Upstreaming 201

Mathieu Poirier

Linaro
connect
Budapest 2017

ENGINEERS
AND DEVICES
WORKING
TOGETHER

Upstreaming 201

- Highlights from Upstreaming 101
- Get you the right tools to start contributing upstream
- Things to keep in mind when preparing a patch
- Follow-up after a patchset has been submitted
- Overview of commonly used tags
- Any question you may have about upstreaming code

Highlights from Upstreaming 101

- Have the right mindset
  - Be Altruist → we win as a community, not as individuals
  - Be Polite → people genuinely want to help
  - Be Patient → the community has no deadline
  - Be Factual → why your submission is important, no “hunch”
  - Be Humble → bragging is unpleasant and doesn’t lead anywhere
  - Be open to change → you will likely have to rewrite/amend your code several times

- Upstreaming 101/201 is a summary of these documents:
  - Documentation/process/*.rst
  - Documentation/process/submitting-patches.rst

- Read those documents **before** trying to upstream anything

A Word About your Name in the Community

- Email/patches you send out stay forever (see marc.info mailing list archive)
- Maintainers are very busy→ Don’t waste their time
- Be mindful of what you do→ Don’t pick fight with people
- Keep people on the CC list and avoid point-to-point discussions
- Once you’re in the blackbook, it is hard to get out of it

  http://lists.openwall.net/netdev/2016/03/16/164

Preparing Your Patches: The Right Subject Line

- The subject line of a patch is very important - many people filter on it
- Also help people quickly identify the purpose of the patch
- Example:

```
$ git log --oneline -5 kernel/sched/deadline.c
176cedc4ed14 sched/dl: Fix comment in pick_next_task_dl()
9846d50df3de sched/deadline: Fix typo in a comment
61c7aca695b6 sched/deadline: Fix the intention to re-evalute tick dependency for
d8206bb3ffe0 sched/deadline: Split cpudl_set() into cpudl_set() and cpudl_clear
12bde33dbb3e cpufreq / sched: Pass runqueue pointer to cpufreq_update_util()
```

- The subject of the new patch should likely be “**sched/deadline:** ...”

Preparing Your Patches: Signed-off-by

- Add a “Signed-off-by:” statement to **every** patch you send
- Cover letters don’t need a SOB (Signed-off-by)
- Certifies that:
  - You are the author of the patch
  - You have the right to publish this code as open-source
- Legally Binding → you are responsible for your actions

- Example:

```
Signed-off-by: Random J Developer <random@developer.example.org>
```

- Other particularities about SOBs → read section 11 of **`Documentation/process/submitting-patches.rst`** for all the details

Preparing Your Patches: The Right Format

- Use "**`git format-patch`**" to generate patches → nothing else
- Always compile your code
  - Don't get caught with uncompiled code by a maintainer (it happens all the time)
  - Generally a good idea to compile for both ARM and x86

- Clean patches using the "**`checkpatch.pl`**" script

```
./scripts/checkpatch.pl 0001-sched-core-Fix-rd-rto_mask-memory-leak.patch
total: 0 errors, 0 warnings, 8 lines checked

0001-sched-core-Fix-rd-rto_mask-memory-leak.patch has no obvious style problems and is ready...
```

- Checkpatch.pl assert the code is **syntactically** correct, nothing more

Sending Your Patches: Targeting The Right People

- Sending patches to a mailing list isn't enough
  - the maintainer has to be notified

- To know who and where to send patches to, run the `get_maintainer.pl` script:

```
$ ./scripts/get_maintainer.pl 0001-sched-core-Fix-rd-rto_mask-memory-leak.patch

Ingo Molnar <mingo@redhat.com> (maintainer:SCHEDULER)
Peter Zijlstra <peterz@infradead.org> (maintainer:SCHEDULER)
linux-kernel@vger.kernel.org (open list:SCHEDULER)
```

- **get_maintainer.pl** is very chatty, try to understand why people are only list

Sending Your Patches: `git send-email`

- The command **`git send-email`** is part of the git tools
- Best way to get things right
- Maintainers can't apply patches that aren't formatted properly
- Lots of documentation available on how to setup a working environment

- Before sending a set of patches out for review, send them to yourself first
  - Avoids spelling mistake and typos
  - Allows you to catch anything you don't want to see going out like company disclaimers
  - Use the **--suppress-cc=all** option or patches will be sent to CC'ed individuals
  - Do yourself a favour, just do this extra step compulsively.

I Have Sent My Patches For Review - Now What?

- Double check on the mailing that patches have been published
- Most maintainers will reply within 7 days
  - It can take longer based on the maintainer's schedule

- If you don't get anything back within 7 days
  - Look for the maintainer on the mailing list - good indication on how busy they are
  - Gentle ping:
    - **Politely** ask the maintainer if they have seen your patches
    - **Politely** ask the maintainer if there is anything you can do to make reviewing your patches easier for them
    - Don't be surprised if they are annoyed by your ping

- It is always a very bad idea to pressure a maintainer
  - The community is not bound by internal company deadlines

What To Do With Comments?

- **NEVER** ignore comments that were received
- If you don’t understand a comment, ask questions
  - It is better to ask questions then sending a new revision with the same problems
- If you don’t agree with a comment, justify your approach
- If people like your code they may reply with “Reviewed-by” or “Acked-by”
  - Don’t forget to take note of them
  - Those are needed for your next revision
  - Greatly motivates a maintainer to accept your code

Sending A New Revisions

- When **all** comments have been addressed, it is time to send another revision
- It greatly helps the reviewers to add a version tag in the header:

```
[PATCH v2] soc: ti: knav_dma: Fix some error handling
```

- Revision number can be added by hand or by using the "`--subject-prefix`" option when using `git format-patch`:

```
$ git format-patch --subject-prefix="PATCH v2" ....
```

- As a rule of thumb, wait 24 hours between revisions
  - Avoids flooding the mailing list
  - Reduces the probability of people reviewing the wrong version
  - Gives you time to think about your code again

When Your Patch Gets Accepted

- Most subsystem maintainers will notify you via email
- Others will simply add your patch to their tree without notification
- It is your responsibility to check that everything is working properly
  - In linux-next
  - In the next cycle when -rc1 is released

Sending More Than One Patch In A Set

- Big patches are always harder to get accepted
  - Huge burden on the reviewers
  - You will likely be told to split a big patch before it can be reviewed

- Split patches in small chunks
  - Each chunk needs to be self-contained
  - Each chunk needs to be bisectable (see git bisect for details)

- When working with a feature that spans multiple patches:
  - Always use a cover letter → “`git format-patch --cover-letter ...`”
  - Patch version also applies when working with patchsets

A Word On “Reviewed-by” And “Acked-by”

- As you become more familiar with a subsystem you may start reviewing other peoples’ work (greatly encouraged)

- Be careful when using the “Reviewed-by” and “Acked-by” tags

- If you don’t understand the difference, you may convey something you did not intend to.

The “Reviewed-by” Tag

- The strongest way to signify your approval and esteem of a patch
- Indicates that a reviewer has spent time to go through a patch and understand what it does in details
- Implies the reviewer has knowledge of the subsystem and the patch won’t break existing code
- May entice a maintainer to spend less time on a patch then they would normally have done
    - As such, if something goes wrong you may have to answer questions

The “Acked-by” tag

- Can be strong or weak, it depends on who gives it
- If you are **not** the maintainer of the subsystem, an “Acked-by” stipulate that you have reviewed the code - nothing more (weak semantic)
- When given by the maintainer of a subsystem, an “Acked-by” means the code can be merged by someone else (strong semantic)

Bonus: The “Tested-by” tag

- Very good way to make news friends in the open source community
- Gets you familiar with a subsystem
- Allows you to see how small changes affect things
- Really helps maintainers having greater confidence in a patch

Your Turn To Talk

Questions ?

Thank You

#BUD17

For further information: www.linaro.org

BUD17 keynotes and videos on: connect.linaro.org