Overview

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

https://kernel.org
Who Exactly Contributes to Mainline?

from list of top 4.4 contributors: http://www.remword.com/kps_result/4.4_whole.html
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 tree
- Gatekeepers
  - Control acceptance of incoming patches
  - Acceptance criteria varies
Maintainer numbers

- 1087 unique maintainers in v4.4
  
  ```bash
  $ grep "^M:.*/" MAINTAINERS | sort | uniq | wc -l
  1033
  ```
  
- 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 ...
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](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
- #armlinux on freenode for ARM kernel discussions
Preparation

- Read and understand
  - Documentation/SubmittingPatches
  - Documentation/SubmitChecklist
  - Documentation/devicetree/bindings/ABI.txt
  - .../devicetree/bindings/submitting-patches.txt
  - Greg Kroah-Hartman, "How to piss off a kernel subsystem maintainer".
    - http://www.kroah.com/log/linux/maintainer-03.html
    - http://www.kroah.com/log/linux/maintainer-06.html
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

- Multipart series subject line
  - Subject: [PATCH 01/11] subsystem: summary phrase
- Version 3 of a single patch submission
  - Subject: [PATCH v3] subsystem: summary phrase
- RFC patch submission
  - 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 commit 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:
- 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
  ○ Don’t ignore comments!
Feedback on Mailing Lists

● Use a sane email client
  ○ Plain text wrapped at 72 columns (unless it’s a diff)
    ■ Working threading
    ■ Saves messages in a format git understands
  ○ Advice on configuring various mail user agents
    ■ Documentation/email-clients.txt

● 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 conclusion quickly
  ○ Incorporate agreed upon comments and quickly submit a new version
  ○ 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 and include a changelog
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
Linaro Limited Lifetime Warranty

This training presentation comes with a lifetime warranty.

Everyone here today can send any questions about today’s session, at any point in the future, to support@linaro.org.

Members can also use this address to get support on any other Linaro output. Engineers from club and core members can also contact support to discuss how premium services can help you with additional services and training.

Thanks to Andrew Hennigan for introducing me to the idea of placing a guarantee on training.
Thank You

#LAS16

For further information: www.linaro.org or support@linaro.org

LAS16 keynotes and videos on: connect.linaro.org
Preview of 201 - Tools and techniques

- Filtering mailing lists
- Source navigation
- Static checkers
- Handling regressions
- Bisectability testing
Filtering mailing lists

Why?
It is impossible to read **everything** posted to LKML - “like drinking from a fire hose”... ... but if you don’t read **anything** on the mailing lists you will never absorb enough kernel culture

How?
Subscribe anyway and employ **really** aggressive filters to focus on things that interest you
Filtering mailing lists - Good habits

- Have a whitelist - delete **everything** else
  - Don’t hoard old messages... search engines can find anything you miss
- Create a filter to keep an eye on code you have written
  - It’s good for your reputation to keep an eye on things you write
  - You can use a highly focused filter here... only care about your driver/code
- Create a filter to keep an eye on what you are working on now
  - Review comments from similar drivers will now cross your desk automatically
  - Use `git log drivers/of/interest.c` to get some idea of useful subject lines to filter
  - Create a generic filter here... you want to see what other people are doing right/wrong
- Still fairly high volume
  - Keep the mail part of your normal mailbox flow
  - Colourize/mark mail from the list so your normal mail stands out compared to ML traffic
  - Make sure **only** ML traffic is tagged (i.e. if you **and** ML are in To/Cc: do **not** tag like ML)
  - For tagged mail, press Delete straight away if the subject line doesn’t interest you
Filtering mailing lists - Special gmail account

- LKML and most other Linux lists do **not** require you to be a member in order to post them
  - You don’t need to send and receive messages from same e-mail account
- Create a special purpose gmail account to pre-filter mail
  - Register your work e-mail as a forwarding address in gmail
  - Write filters to forward targeted messages to your work e-mail
  - After forwarding, filter everything else to the Trash (where it will be deleted after 30 days)

**Advantages**
- Easy to colourize ML messages: if “X-Forwarded-For” does not contain “mygmailname”
- Never conflict with messages which include you in To:/Cc: (should be more obvious that ML)
- Even with auto-delete set you can still search the last 30 days of traffic to quickly dig out and save patchsets. Easier than copy ‘n paste from archival web sites

**Disadvantages**
- Initial setup is more complex
Socialising on IRC

IRC channels on the freenode network is a good place to get informal contact with developers and maintainers (during periods when timezones overlap).

Start with #armlinux.

Finding channels of interest can be hard (IRC is not listed in MAINTAINERS). Try: /msg alis list #linux-m*

It’s OK to sit and “watch” until you feel confident with the culture.

Don’t ask permission to ask a question. If you have a question, just ask it.

Great place to follow up on code reviews. For example: If reviewers idea won’t work but you can think of something similar and want to discuss that.
Thank You

#LAS16

For further information: www.linaro.org or support@linaro.org
LAS16 keynotes and videos on: connect.linaro.org