Qualcomm

Open Source Summit India 2026

Guide to Becoming a Linux Kernel Maintainer

Krzysztof Kozlowski

Senior Staff Engineer
Qualcomm Wireless GmbH

krzk@kernel.org, @krzk@social.kernel.org

Introduction

- Krzysztof Kozlowski
- I work for Qualcomm

Introduction

- Krzysztof Kozlowski
- I work for Qualcomm
- I am a co-maintainer of a few subsystems in the Linux kernel
  - SoC subsystem (formerly arm-soc)
  - Devicetree bindings
  - Memory controller drivers
  - Samsung Exynos SoC ARM/ARM64 architecture

Introduction

- Krzysztof Kozlowski
- I work for Qualcomm
- I am a co-maintainer of a few subsystems in the Linux kernel
  - SoC subsystem (formerly arm-soc)
  - Devicetree bindings
  - Memory controller drivers
  - Samsung Exynos SoC ARM/ARM64 architecture
- I contributed a lot to the Linux kernel, although now I mostly read the code
  - git shortlog -s -n --no-merges | head -n 5
  - Basically, since v3.11, I have contributed to every Linux kernel release

Talk Overview

- Why do we need maintainers?
- Prerequisites
- A few ideas - discussion
  - Upstream
  - Reviews
  - Orphaned subsystems

Why Do We Need Maintainers?

Who Is a Maintainer?

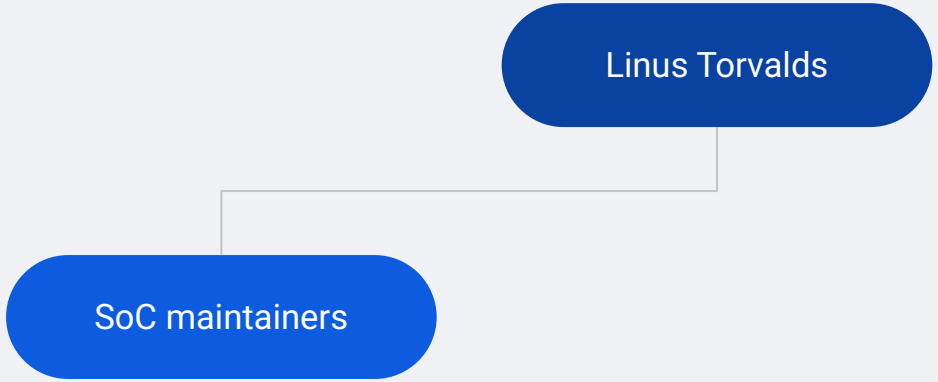
- Person with a known identity
  - Responsible for a piece of code: driver, subsystem
  - Providing reviews, handling incoming patches
  - Participating in community discussions
  - Handling bug and regression reports
    - Not necessarily doing the work
- IOW, person who cares about their codebase
  - https://www.kernel.org/doc/html/latest/maintainer/feature-and-driver-maintainers.html
- One cannot me a maintainer not being a contributor first

Subsystem Maintainers

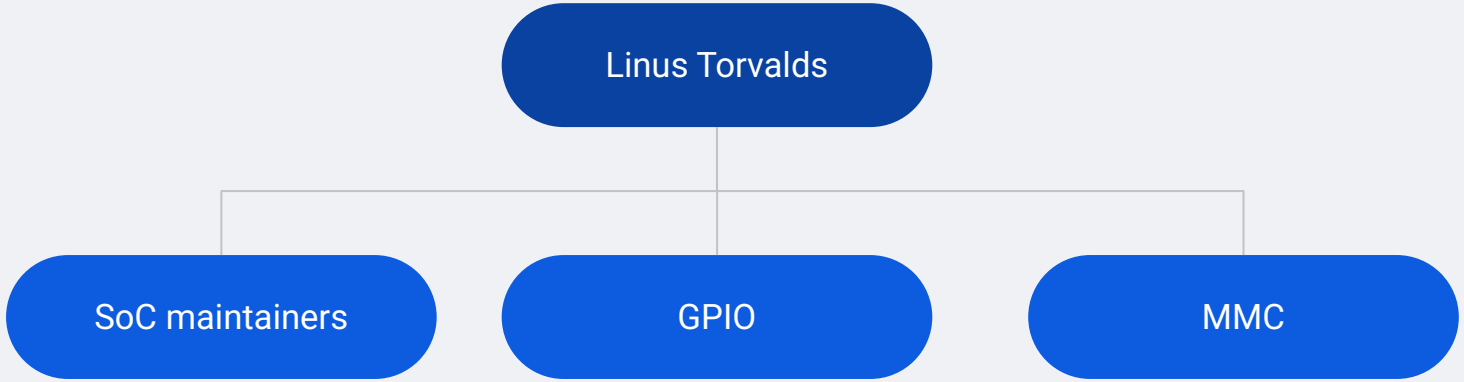
- Development is organized in hierarchical way
- Parts of code, often called subsystems, are maintained by dedicated maintainers which:
  - Have their own Git tree
  - Review and take patches from the mailing lists into that tree
  - Send accepted code to their upstream maintainer (or to Linus)
  - Around ~300 of such maintainer trees
    - But many more maintainers, a few per subsystem sometimes

Maintainers Hierarchy Example

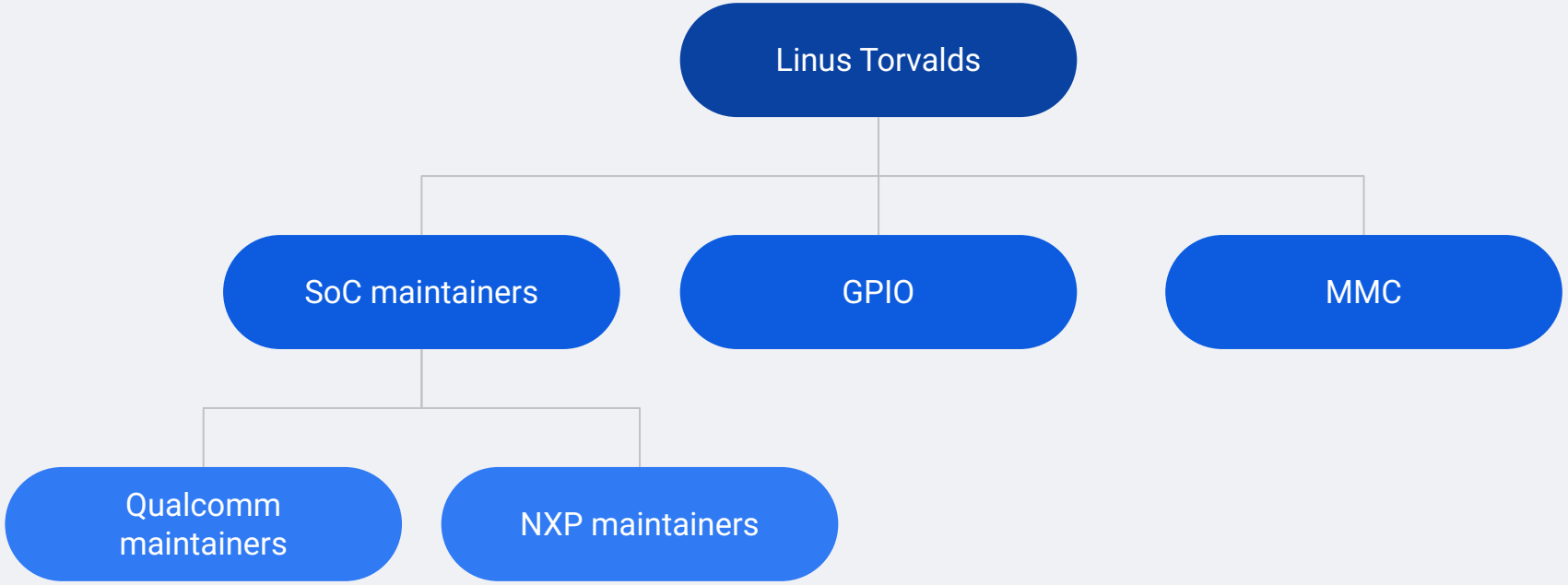
Maintainers Hierarchy Example



Maintainers Hierarchy Example



Maintainers Hierarchy Example



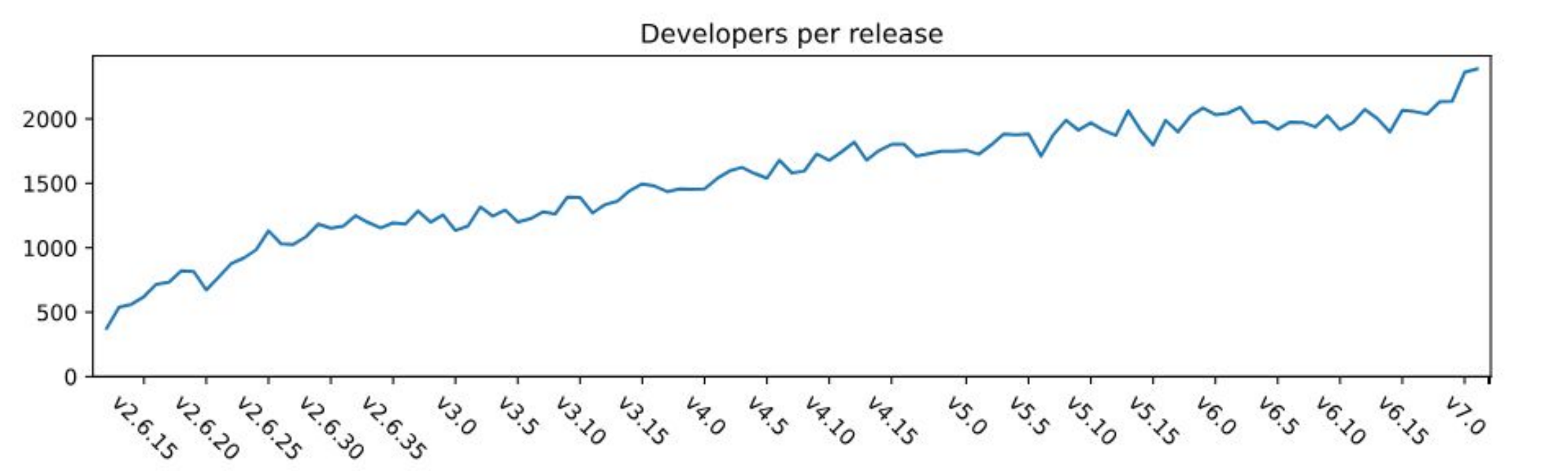
Driver Maintainers

- Maintainers responsible for specific drivers
- Usually not handling the patches, but providing only reviews and addressing bug reports

Why Do We Need Maintainers - Numbers

- Linux kernel it is not the biggest single software project …
  - Cloned repo has ~4.5 GiB (to compare with 68 GiB for Chromium)
- … but it is big
  - 1.4 million of commits
- … and developed fast, with a new release every two months:
  - ~14 000 regular commits
  - ~1300 merge commits
  - ~2300 individual authors contributing to each release
    - From around ~220 identified companies
  - Around 300 first-time contributors to each release
    - … actually trend might change: ~500 first-time contributors for v7.0 and v7.1
  - Source for most of the data here: https://lwn.net/ksdb/releases/

Why Do We Need Maintainers - Numbers



Source: https://lwn.net/ksdb/releases/

Why Do We Need Maintainers

- More contributors
- More submitted code
- More reviews and maintenance needed
- Business reasons:
  - We might remove code, which is not maintained
  - We might refuse code from companies, which repeatedly neglected their maintainership duties
  - Source: https://www.kernel.org/doc/html/latest/maintainer/feature-and-driver-maintainers.html#non-compliance
- But why am I talking about this?

Prerequisites

Contributions

- This is almost a cliché...
- But you need to be an active Linux kernel contributor
  - How to use email-based workflow
  - How to use Git
  - How the Linux kernel works
  - How the community process works (not only submitting-patches.rst)

Source: https://xkcd.com/1597 CC BY-NC 2.5

Trust

- All our interactions are based on trust
- Various maintainers have different trust level
  - Obviously Linus Torvalds looks differently at pull requests from Greg KH than some random person
- **Maybe the single most important factor?**
  - xz / Jia Tan
- Trust cannot be built fast
  - AI/LLM slop reviews won't help you - we will spot that
    - And might result in a big mess...
    - https://lwn.net/Articles/1077035/

Time

- Becoming a kernel maintainer was easy.
  Keeping the maintainership is the difficult part.

- You need to timely respond to mailing list activities
  - Within 1-2 weeks maximum
  - Within a few working days for some subsystems (e.g. netdev)

A Few Ideas

Path to Maintainership - Upstream

- Upstream new drivers or subsystem
- By default add yourself as the maintainer of that piece
- We want the code to be maintained, so don’t be shy
- It is not just a “title” - you still have duties
  - “*Subsystem maintainers may remove code for lacking maintenance.*” https://www.kernel.org/doc/html/latest/maintainer/feature-and-driver-maintainers.html#non-compliance
- Recently common issue around new SoCs support
  - Vendor submits new SoC which gets accepted via soc@ team
  - Vendor still thinks soc@ team will handle their patches
  - No, vendor is the maintainer now of that SoC, so it should handle the patches and send pull request to soc@

Path to Maintainership - Reviews

- We need more meaningful reviews
- Sashiko helps a lot, but it does not replace the community
- Great way to grow your skills
- Don’t nit-pick
  - Hypocrite speaks…
  - “*Reviewers are highly encouraged to do more in-depth review of submissions and not focus exclusively on process issues…*”
    https://www.kernel.org/doc/html/latest/process/maintainer-netdev.html

Path to Maintainership - Reviews

- Give meaningful review in a good faith
  - Even if you are not a pro
- But do not send bunch of “LGTM, Reviewed-by …”
  - You will not build trust
- Review before subsystem maintainer, to offload him
  - Nothing more annoying than to see “LGTM” after the maintainer did all the heavy-lifting review

Path to Maintainership - Reviews - Toolset

- Get emails to your favorite mail client
  - lei + lore: https://people.kernel.org/monsieuricon/lore-lei-part-1-getting-started
  - korgalore: https://korgalore.docs.kernel.org/
- b4 review
  - https://b4.docs.kernel.org/en/latest/reviewer/getting-started.html
- Understand “Reviewer’s statement of oversight”
  - https://www.kernel.org/doc/html/latest/process/submitting-patches.html#reviewer-s-statement-of-oversight

Path to Maintainership - Orphaned Subsystems

- If you have trust and time
- There are a few orphaned subsystems which need help
  - Not giving you the list publicly (but we do have it)

Any More Ideas?

- Any more ideas?

Summarizing

- Be involved and build trust
- Don’t cut corners
- Don’t be shy

Thank you

Krzysztof Kozlowski
krzk@kernel.org, @krzk@social.kernel.org

References

- Kernel Maintainer Handbook: https://www.kernel.org/doc/html/latest/maintainer/index.html
- Maintainer subsystem profiles: https://www.kernel.org/doc/html/latest/maintainer/maintainer-entry-profile.html
- KSDB: https://lwn.net/ksdb/releases/
- Development statistics for the v7.1 kernel: https://lwn.net/Articles/1077425/
- Beginner Linux kernel maintainer's toolbox (LPC 2023) https://lpc.events/event/17/contributions/1498/
- lei + lore: https://people.kernel.org/monsieuricon/lore-lei-part-1-getting-started
- korgalore: https://korgalore.docs.kernel.org/
- b4 review: https://b4.docs.kernel.org/en/latest/reviewer/getting-started.html
- Reviewer’s statement of oversight https://www.kernel.org/doc/html/latest/process/submitting-patches.html#reviewer-s-statement-of-oversight

Thank you

For more information, visit us at qualcomm.com & qualcomm.com/blog