# arm-soc

Silly kernel hackers! Socks are for feet.

Olof Johansson, Google

## Mandatory ARM Linux history slide

- Maintained by Russell King since the dawn of time
- Patch volume from platform vendors grew too large to keep up with
- Around 2009, platform maintainers started merging straight to Linus
- Famous March 2011 blow-up from Linus
- arm-soc started by Arnd Bergmann in July 2011
- I joined in November 2011
- As of August 2012, Linus was generally happy with the state of affairs

## What problems does arm-soc solve?

- Lack of coordination and sharing between platforms
- Keeping a code quality bar
- Code review of merged code
- Bringing online new platform maintainers
- Gives a common view of what's going on across the various vendors

## "The board file mess"

- One of the major things Linus was unhappy with
- Lack of standardized software platform (c.f. x86 ACPI)
- Hardware vendors try to differentiate by hardware design
- This spills over into unnecessary software differentiation
- Resulting in
  - Code and infrastructure duplication
  - Per-platform abstractions (OMAP hwmod, etc)
  - ...or worse, lack of abstraction all together

## "The board file mess": Solutions?

- Device tree conversion
  - Remove need for code changes for minor (or major) hardware changes
- Single zImage
  - Solves a different problem (for distros)
  - Has required a lot of very useful cleanups
- UEFI/ACPI?
  - Greener grass? Painted gravel?

## "The board file mess"

In reality a combination of all of the above is happening

- Lots of code refactoring and cleanup
  - Splitting out to proper drivers and subsystems
- Separating hardware description from code
  - DT, ACPI
- The road has been long and somewhat painful
- ... and still ongoing

- Lots of cleanup, but also lots of churn

## Churn, you say?

3.8-rc1 announcement from Linus:

*"18% was architecture updates (with various ARM platforms being the bulk of it as usual, sigh)."*

Two years later, are we getting in trouble again?

## 3.8 merge window

- Large churn due to header file moves
- Unusual number of internal conflicts
  - This is mostly something for me and Arnd to deal with
- Holidays and vacations, including Stephen Rothwell
  - Code merged during that time didn't get early notice of conflicts with other trees
- ARM (and dts) code merged through other maintainers
  - Platform maintainers weren't even cc:d
  - ...but sfr noticed the conflicts and so should we

## All doom and gloom? No.

- We're still doing well
- Tweaking our internal merge resolution
- Keeping a closer eye on sfr's merge conflict emails
- Some of the larger code moves are nearing completion
  - Single zImage in particular
- Code quality is still good

...but we need to keep an eye on the long game

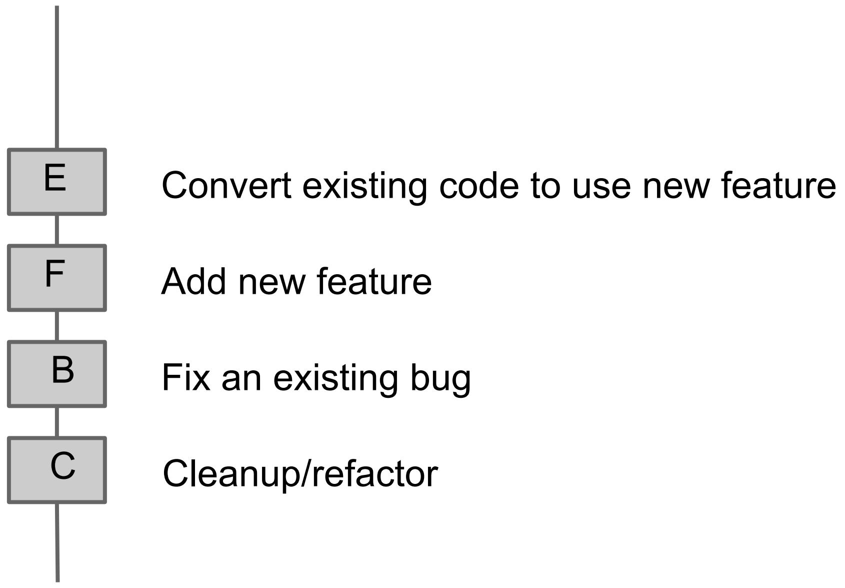
## So, you have an ARM platform to upstream?

- Expect a bit of learning curve
  - Even for experienced maintainers of other areas
  - Be flexible and expect to shuffle patches around
- Non-linear tree organization and merge structure
  - Not just one large branch that merges everything
  - Because of this, developer and maintainer workflow differs more than some other subsystems
  - Categories vs topics

## Topic (feature) branches

- Inherent to developer workflow
- Keeps related changes together
- Usually builds up several short series of patches
- Independent series kept on separate branches
- Posting of patch series for review, etc
- Keeping series as one unit is useful for testing

## Typical topic branch



## arm-soc categories

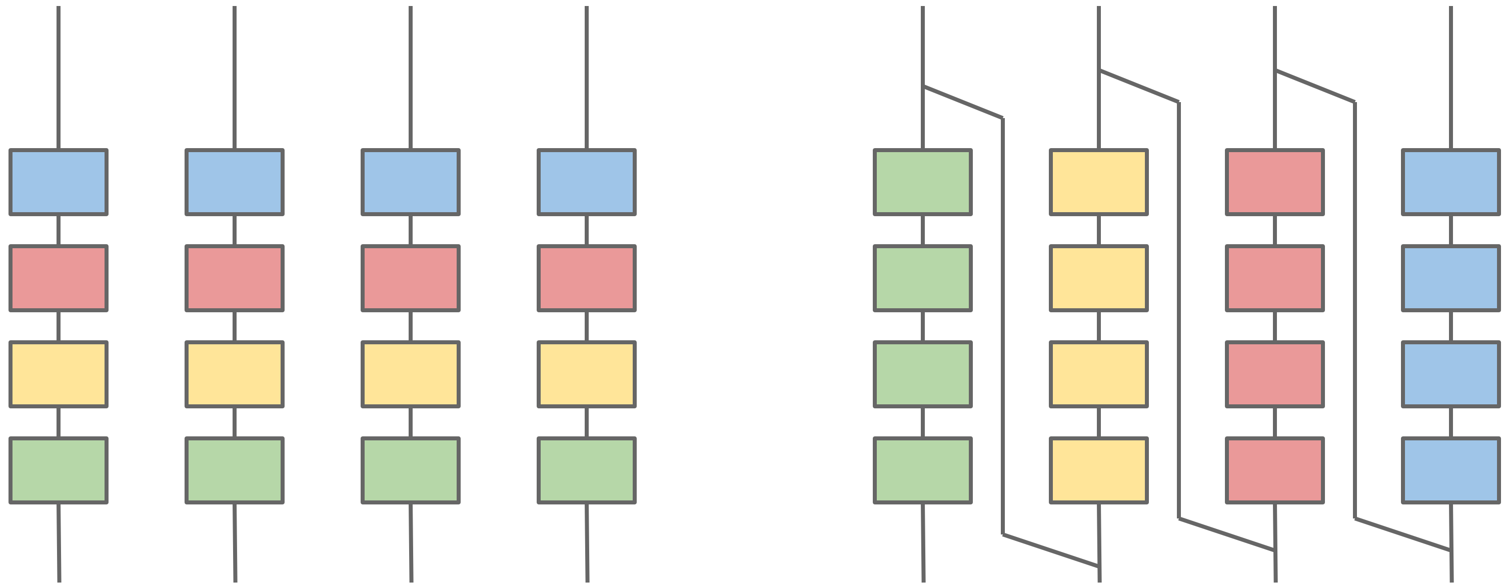
- Used instead of topics
- Top-level branch organization
- Usually consists of:

```
next/fixes-non-critical
next/cleanup
next/multiplatform
next/soc
next/drivers
next/boards
next/dt
...
```

## Mapping topics into categories

- Done by the platform maintainers
- Useful for developers to know
  - Organizing your patch series appropriately
- Maintainer splits a series into the categories when applying
  - Cleanup patches go into cleanups, fixes to fixes, SoC common changes to soc branch, etc etc
- Base branches on top of each other
  - Allows for dependent patches to go in separate categories
  - Avoid circular dependencies!

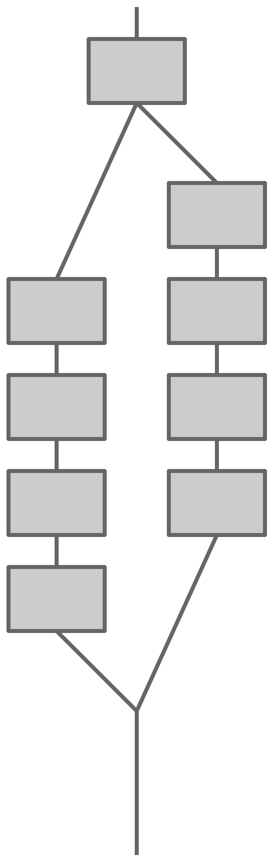
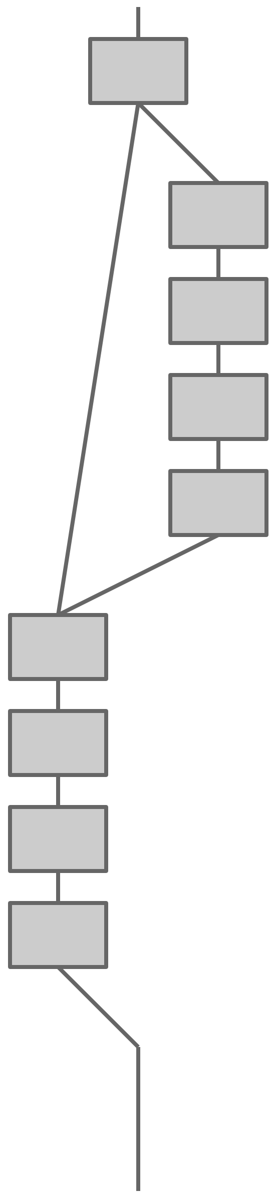
## Mapping topics into categories

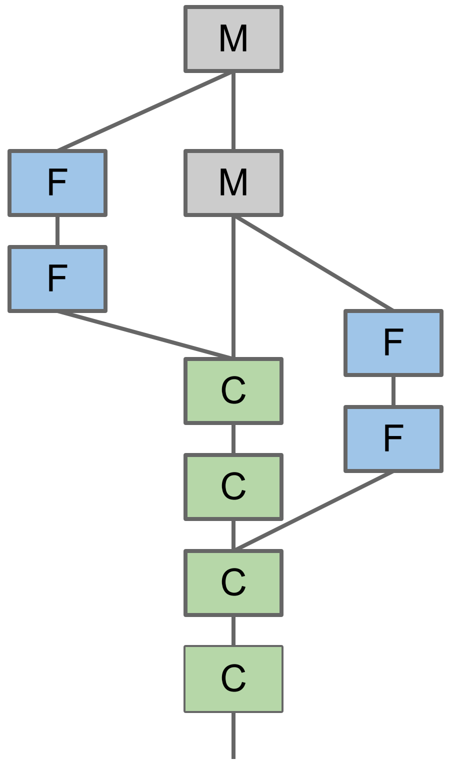


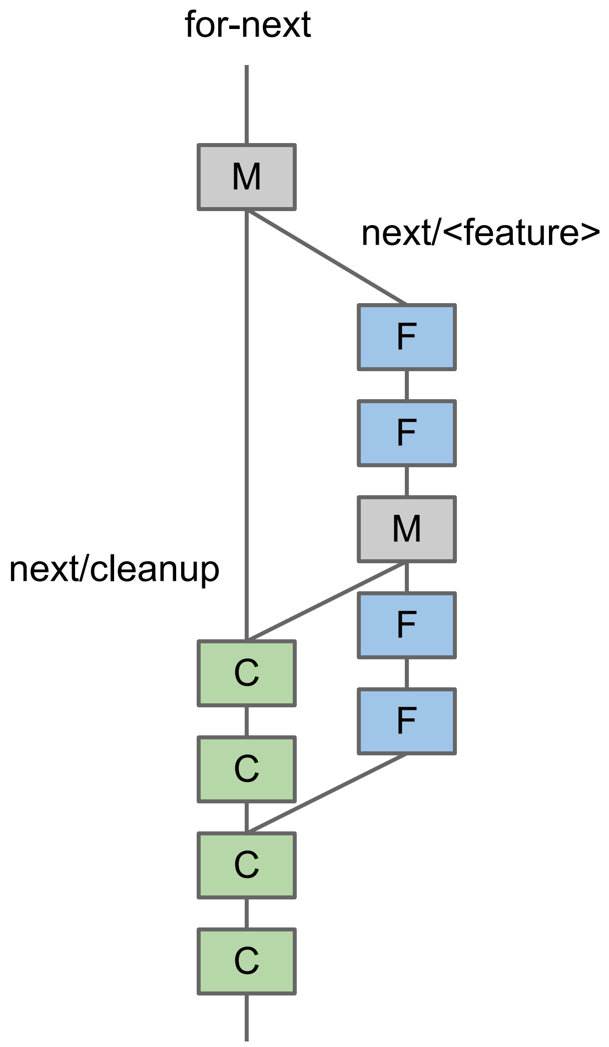
## A few words about bisectability

- Be careful to keep bisectability across branches

- Linear history vs branches





| | |
|---|---|
| C: | cleanup |
| F: | feature |
| M: | merge commit |

## More complicated cases

- In reality, some patches go through other trees

- How to handle dependencies?
  - Device tree conversions
  - Adding platform data contents
  - New drivers that need platform plumbing

## General rules of dependencies

- Adding an external dependency is a **three-way handshake**
  - You (platform maintainer)
  - Other subsystem maintainer
  - arm-soc maintainers
- Always do it over email, not IRC
- Patches need to be on a 100% stable branch
  - **Never, ever rebased**
- Pulled into both trees (driver + arm-soc)
- Might merge up through arm-soc if we merge first

## External dependencies: New driver

- Easy case
- Driver patch goes through driver maintainer
  - Or, have him give you an Acked-by with agreement to merge through arm-soc
  - Preference varies between maintainers
  - Depends on what other work they have going on with their own tree
- Add DT entries in branch through arm-soc
- Bisectability should be preserved
  - Even if driver and DT is merged separately, bisectability is kept -- driver just won't probe

## External dependencies: DT conversion

- Doesn't have to be complicated
- Driver patch to maintainer to fill platform_data from device tree
- DT update through arm-soc
- Keep platform_device/data for one release
    - Avoids extra dependencies
- Next release, remove platform_device registration
    - Everybody loves code removal!

## External dependencies: platform data

- This can be messy
- Same as new driver: If maintainer acks, merge through arm-soc
- If you can easily make the driver work with both new and old data, do it
    - Avoids dependency, remove fallback in next release
- If not, stage a patch that adds structure members in a separate branch
    - Base driver patch on this for the driver tree
    - Merged into arm-soc
- Use your judgement on which approach to take

## Dos and Dont's

- Want us to apply a patch directly? Tell us, don't assume we will
  - We get a lot of patches our way, most for review
- Send pull requests early
- Ask downstream users to use our tree
  - Avoids locking in your downstream users before code is accepted
  - Repeated pull requests of the same branches are fine
- Send your pull requests using signed tags
  - Provides a place to document the contents and makes it easy for us to include in the merge.
- Test arm-soc for-next branch and linux-next!
  - Short-circuits the loop on breakage
  - Keep your platform bisectable

## Summary

- arm-soc and ARM platform development is still going strong
- Need to be diligent about merge paths to avoid conflicts
  - Make sure your developers know where to submit code!
- Getting used to our merge flow might take a couple of releases
  - Study existing maintainers workflow
  - Avoid downstream direct users until you're more familiar

## Still awake?

Questions?

## What will be new in 3.9

IRQ controller and timer cleanups
OMAP2+ multiplatform
WM8x50 support
Tegra T114 support
Samsung header moves towards multiplatform
shmobile switches to pinctrl
zynq support for real hardware (SMP)