QEMU Status Report

KVM Forum 2020

Paolo Bonzini, Red Hat
Distinguished Engineer
2019 highlights

- Deprecated Python 2 support
- Kconfig integration
- “Fast boot” via PVH and mmap
- Introduced Sphinx for documentation
2019 highlights update

- Remove Python 2 support, follow Python 3 lifecycle
- Completed the conversion to Sphinx!
2020 highlights

- New targets (AVR, RX) and boards
- virtiofsd (virtio filesystem daemon)
- Improved CI
- Meson build system
Improved and consolidated CI

- Gitlab CI: main CI entry point + container registry
- Travis CI: various build combinations + non-x86 Linux
- Cirrus CI: non-Linux systems
  - FreeBSD, macOS, MSYS2
- OSS-Fuzz
- Shippable
- Coverity (daily)
Plans for future CI

- Provide custom runners for Gitlab
- Patchew integration with Gitlab
- Phase out Travis further
- Identify further “holes” between Peter’s setup and CI
Technical debt

- Growth by accretion
- Limited documentation
- Few people knowing the details
- Limited interoperability
Technical debt

- Documentation
- QOM
- Build system
Documentation

- makeinfo, texi2pod + pod2man
  - Not used for developer documentation
  - Extensible with Make + shell
  - Manual build at release time
- Sphinx
  - kernel-doc integration
  - Extensible in Python (+ Perl for scripts/kernel-doc)
  - qemu.readthedocs.io
QOM

“I've started an effort to introduce a consistent object model to QEMU.” – July 2011

“QOM provides a type system that lets objects expose properties to multiple channels.” – January 2020
QOM

- Easily accessible documentation
- Shorter boilerplate
- Easier “qdev” API
Meson build system

“The problem with programmers is that, when they have a problem, they start to program”
Meson build system

WL_U := -Wl,-u,
find-symbols = $(if $1,$(sort $(shell $(NM) -P -g $1 | $2)))
defined-symbols = $(call find-symbols,$1,awk '$$2!="U"{print $$1}')
undefined-symbols = $(call find-symbols,$1,awk '$$2="U"{print $$1}')
process-archive-undefs = $(filter-out %.a %.mo,$1) \ $(addprefix $(WL_U), $(filter $(call defined-symbols,$(filter %.a, $1)), $(call undefined-symbols,$(filter %.mo, $1)))) $(filter %.a,$1)
extract-libs = $(strip $(foreach o,$(filter-out %.mo,$1),$($o-libs)))
LINK = $(call quiet-command, $(LINKPROG) $(CFLAGS) $(QEMU_LDFLAGS) -o $@ \ $(call process-archive-undefs, $1) \ $(version-obj-y) $(call extract-libs,$1) $(LIBS),"LINK","$(TARGET_DIR)$@")
expand-objs = $(strip $(sort $(filter %.o,$1)) $(foreach o,$(filter %.mo,$1),$($o-objs)) $(filter-out %.o %.mo,$1))
define save-vars
$(foreach v,$1,
 $eval save-vars-$v := $(value $v))
$eval save-vars-$v := $(foreach o,$(save-vars-$v),
 $if $(o-cflags), $o-cflags $(eval save-vars-$o-cflags := $(o-cflags))$(eval $o-cflags := )) 
 $if $(o-libs), $o-libs $(eval save-vars-$o-libs := $(o-libs))$(eval $o-libs := )) 
 $if $(o-objs), $o-objs $(eval save-vars-$o-objs := $(o-objs))$(eval $o-objs := )) 
$eval $v := ))
endef
define load-vars
$eval $2-new-value := $(value $2))
$(foreach v,$1,
 $eval $v := $(value save-vars-$v))
$foreach o,$(save-vars-$v),
 $eval $o := $(save-vars-$o) $(eval save-vars-$o := )) 
$eval save-vars-$v := )
$eval saved-vars-$v := )
$eval $2 := $(value $2) $(2-new-value))
endef
define fix-paths
$(foreach v,$3,
  $(foreach o,$($v),
    $(if $($o-libs),
      $(eval $1$o-libs := $($o-libs))
    $(if $($o-cflags),
      $(eval $1$o-cflags := $($o-cflags)))
    $(if $($o-objs),
      $(eval $1$o-objs := $(addprefix $1,$($o-objs))))
    $(eval $v := $(addprefix $1,$(filter-out %/,$($v))) \ 
      $(addprefix $2,$(filter %/,$($v)))))
  endef

define unnest-var-recursive
$(eval dirs := $(sort $(filter %/,$($3))))
$(eval $3 := $(filter-out %/,$($3)))
$(foreach d,$(dirs:%/=%),
  $(call save-vars,$2)
  $(eval obj := $(if $1,$1/)$d)
  $(eval -include $(SRC_PATH)/$d/Makefile.objs)
  $(call fix-paths,$(if $1,$1/)$d/,,$2)
  $(call load-vars,$2,$3)
  $(call unnest-var-recursive,$1,$2,$3))
endef

define unnest-vars
$(if $1,$(call fix-paths,$1/,,$2))
$(foreach v,$2,
  $(call unnest-var-recursive,$1,$2,$v)
  $(foreach o, $(filter %.mo,$($v)),
    $(foreach p,$($o-objs),
      $(if $($o-cflags), $(eval $p-cflags += $($o-cflags)))
      $(if $($o-libs), $(eval $p-libs += $($o-libs)))))
endef
Meson build system

• Building QEMU is a stream
  • Each file should only be read once
  • Gather data, then process it
• Cannot convert everything at once
  • Establish the foundation ✓
  • Start with the low-hanging fruit ✓
  • Everything else can come in later
• Work with Meson upstream
# Meson build system

<table>
<thead>
<tr>
<th>Shell + Make</th>
<th>Non Turing-complete DSL</th>
</tr>
</thead>
<tbody>
<tr>
<td>POSIX shell</td>
<td>Commands as arrays of strings</td>
</tr>
<tr>
<td>(Trying to be) declarative</td>
<td>Imperative (but no aliasing or mutability)</td>
</tr>
<tr>
<td>Strings</td>
<td>Strings, arrays, dictionaries</td>
</tr>
</tbody>
</table>
Meson build system

Before:

8647 configure
1296 Makefile
985 tests/Makefile.include
440 rules.mak
397 default-configs/*
287 Makefile.target
215 Makefile.objs
97 {qapi,trace}/Makefile.objs
129 scripts/create_config

12493 total

After:

7289 configure
1989 meson.build
655 default-configs/**
542 tests/{,qtest/}meson.build
264 Makefile
158 tests/Makefile.include
223 {qapi,trace}/meson.build
69 docs/meson.build
1008 scripts/ninjatool.py
130 scripts/mtest2make.py
49 scripts/undefsym.py

11364 total

Red Hat
Hall of Fame
Google Summer of Code + Outreachy

3 projects accepted, passed and merged!

- César Belley: Virtual FIDO/U2F security key
- Filip Božuta: Extend linux-user syscalls and ioctls
- Ahmed Karaman: TCG Continuous Benchmarking
Shoutouts

• Thomas Huth and Alex Bennée: CI
• Markus Armbruster, Daniel Berrangé, Eduardo Habkost: QOM and qdev refactoring
• Eduardo Habkost, Peter Maydell: documentation
• Richard Henderson: all things TCG and more
• Laurent Vivier, Philippe Mathieu-Daudé: hobbyist spirit
• Peter Maydell: merging everything
What’s next?
More gitlab?

- Static site generation
- Primary repository
- Release process
- Issue tracking
- Wiki
Rethinking the QEMU API

• "Making QEMU easier for management tools and applications" (Stefan Hajnoczi + 179 more messages over 1.5 months)

• “QEMU has 131 command line flags.” (John Snow)
  • More API, less command line!
  • Embrace QMP for configuration, extending -preconfig

• “Official” bindings for QAPI through code generation
Multiprocess QEMU (and Rust-y QEMU)

- vhost-user servers with qemu-storage-daemon
- Possibly vfio-user?
- “Why QEMU should move from C to Rust “ (Stefan again!)
  - “Learning Rust is humanly possible, writing bug-free C code is not.”
- Looking at integration of QAPI with Rust
Thank you