Upstreaming 101

Fundamentals

Linaro

Who Am I? Why Listen To Me?

- Broadcom LT Technical Lead
  - Upstreaming BCM281xx SoC
  - Mentoring others on upstreaming
- Upstream kernel experience
  - 15 years kernel experience
  - Kernel developer at Motorola, MontaVista, Embedded Alley, TI, and Linaro.
  - Developed and maintained PowerPC VME/cPCI/4xx, RapidIO, and SigmaTel ALSA HDA codecs in the mainline kernel.
  - Upstream driver work for TI AM335x and BCM281xx ARM SoCs.

Overview

- Target audience
  - Developers
  - Engineering managers
- Focus is on Linux kernel upstreaming
- What is upstreaming?
  - Define what it is first
- How to upstream?
  - Process and Mechanics

Prerequisites

- Familiar with source code control concepts
- Familiar with git terminology (pulls, topic branches, etc.)
- Technical understanding of kernel level software

What is upstreaming?

- Linux kernel context
- Upstream means to move software into the top level Linux repository
- This is Linus Torvalds' Linux repository (aka “mainline”)

The Linux Kernel Archives

About Contact us FAQ Releases Signatures Site news

Protocol	Location
HTTP	https://www.kernel.org/pub/
FTP	ftp://ftp.kernel.org/pub/
RSYNC	rsync://rsync.kernel.org/pub/

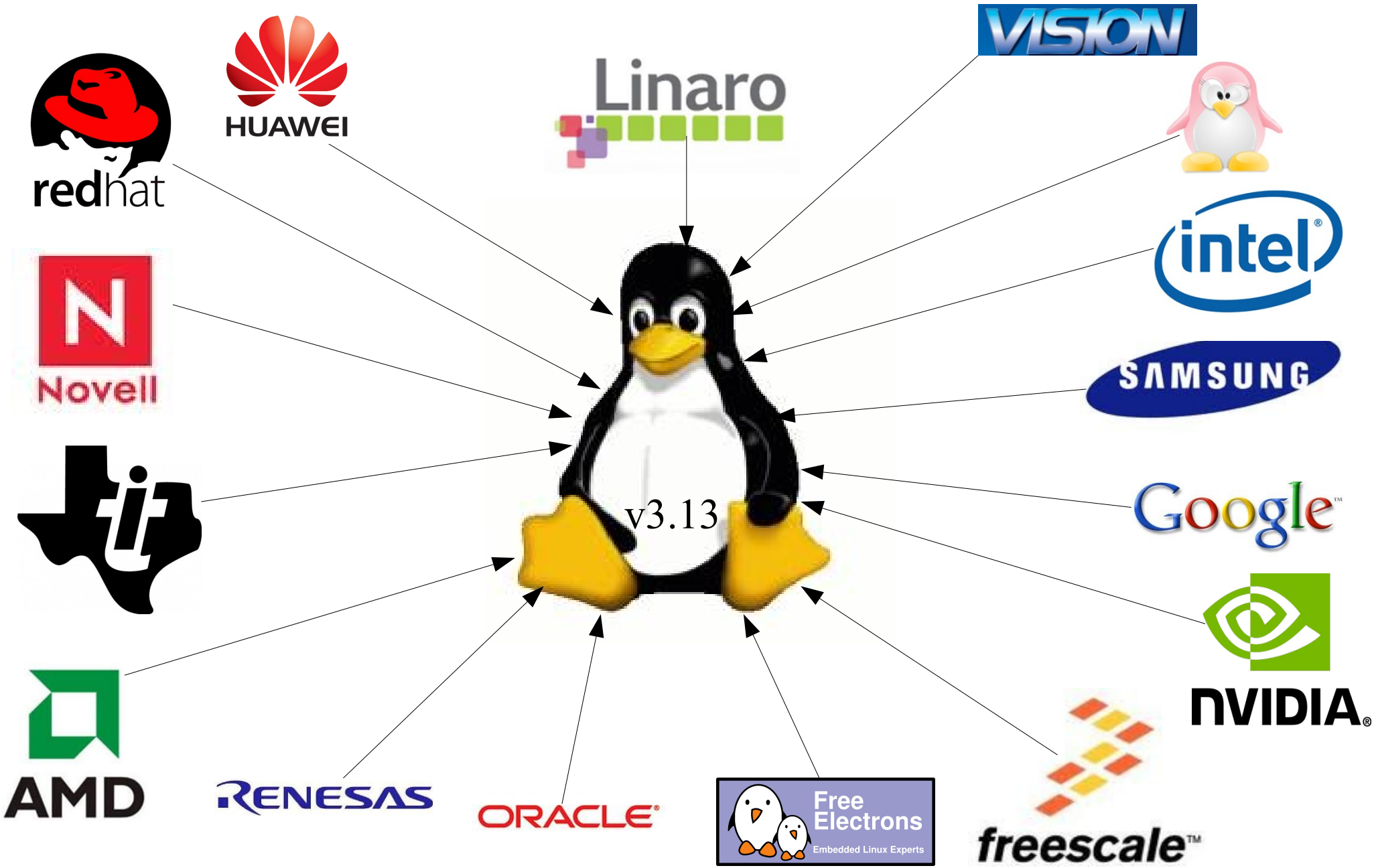
Latest Stable Kernel:

3.12.6

mainline:	**3.13-rc7**	2014-01-04	[tar.xz]	[pgp]	[patch]	[view patch]		[cgit]
stable:	**3.12.6**	2013-12-20	[tar.xz]	[pgp]	[patch]	[view patch]	[view inc]	[cgit] [changelog]
stable:	**3.11.10 [EOL]**	2013-11-29	[tar.xz]	[pgp]	[patch]	[view patch]	[view inc]	[cgit] [changelog]
longterm:	**3.10.25**	2013-12-20	[tar.xz]	[pgp]	[patch]	[view patch]	[view inc]	[cgit] [changelog]
longterm:	**3.4.76**	2014-01-08	[tar.xz]	[pgp]	[patch]	[view patch]	[view inc]	[cgit] [changelog]
longterm:	**3.2.54**	2014-01-03	[tar.xz]	[pgp]	[patch]	[view patch]	[view inc]	[cgit] [changelog]
longterm:	**3.0.101 [EOL]**	2013-10-22	[tar.xz]	[pgp]	[patch]	[view patch]	[view inc]	[cgit] [changelog]
longterm:	**2.6.34.14**	2013-01-16	[tar.xz]	[pgp]	[patch]	[view patch]	[view inc]	[cgit] [changelog]
longterm:	**2.6.32.61**	2013-06-10	[tar.xz]	[pgp]	[patch]	[view patch]	[view inc]	[cgit] [changelog]
linux-next:	**next-20140108**	2014-01-08						[cgit]

Who Exactly Contributes to Mainline?

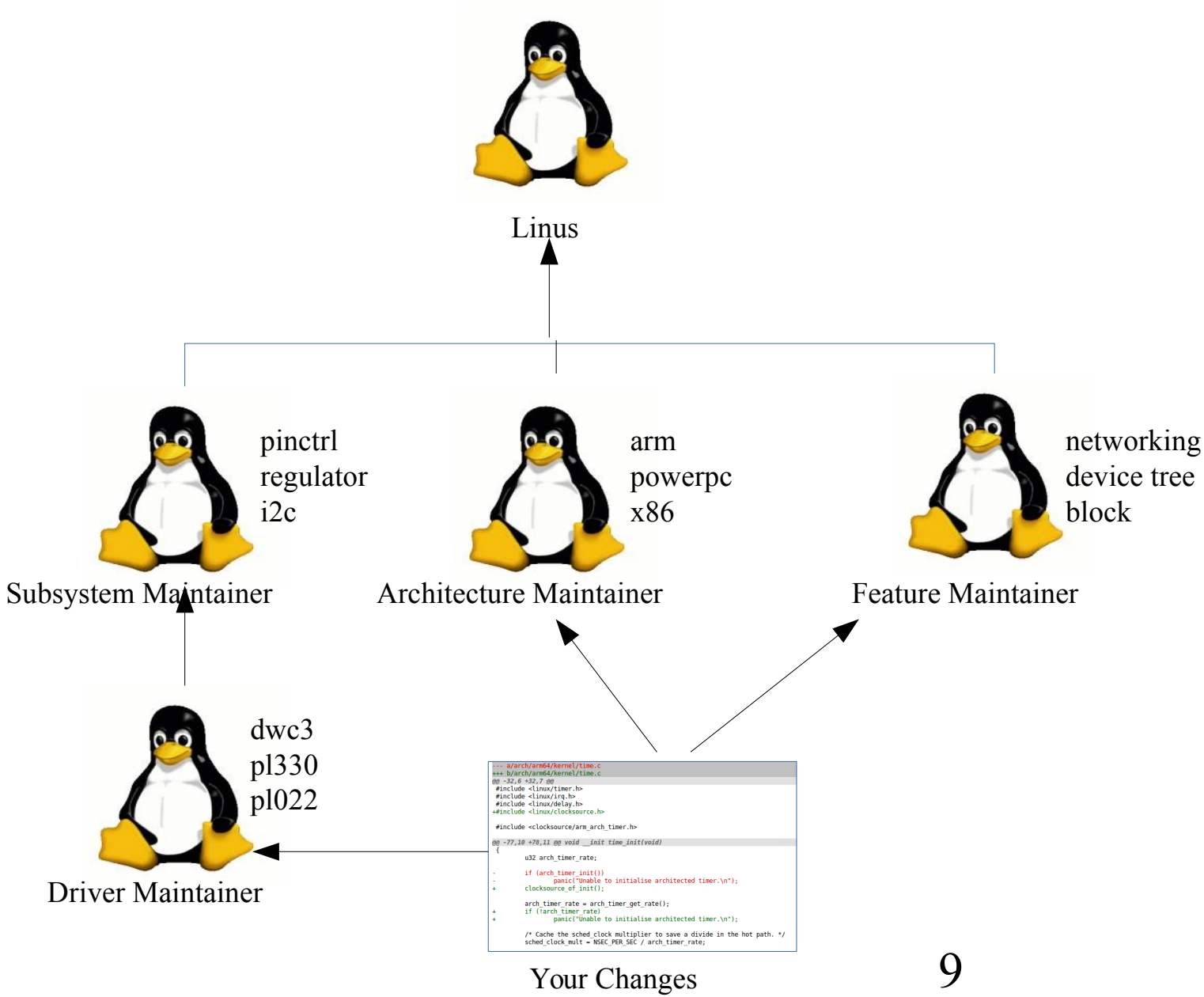
(From list of top 3.13 contributors: http://lwn.net/Articles/579081/)



Swimming upstream to mainline

- Distinct hierarchy of repositories
- Repositories are git trees
  - One or more topic branches that feed into the mainline kernel
- Different owners for each repository in the tree

Upstream Code Flow



Maintainers

- Component code owners
  - Subsystem
  - Driver(s)
  - Filesystem
  - Architecture/Platform
- Responsible for a slice of the kernel source tree
- Gatekeepers
  - Control acceptance of incoming patches
  - Acceptance criteria can vary

Maintainer Numbers

- 907 unique maintainers

  $ egrep "^M:.*" MAINTAINERS | sort | uniq | wc -l

  907

- Each subsystem/component has one or more maintainers
- Example MAINTAINERS entry:

  ARM PORT

  M: Russell King <linux@arm.linux.org.uk>

  L: linux-arm-kernel@lists.infradead.org (moderated for non-subscribers)

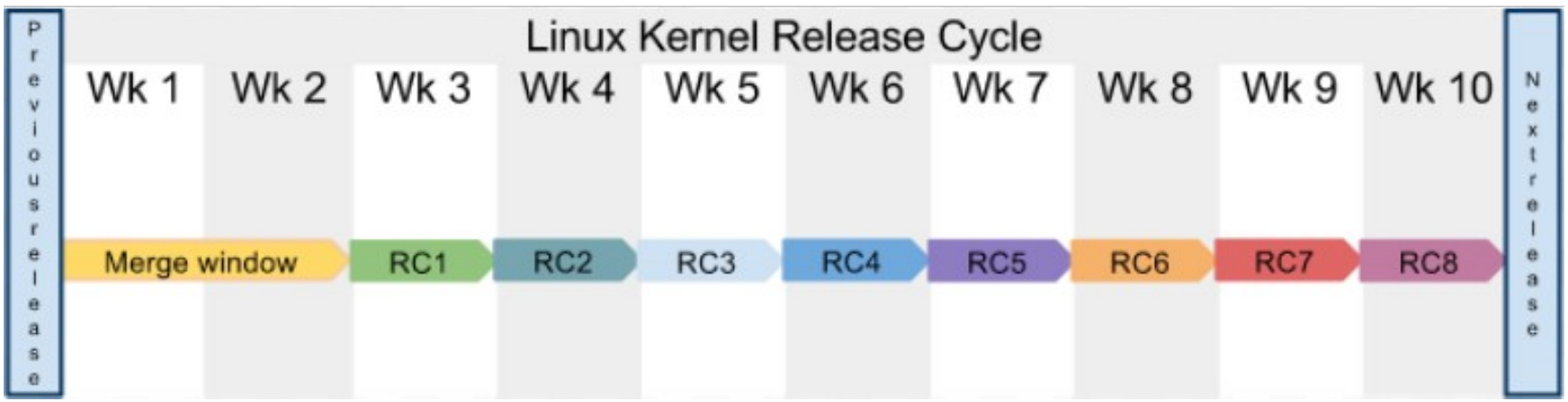
  W: http://www.arm.linux.org.uk/

  S: Maintained

  F: arch/arm/

Understanding Merge Windows

- Merge windows open every 10 weeks +/- 1 week
- Merge window is open for 2 weeks
- New functionality is only taken into Linus Torvalds' tree during the merge window



Understanding merge windows

- Merge window planning
  - New functionality needs to be accepted in maintainer trees usually by the -rc6 or -rc7 release
  - After -rc7 most maintainers will only be accepting fixes
- Less than 7 weeks after a merge window closes to have a maintainer queue a patch for the next merge window.

How to Upstream?

- Preparation
- Creation
- Posting
- Feedback
- Maintenance
- How Long Does it Take?

Preparation

- Know your content
  - Your contribution fits into a kernel framework. What is it?
  - Write your contribution to conform to the current framework standards and kernel APIs
- Know who else is doing work in your area upstream
  - Is anybody doing work related to the framework that could affect framework APIs?

Preparation

- Review Documentation/* for clarification on APIs and frameworks
- Review Documentation/devicetree/bindings/* for clarification on Device Tree bindings and best examples.
- Read *devicetree* mailing list to learn about DT best practices
  - http://vger.kernel.org/vger-lists.html#devicetree

Preparation

- On what mailing lists and IRC channels are similar contributions discussed?
  - Follow these forums and understand the direction the frameworks are moving in APIs and style.
  - Ask questions, if necessary, to clarify what APIs to make use of before writing your code.
- Read *linux-arm-kernel*, at a minimum
  - http://lists.infradead.org/mailman/listinfo/linux-arm-kernel
- *#armlinux* on freenode for ARM kernel discussions

Creation

- Read and understand
  - Documentation/SubmittingPatches
  - Documentation/SubmitChecklist
  - Documentation/devicetree/bindings/ABI.txt
  - .../devicetree/bindings/submitting-patches.txt

Creation

- Use git for code management
- Logical division of commits
  - Small changes
  - Functionality
  - Individually complete (bisectability)
- Logical commits allow for ease of review and speed acceptance

Creation

- Use the proper subject line
  - *Subject: [PATCH 01/11] subsystem: summary phrase*
  - *Subject: [PATCH v3] subsystem: summary phrase*
  - *Subject: [PATCH RFC] subsystem: summary phrase*

Creation

- Take time to create a quality commit log message
  - Why the patch is needed
  - What the patch implements
  - How the patch is implemented.

    "The conditional in foo() did not handle case bar and broke platform baz. Add an additional conditional and error path to foo() to handle bar."
- Each commit must have a well-formed log

Creation

- Create patches with *git format-patch*
  - *--cover-letter* for a patch series
  - The cover letter contains an overview describing the purpose and scope of the entire series.
- Use *scripts/checkpatch.pl* to verify coding style and semantics
- Use *scripts/get_maintainer.pl* to verify maintainer list for submission.

Posting

- Post patch or patch series
  - Maintainers on To:
  - Mailing lists on Cc:
  - Other interested parties on Cc: (other commiters to functionality)
- Use *git send-email* to post patches/series
- Expect comments!

Feedback on Mailing Lists

- No response
  - Be patient, maintainers are very busy
  - Wait one week to resend if no response
- Tough questions
  - Be prepared to justify your decisions or approach in great detail
  - Maintainers aren't always correct, be strong and concise in your justifications.
  - If you don't understand a comment, ask for clarification!

Feedback on Mailing Lists

- Mailers
  - Use a sane mail user agent like mutt
  - Wrap at 72 columns
- Getting flamed
  - No need to worry about this if you are following the documented practices.

Feedback on Mailing Lists

- Making changes
  - Be responsive! Address comments via discussion and come to a mutual conclusion quickly
  - Incorporate agreed upon comments and quickly submit
  - Be prepared to not get an acceptable comment resolution on the first try
  - Expect **many** iterations
- Resubmission
  - Increment the version number in the subject line for the patch series

Maintenance

- Once accepted, now what?
  - Need to follow mailing lists for upcoming changes
  - Help review any new changes within the same area as your contribution
  - Test, test, test

Summary

- Preparation is key to success
- RTFM on everything
- Ask questions
- Act with a sense of urgency on comments
- Understand merge window timing

Questions?

Are LTS/LTSI/LSK upstream?

- Long Term Stable is a maintenance tree
  - Stable tree picked yearly to be LTS and maintained by Greg KH.
  - http://kroah.com/log/blog/2013/08/04/longterm-kernel-3-dot-10/
- Long Term Stable Initiative is a derivative of LTS that allows vendors to commit patches that are not ready for mainline
  - http://ltsi.linuxfoundation.org/what-is-ltsi
- Linaro Stable Kernel is a production-oriented derivative of LTS that contains not-yet upstream Linaro sponsored features
  - https://wiki.linaro.org/LSK