How (and why) Chrome OS Works with Upstream Linux

Doug Anderson, Google
About Chrome OS

- Chrome OS devices auto-update every ~6 weeks
- Devices are supported for ~6 years
- Lots of devices (over 100)

Impossible to maintain a private kernel for every device
The Chrome OS kernel

- Pick LTS kernel every year because we merge stable releases
- All devices that year use that kernel
- Still end up with 6 - 7 active kernel branches, but better than 50

Kernel development must be in public to share between partners and avoid per-SoC hacks since many disparate devices share one kernel
Planned: kernel uprevs

- Looking to uprev (switch to a newer kernel version) at least once during device lifetime
- LTS lifetime doesn't match ~6 year support
- Even if LTS did promise 6 - 7 years and backported every fix, it's still a pain to maintain so many kernels.

Must make strong effort to upstream patches so uprev isn't impossible
How Chrome OS upstreams patches

- Submit patches to the mailing list
- Don't submit to Chrome OS tree if changes requested on the list
- Give some time (at least a day, ideally a week) for upstream to respond
- If upstream is silent, a Chrome OS Engineer can review and land FROMLIST
- If possible, wait until a patch lands in a maintainer tree
- Chrome OS Engineers encouraged to review on the list rather than privately on the Chrome OS gerrit
Developing an upstream culture

- Chrome OS isn't homogeneous
- Some Engineers / teams are more on the upstream bandwagon than others; that's just reality
- We are continuing to build a culture that works upstream
- Upstream can be a scary place. Hopefully getting better but still hard for people to get used to
- If an Engineer is new (not confident about posting upstream), sometimes we do a private review of their patch first
Problem: silence

- Silence is not Golden. Silence is deadly
- The best thing we can do to combat silence is to find others interested in our patch
- Second best thing is to find someone (even a co-worker) who will provide an honest review of your code to help offload maintainer
Problem: delayed feedback

- Upstream reviewers / maintainers don't get paid to ship our product
- They are busy, have lives, and have other priorities
- Feedback can come weeks or months after submission, after patch has already landed in Chrome OS
- Important to still prioritize a response / re-spin. We still want the patch to land upstream even if we already landed a patch in Chrome OS
- If possible, we pick FIXUP patches
- If we want to post future patches that build upon this patch: no good answer
Problem: yak shaving requested

- Upstream will sometimes request that we clean the subsystem before our patch can land
- Major subsystem cleanup can be outside the scope of our project
- Sometimes upstream will still let you land and leave the yak unshaved; if not then no good answer
Problem: don't spin so fast

- Mailing lists make it hard to follow conversations across revisions
- Reviewers often don't like quick spins of patches while they are still being discussed
- Chrome OS policies (should land things FROMLIST and resolve all feedback) combined with deadline pressure from projects tend to encourage quick spins, though
- Solution: how fast to spin depends on which maintainers and reviewers are involved. Definitely give time before reposting if there are open questions
What the Chrome OS tree looks like

- **UPSTREAM**: picked directly from Linus's tree with "git cherry-pick -x"
- **FROMGIT**: picked from a git tree, usually a maintainer tree that will feed mainline; contains a reference to the tree and git hash (which hopefully won't go away).
- **FROMLIST**: picked from a mailing list; contains a reference to it
- **BACKPORT**: applied to one of the above when there were merge conflicts
- **FIXUP**: Amend a previous commit (maybe we landed an early version)
- **CHROMIUM**: not destined for upstream (ideally just config / early dts)

https://chromium.googlesource.com/chromiumos/docs/+/master/kernel_faq.md#UPSTREAM_BACKPORT_FROMLIST_and-you
Why Chrome OS is good for upstream

- Upstream gets benefits of features, bugfixes, testing, HW support right away
- Upstream has a chance to set direction before it's all done/tested "the wrong way" and later has to be redone by upstream
- Requires HW vendors to work with upstream too
- Chrome OS provides review bandwidth upstream
- Chrome OS merges linux-stable and thus provides extra testing there (and we help tag patches for stable)

~10 years ago there was much angst about Android doing all private development. Chrome OS tries to not replicate that.
Why upstream is good for Chrome OS

- Makes uprevs possible
- Starting the next project is easier since most things are already upstream
- "Free" code reviews; often finds much better solutions / identifies problems
- "Free" bugfixes as others use the code / fix problems
  - Some of this comes from staying aligned with linux-stable
- Sometimes is seems like a huge pain to address all upstream feedback but in the end we get something that's solid
Why Chrome OS is bad for upstream

- Floods the community with less experienced Engineers from partners who are only there because Chrome OS forced them to be there
- People participating are more interested in a quick solution than great long term solutions

Hopefully some of the above is offset by the "good" stuff, plus Chrome OS does fund some long term cleanups outside the context of projects
Why upstream is bad for Chrome OS

- Things are much less secret; people know what's being worked on
  - Does it really matter? Yeah, you're working on new devices with new CPUs.
- Engineers "waste" their time on upstream requests
- Longer patch/commit/test cycle because we spend some time waiting for upstream reviews
- Competitors benefit from our fixes right away (though we benefit from theirs)

Downsides are really not so serious.
How upstream can help Chrome OS

- Communicate, even if just saying "I don't have time to review but here's a path forward"
- Be pragmatic; bias slightly more toward land and iterate
- Take into account that Chrome OS is trying to ship products with a deadline
- If possible, maintainers can provide stable git hashes in their trees for Chrome OS FROMGIT picks
  - If a maintainer rebases and Chrome OS picked FROMGIT, our old git hash is meaningless

Upstream clearly has no obligation here!

...but maybe I've convinced you that upstream benefits from Chrome OS relationship so it's worth working with us?
Data Points

- Rockchip rk3288/rk3399 are fairly well supported upstream because of Chrome OS efforts
  - Upstream work on rk3288 meant that rk3399 project was easier
  - We are attempting to uprev rk3288 from 3.14 to 4.19, which isn't as crazy as it sounds
  - Other Rockchip SoCs (even non-Chrome OS ones) have pretty good upstream support because Rockchip has been convinced of the value

- When Chrome OS switches kernel versions it's still not trivial but definitely not a horror story
Conclusion

- Balance between shipping and making deadlines and the "it'll land when it's ready" of upstream is precarious, but worth it
- In the day and age of crazy security vulnerabilities, Internet-connected devices, and long support lifetimes there's not much choice
- Maybe other companies will want to follow Chrome OS model?