Developing Libvirt

Michal Prívozník
mprivozn@redhat.com

https://mprivozn.fedorapeople.org/

Beijing 2016
Introduction

What will you learn?

- How upstream works
- How downstream works
Terms clarification

**list** The mailing list libvir-list@redhat.com where patches, design decisions, general discussion take place.

**upstream** Raw product from repository.

**downstream** Stabilized product, where only bug fixes are backported.

**BZ** The Bugzilla bugzilla.redhat.com. For both upstream and downstream bugs.
Upstream process

1. You spot/are assigned a bug/RFE\(^1\)
2. Investigate the code, try to understand it
3. Write a patch, test it locally
4. Send the patch to the list
5. Wait for review
   positive Congratulations, you’re done
   negative Goto 2

\(^1\)Request For Enhancement
1. You spot/are assigned a bug/RFE

**bug** Is it really a bug? Sometimes easy to answer (e.g. a crasher), sometimes harder (should an attribute be formatted in domain XML?).

**RFE** It’s crucial to understand the feature. Get clear design.

In both cases, if in doubt, it’s better to ask on the list.
1. You spot/are assigned a bug/RFE

- Try to find in the BZ if the bug is not already reported
- If it is:
  - Has somebody assigned, ask them
  - Note down BZ number since it’s needed later
2. Investigate the code, try to understand it

- Libvirt’s source code is complex. Functions are called from many places.
- Many (hidden) dependencies.
- There’s a lot of info on Web/Wiki pages.

If in doubt, it’s better to ask on the list.
3. Write a patch, test it locally

- Commit message is important
  - Talk to me like to a child, explain details
  - Help maintainers (blame commits, put BZ links)
- One semantical change per commit
  - For instance, code refactor and bug fix should be two different commits.
- make all syntax-check check
Bad example:

commit 265ea811cbc176c007f9771ca968b4b2ec7d5afe
Author: Michal Privoznik <mprivozn@redhat.com>
AuthorDate: Tue Jan 25 18:31:00 2011 +0100

bugfix for https://bugzilla.redhat.com/show_bug.cgi?id=671050

virsh simply refutes to detach-interface in case when multiple interfaces are attached and --mac is not specified.

Good example:

commit 1268820a632846267b75432dc155779a716e289c
Author: Eric Blake <eblake@redhat.com>
AuthorDate: Thu Apr 23 08:13:53 2015 -0600

build: avoid obsolete index()

Commit 2a530a3e5 is not portable to mingw, which intentionally avoids declaring the obsolete index(). See also: https://bugzilla.redhat.com/show_bug.cgi?id=1214605

* src/util/virstring.c (virStringStripControlChars): Use strchr.

Signed-off-by: Eric Blake <eblake@redhat.com>

3. Write a patch, test it locally

- Fixing bug? Introduce test
- Implementing RFE? Introduce test
- Documentation, RNG schema
- Follow Contributor guidelines
  http://libvirt.org/hacking.html
Upstream process (cont)

4. Send the patch to the list

- Use of git send-email is strongly recommended
- Don’t hand edit generated patches\(^2\).
- Make sure your patches are applicable onto current HEAD
- Don’t CC random people, all developers are subscribed to the list.

\(^2\)Except writing notes at the correct place, and even with that \textit{git-notes} is preferred
5. Wait for review

- On positive review patches are merged
- On negative review, rewrite the patches as suggested
  - A question in the review doesn't mean “change it”.
- Resend with subject prefix altered
  - `git format patch -v2 ...`
- If you get no review in two weeks, ping
Subprojects

1. Language bindings (python, perl, ruby, C#, Java, ...)
2. Different models (CIM, QPID, GObject, SNMP)
3. Others (designer)

- Use the list with correct prefix
  ```bash
  subproj.git $ git config format.subjectprefix subproj][PATCH
  [libvirt-python][PATCH] event-test: support ...
  ```
Downstream process

Upstream first!

1. Selected upstream patches are backported
   
   git cherry-pick -sex $hash

2. ABI stability preserved, no API backport possible

3. Merge conflicts resolving

4. Backward compatibility (not losing domains, their state, . . . )
Questions?