb_Library_add_exception_objects
- **gb_*_use_*** appends & creates dependency between 2 targets in different Makefile
  - gb_Library_use_libraries
- **gb_*_set_*** overwrites something
  - gb_Library_set_componentfile
- **gb_*_get_*** looks something up & returns it
  - gb_Library_get_runtime_filename

Top-level gbuild Files

- solenv/gbuild: build system implementation
  - solenv/gbuild/platform: platform specific bits
- Repository.mk: define all link targets, jars, packages
- RepositoryExternal.mk: bundled external libraries/jars
- RepositoryFixes.mk: ugly hacks
- RepositoryModule_host.mk: lists all modules
- RepositoryModule_build.mk: lists all modules for cross-compilation
- config_host.mk/config_build.mk: build configuration generated by configure
- workdir
- instdir

Common Variables

- OS: WNT, MACOSX, LINUX, ANDROID, ...
- CPUNAME: INTEL, X86_64, POWERPC64, ...
- COM: GCC, MSC
- BUILD_TYPE: list of optional stuff
- Booleans: TRUE or empty string
  - CROSS_COMPILING
  - ENABLE_FOO
  - DISABLE_BAR
  - SYSTEM_BAZ
- BAZ_LIBS / BAZ_CFLAGS

User Makefiles

- */Module_*.mk: modules
- */Package_*.mk: copy files
- */Library_*.mk: dynamic library
- */Executable_*.mk
- */Jar_*.mk
- */Zip_*.mk
- */CppunitTest_*.mk
- */PythonTest_*.mk

GNU make

- make is not a good programming language
  - line ending escape with \
  - comments until end of line
  - no arithmetic
  - no way to declare parameters of functions
  - or name them … $(1) $(2) $(3) …
  - calling non-existent function does nothing
  - long variable/function names
- Mitigation:
  - use auto-completion: e.g. Vim Ctrl+N

GNU make: Conditionals (1)

- Boolean:
- `$(if $(ENABLE_FOO),then-branch,else-branch)`

-
  ```
  ifneq ($(ENABLE_FOO),)
  then-branch
  else
  else-branch
  endif
  ```

GNU make: Conditionals (2)

- Equality:
- `$(if $(filter WNT,$(OS)),then-branch,else-branch)`
- ifeq ($(OS),WNT)

```
ifeq ($(OS),WNT)
then-branch
else
else-branch
endif
```

GNU make: Conditionals (3)

- Conjunction:
- $(if $(and cond1,cond2,cond3,...),then-branch,else-branch)

-
  ```
  ifeq (cond1,)
  ifeq (cond2,)
  then-branch
  endif
  endif
  ```

-
  ```
  ifeq ($(OS)-$(COM),WNT-MSC)
  then-branch
  else
  else-branch
  endif
  ```

GNU make: Conditionals (4)

- Disjunction:
- $(if $(or cond1,cond2,cond3,...),then-branch,else-branch)
- ifneq (cond1cond2,)

```
ifneq (cond1cond2,)
then-branch
else
else-branch
endif
```

- ifneq ($(filter WNT,$(OS))$(ENABLE_FOO),)

```
ifneq ($(filter WNT,$(OS))$(ENABLE_FOO),)
then-branch
else
else-branch
endif
```

How to add a new Module

- foo/Module_foo.mk:

```
$(eval $(call gb_Module_Module,foo))
$(eval $(call gb_Module_add_targets,foo,\
    Library_foo \
))
```

- foo/Makefile: boilerplate, just copy it
- foo/README: what is it?
- RepositoryModule_host.mk: insert foo into long list
- RepositoryModule_build.mk: if necessary for cross compile

How to add a new Package (1)

- Package_foo.mk:

```
$(eval $(call gb_Package_Package,foo,\
    $(call gb_CustomTarget_get_workdir,foo/generated)))
$(eval $(call gb_Package_add_files,\
    foo,$(LIBO_SHARE_FOLDER)/filter,\
        misc/oox-drawingml-adj-names \
        misc/vml-shape-types \
))
```

Source Directory

Target Directory

How to add a new Package (2)

- Module_*.mk:

```
$(eval $(call gb_Module_add_targets,*,\
    Package_foo \
))
```

- Repository.mk:

```
$(eval $(call gb_Helper_register_packages_for_install,ooo, \
        foo \
))
```

Installation Module

How to add a new Library (1)

- Library_foo.mk:

```
$(eval $(call gb_Library_Library,foo))
$(eval $(call gb_Library_set_include,foo,\
      $$(INCLUDE) \
      -I$(SRCDIR)/foo/inc \
))
$(eval $(call gb_Library_use_sdk_api,foo))
$(eval $(call gb_Library_use_libraries,foo,\
    cppu \
    sal \
))
$(eval $(call gb_Library_add_exception_objects,foo,\
    foo/source/bar \
))
```

How to add a new Library (2)

- Module_*.mk:

```
$(eval $(call gb_Module_add_targets,foo,\
    Library_foo \
))
```

- Repository.mk:

```
$(eval $(call gb_Helper_register_libraries_for_install,OOOLIBS,ooo, \
    foo \
))
```

“Layer” Installation Module

How to add a new UNO Library

- foo.component:

```
<component loader="com.sun.star.loader.SharedLibrary"
        environment="@CPPU_ENV@" prefix="foo"
        xmlns="http://openoffice.org/2010/uno-components">
    <implementation name="com.sun.star.comp.foo.FooImport">
        <service name="com.sun.star.document.ImportFilter"/>
    </implementation>
</component>
```

- Library_foo.mk:

```
$(eval $(call gb_Library_set_componentfile,foo,foo/util/foo))
```

Thank you for listening
Questions?

All text and image content in this document is licensed under the Creative Commons Attribution-Share Alike 4.0 License (unless otherwise specified). "LibreOffice" and "The Document Foundation" are registered trademarks. Their respective logos and icons are subject to international copyright laws. The use of these therefore is subject to the trademark policy.