A Guide to the Linux Kernel Development Process

Jonathan Corbet
LWN.net
corbet@lwn.net

Agenda

Why participation matters
Guiding principles
Trees
Some tips

For more information

ldn.linuxfoundation.org/book/
how-participate-linux-community

-- or --

Documentation/development-process/

Why?

Why?

The kernel is the core of a Linux system

Why?

It's how you get the kernel to meet your needs

> “Well, this is open source... you don't get to request new features, you get to implement them”
>
> -- James Bottomley

Why?

External code is expensive

Why?

External code is lower quality code

Why?

In-tree code can be improved by others

Why?

That is how our community works

Why this talk?

Working with the kernel is not hard

...if you understand how the process works

“So, I've had enough. I'm out of here forever. I want to leave before I get so disgruntled that I end up using Windows. I may play occasionally with userspace code but for me the kernel is a black hole that I don't want to enter the event horizon of again”

-- Con Kolivas

Some guiding principles

...which should help in understanding how the kernel is made

Upstream first

Code goes into the kernel first

- ...before going to customers
- ...before user space depends on it

Not a differentiator

Vendors should not differentiate their offerings at the kernel level

Not a differentiator

Vendors should not differentiate their offerings at the kernel level

(See “Upstream first”)

Technical quality over all

Code quality outweighs:

- Company plans
- User desires
- Existing practice
- Developer status

Long-term view

Kernel developers expect to be maintaining the code 5-10 years from now

Peer review

No code is perfect

- it can always be improved
- heed requests for changes

Developers are individuals

...separate from their employers

No ownership of code

Even code you wrote

No regressions

...even to fix other problems

> “So we don't fix bugs by introducing new problems. That way lies madness, and nobody ever knows if you actually make any real progress at all. Is it two steps forwards, one step back, or one step forward and two steps back?”
> -- Linus Torvalds

No inherent right to inclusion

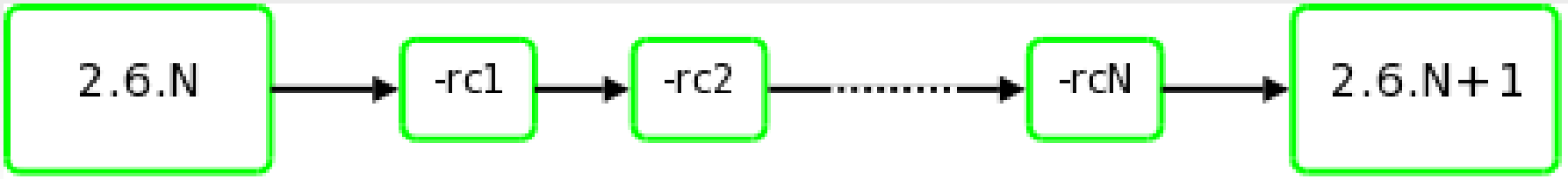
Changes require justification
Other solutions may win out

Trees

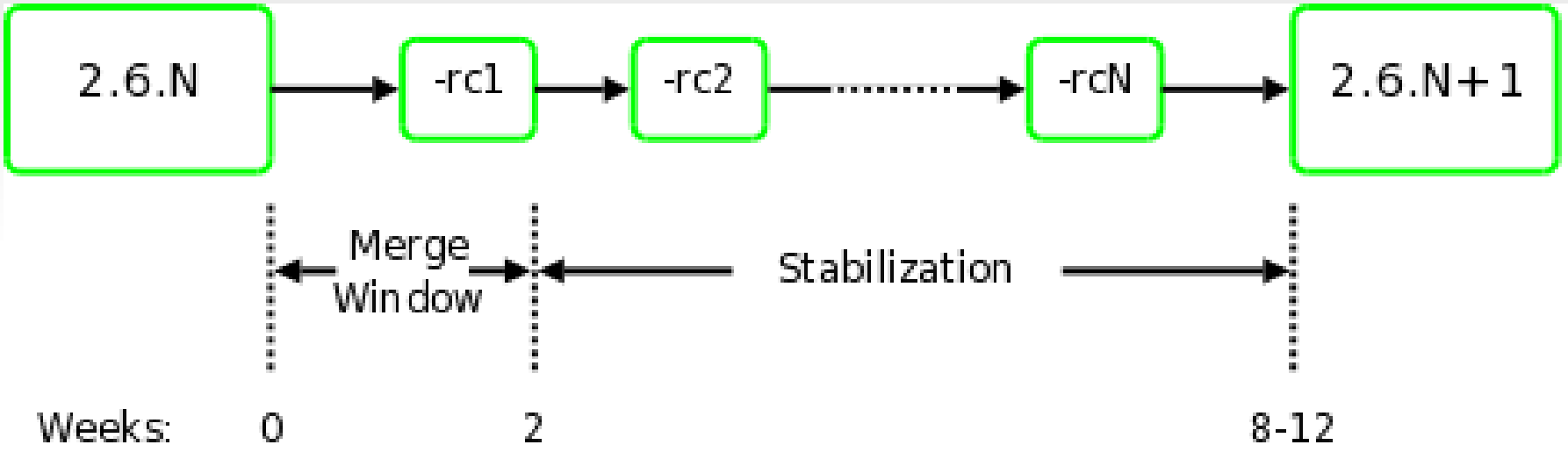
Mainline kernel

- Linus Torvalds's kernel
  - 2-3 month release cycle

The release cycle



The release cycle



-stable

Important updates to the mainline

- Security fixes
- Severe bugs
- Maintained for ~6 months

Distributor kernels

Based on -stable

May include significant changes

- “enterprise” kernels especially

Maintenance period varies

Development trees

linux-next

- Staging area for the next mainline cycle
- Patch integration
- Early testing

-mm

- Now based on linux-next
- Collection point for miscellaneous patches
- More early testing

Subsystem trees

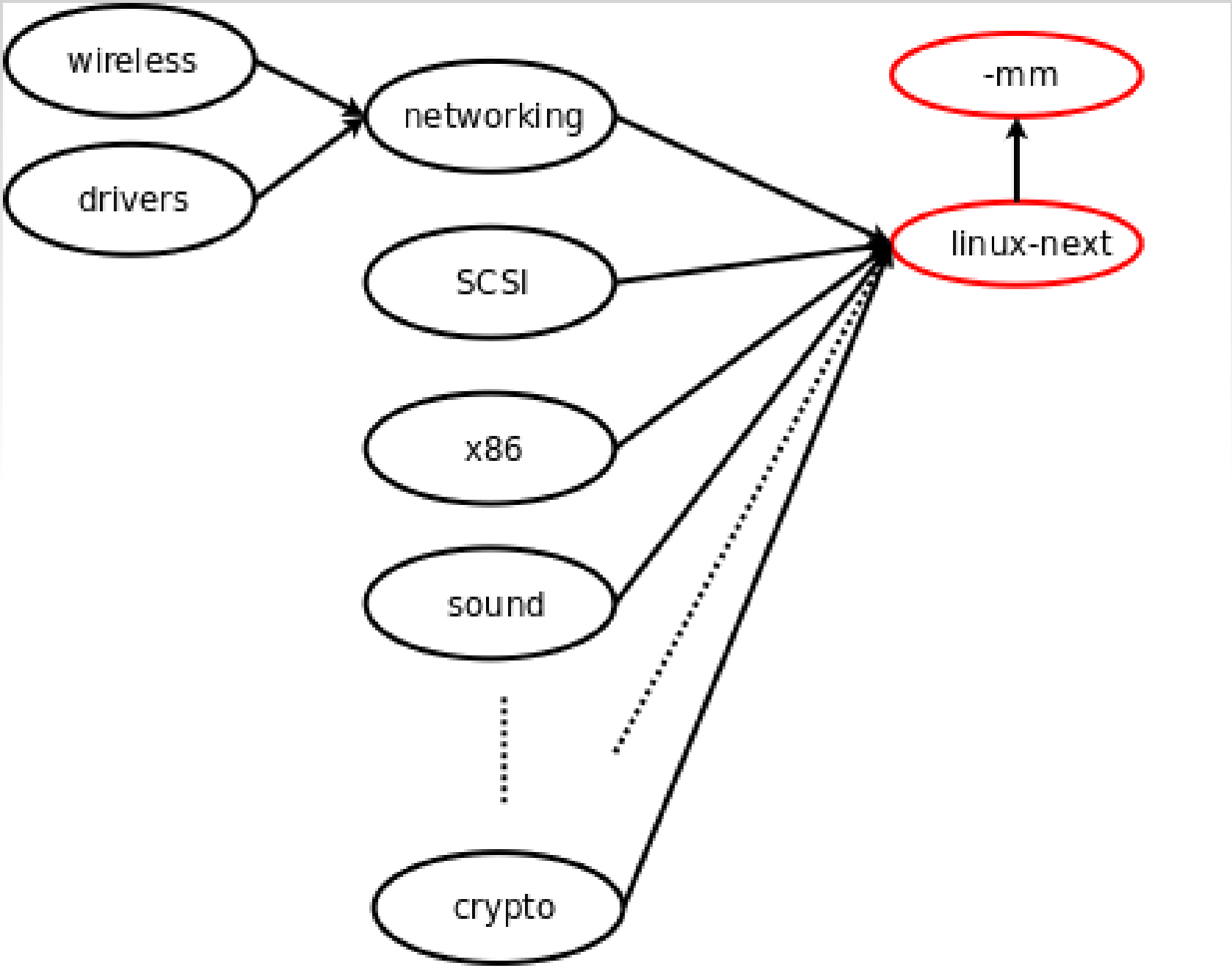
Development for a single subsystem

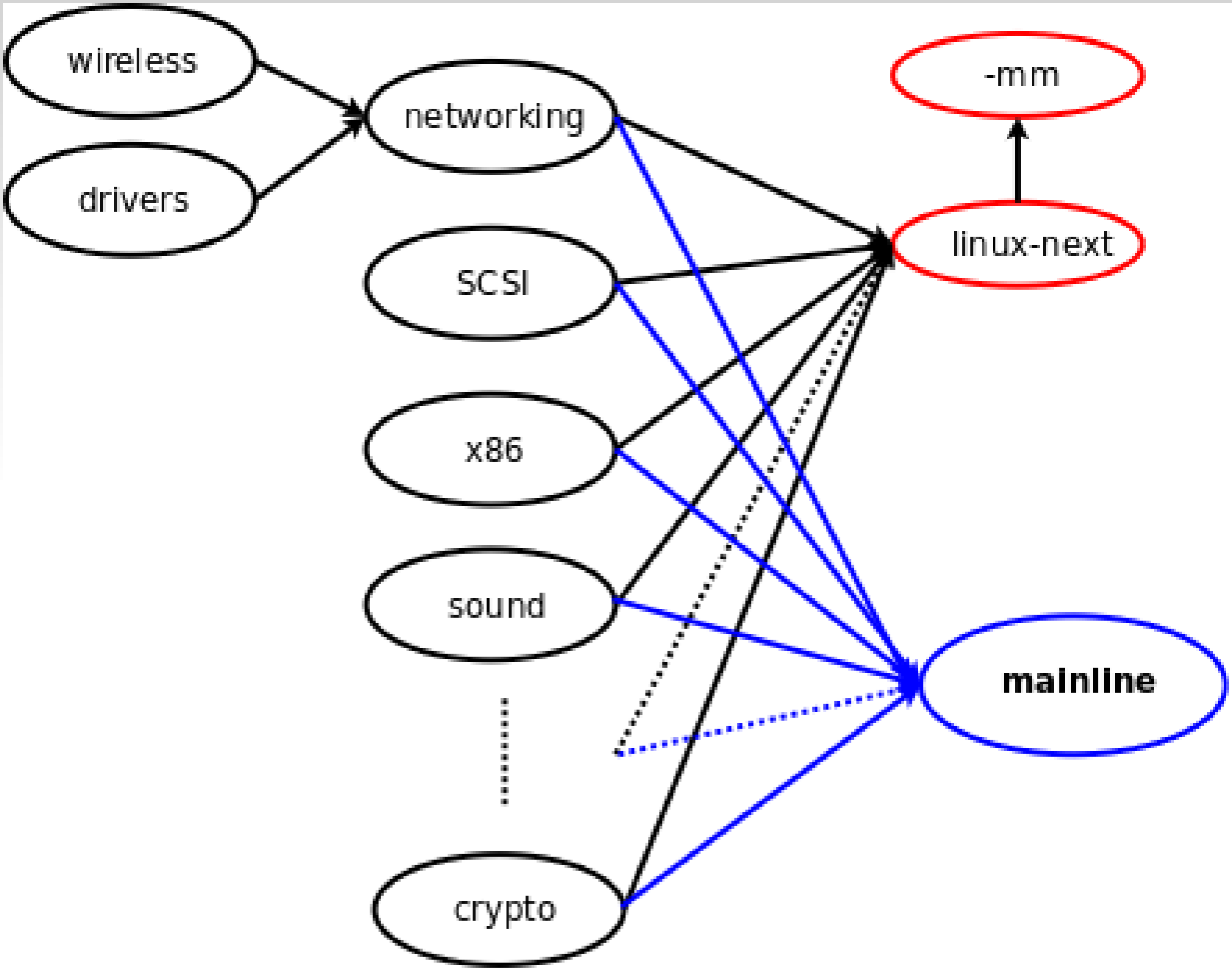
- Feed into the mainline
- ...or another subsystem tree

Subsystem maintainers

...are the true gatekeepers

But their power is not absolute


wireless
drivers
networking
SCSI
x86
sound
crypto
-mm
linux-next


wireless
drivers
networking
SCSI
x86
sound
crypto
-mm
linux-next
mainline

The moral of the story

Developers should target subsystem trees

Tips

Getting started: companies

Develop skills in-house

Getting started: companies

Get legal on board

Getting started: companies

Ensure management understands the process

Getting started: companies

Let your developers contribute

Getting started: developers

The #1 project for all kernel beginners should surely be "make sure that the kernel runs perfectly at all times on all machines which you can lay your hands on".

-- Andrew Morton

Getting started: developers

Review code!

Communication (1)

Communicate your plans early

Communication (2)

Specify requirements carefully

- “Why” instead of “what”

Communication (3)

Listen

Aim for the mainline

...early!

> “Do NOT fall into the trap of adding more and more stuff to an out-of-tree project. It just makes it harder and harder to get it merged. There are many examples of this”
>
> -- Andrew Morton

Posting code

Read the process document

Justify the change

- What problem is fixed?
- What feature is added (and why)?
- Effects on performance?

Where to post?

Find the subsystem mailing list

Find the maintainer

Consider linux-kernel

Expect to make changes

No code submission is perfect

Dealing with reviewers

Do not ignore reviews

Be nice to reviewers

Understand their motivation

- Do not take it personally

Signed-off-by:

Code submissions require a signoff

- ...a statement that it can be contributed under the GPL

About linux-kernel

Volume is high
Discussion is ... uninhibited
It's where the community gathers

Avoiding l-k may be appealing
...but it has its hazards

Follow through

"Dump and run" submissions are not appreciated

Be part of the process

The kernel needs you

Questions?

Development process document

http://ldn.linuxfoundation.org/book/how-participate-linux-community