The kernel report

(OSS EU 2023 edition)

Jonathan Corbet
LWN.net
corbet@lwn.net

Part 1: Statistics

Recent release history

Release	Date	Commits	Devs	1st time
6.0	Oct 2	15,402	2,034	236
6.1	Dec 11	13,942	2,043	303
6.2	Feb 19	15,536	2,088	294
6.3	Apr 24	14,424	1,971	250
6.4	Jun 25	14,835	1,980	282
6.5	Aug 27	13,561	1,921	271

Recent release history

Release	Date	Commits	Devs	1st time
6.0	Oct 2	15,402	2,034	236
6.1	Dec 11	13,942	2,043	303
6.2	Feb 19	15,536	**2,088**	294
6.3	Apr 24	14,424	1,971	250
6.4	Jun 25	14,835	1,980	282
6.5	Aug 27	13,561	1,921	271

Recent release history

Release	Date	Commits	Devs	1st time
6.0	Oct 2	15,402	2,034	236
6.1	Dec 11	13,942	2,043	303
6.2	Feb 19	15,536	2,088	294
6.3	Apr 24	14,424	1,971	250
6.4	Jun 25	14,835	1,980	282
6.5	Aug 27	13,561	1,921	271
6.6*	Oct 29	**<so-far numbers here>**		

Stable updates

Release	Updates	Commits
4.14	324	26,799
4.19	293	26,870
5.4	255	24,537
5.10	193	22,868
5.15	129	18,332
6.1	50	9,021

Part 2: Key questions facing the kernel community

What kernel should users run?

Ways to pick a kernel

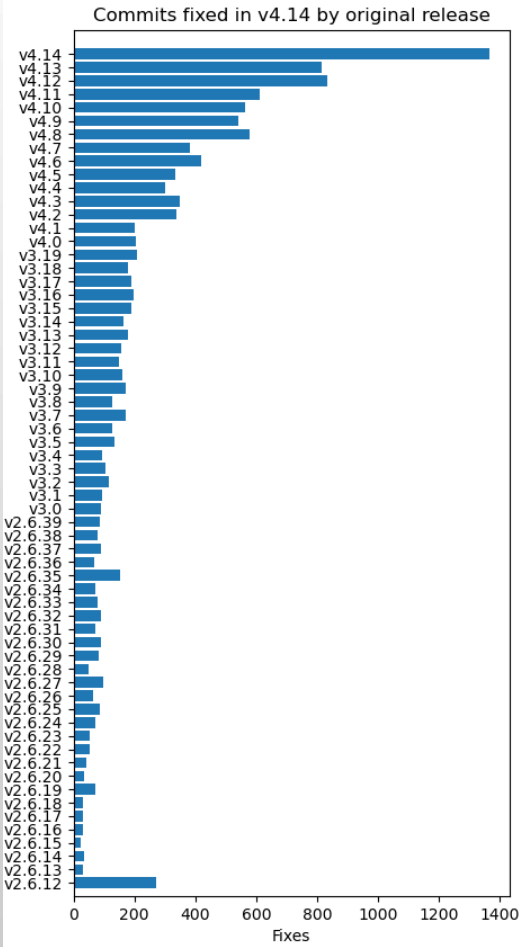
1) Run the latest stable update

You HAVE to take all of the stable/LTS releases in order to have a secure and stable system. If you attempt to cherry-pick random patches you will NOT fix all of the known, and unknown, problems, but rather you will end up with a potentially more insecure system, and one that contains known bugs.
— Greg Kroah-Hartman

Stable updates

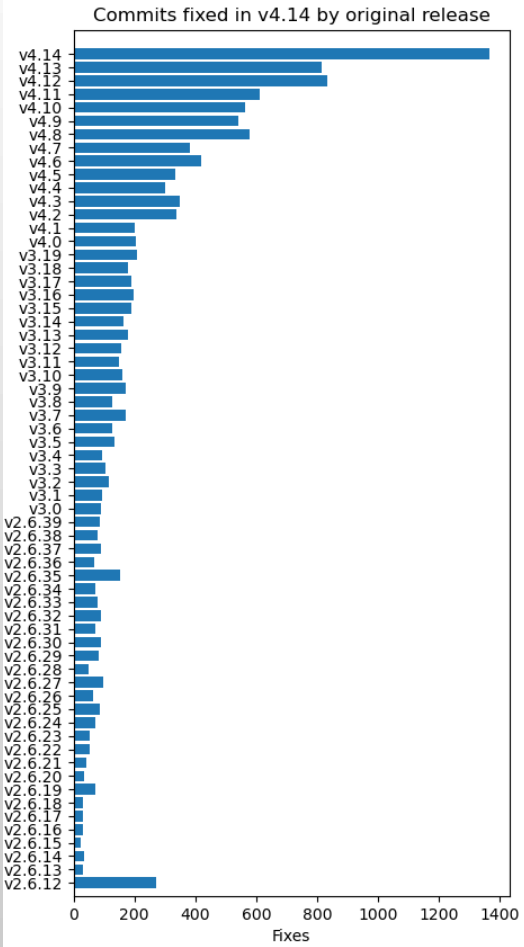
Release	Updates	Commits
4.14	**324**	**26,799**
4.19	293	26,870
5.4	255	24,537
5.10	193	22,868
5.15	129	18,332
6.1	50	9,021

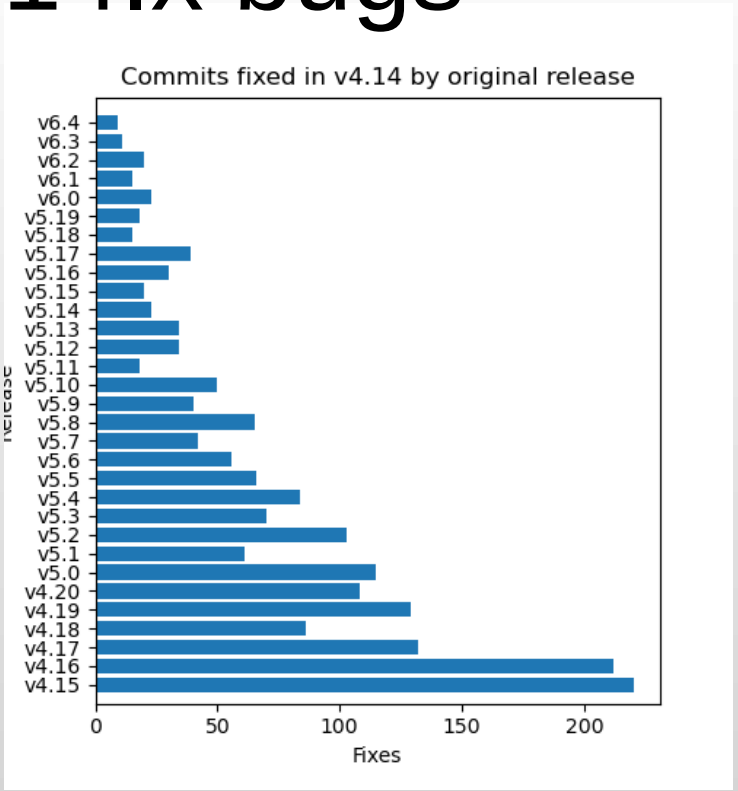
Source of 4.14.x bugs



Android

Source of 4.14.x bugs





Ways to pick a kernel

1) Run the latest stable update

2) Run an old kernel + backported fixes

What is best for our users?

One last note

Release	Updates	Commits
4.14	324	26,799
4.19	293	26,870
5.4	255	24,537
5.10	193	22,868
5.15	129	18,332
6.1	50	9,021

One last note

Release	Updates	Commits
~~4.14~~	~~324~~	~~26,799~~
~~4.19~~	~~293~~	~~26,870~~
~~5.4~~	~~255~~	~~24,537~~
5.10	193	22,868
5.15	129	18,332
6.1	50	9,021

BPF: how far do we go?

BPF review

BPF is an in-kernel virtual machine

- Code loaded from user space

“A safer form of C”

What BPF can do

Packet filtering
TCP congestion control
Traffic control
Routing++ w/XDP
Infrared drivers
Input drivers
System-call filtering (seccomp)
Linux security modules
Tracing and analysis
...

```
# bpftool prog list
2: tracing  name hid_tail_call  tag 7cc47bbf07148bfe  gpl
47: lsm  name restrict_filesystems  tag aae89fa01fe7ee91  gpl
51: cgroup_device  name sd_devices  tag 40ddf486530245f5  gpl
52: cgroup_skb  name sd_fw_egress  tag 6deef7357e7b4530  gpl
53: cgroup_skb  name sd_fw_ingress  tag 6deef7357e7b4530  gpl
54: cgroup_skb  name sd_fw_egress  tag 6deef7357e7b4530  gpl
55: cgroup_skb  name sd_fw_ingress  tag 6deef7357e7b4530  gpl
56: cgroup_device  name sd_devices  tag be31ae23198a0378  gpl
57: cgroup_skb  name sd_fw_egress  tag 6deef7357e7b4530  gpl
58: cgroup_skb  name sd_fw_ingress  tag 6deef7357e7b4530  gpl
59: cgroup_device  name sd_devices  tag be31ae23198a0378  gpl
60: cgroup_device  name sd_devices  tag ee0e253c78993a24  gpl
61: cgroup_device  name sd_devices  tag ee0e253c78993a24  gpl
62: cgroup_device  name sd_devices  tag 3a0ef5414c2f6fca  gpl
<six more...>
```

```
# bpftool prog list
2: tracing  name hid_tail_call  tag 7cc47bbf07148bfe  gpl
47: lsm  name restrict_filesystems  tag aae89fa01fe7ee91  gpl
51: cgroup_device  name sd_devices  tag 40ddf486530245f5  gpl
52: cgroup_skb  name sd_fw_egress  tag 6deef7357e7b4530  gpl
53: cgroup_skb  name sd_fw_ingress  tag 6deef7357e7b4530  gpl
54: cgroup_skb  name sd_fw_egress  tag 6deef7357e7b4530  gpl
55: cgroup_skb  name sd_fw_ingress  tag 6deef7357e7b4530  gpl
56: cgroup_device  name sd_devices  tag be31ae23198a0378  gpl
57: cgroup_skb  name sd_fw_egress  tag 6deef7357e7b4530  gpl
58: cgroup_skb  name sd_fw_ingress  tag 6deef7357e7b4530  gpl
59: cgroup_device  name sd_devices  tag be31ae23198a0378  gpl
60: cgroup_device  name sd_devices  tag ee0e253c78993a24  gpl
61: cgroup_device  name sd_devices  tag ee0e253c78993a24  gpl
62: cgroup_device  name sd_devices  tag 3a0ef5414c2f6fca  gpl
<six more...>
```

What BPF *might* do

The extensible scheduler class

- Write complete CPU schedulers in BPF
- https://lwn.net/Articles/922405/

Why schedulers in BPF?

Easy experimentation
Faster scheduler development
Ad hoc schedulers for special workloads
...

What BPF *might* do

Page aging

Why?

- Adjust memory-management to workload

What BPF *might* do

io_uring integration

Why?

- Better control over sequences of operations
- Create a complete programming environment

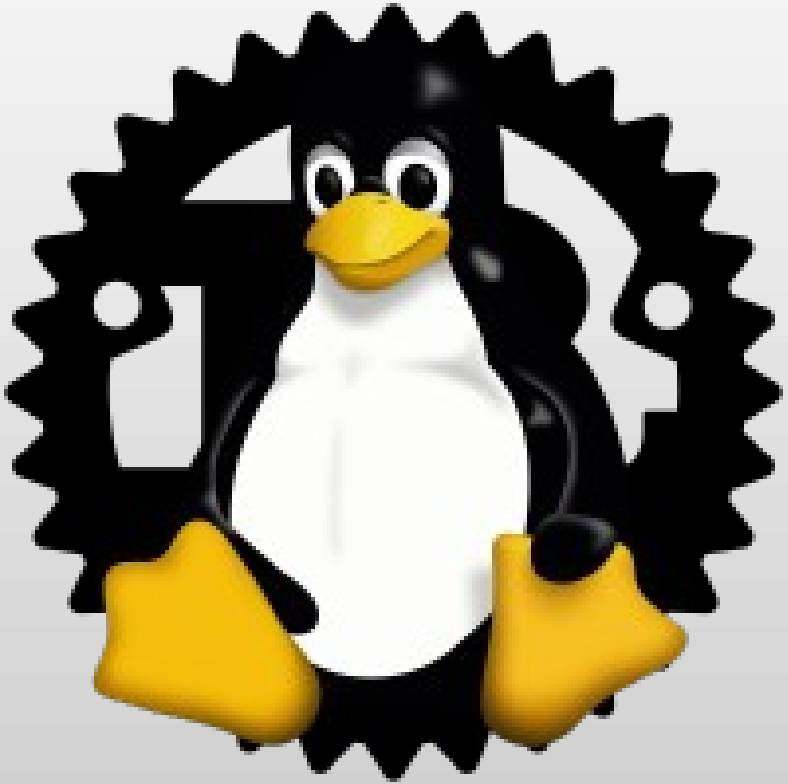
Extensible scheduler class: **rejected**

Why *not* BPF schedulers?

Added maintenance burden
Benchmark gaming
Vendors may require specific schedulers
ABI concerns
Redirection of work on core scheduler

Where do we draw the line?

Rust



Rust has a lot to offer

A stronger type system
No undefined behavior
- No use-after-free problems
- No data races
- Everything initialized
- ...

Attractive to newer developers

Why *not* Rust in the kernel?

A new language adds complexity
The language is still evolving — quickly
Maintainers will need to learn Rust
Lots of glue code
Some things are hard to do in Rust

Why *not* Rust in the kernel?

A new language adds complexity
The language is still evolving — quickly
Maintainers will need to learn Rust
Lots of glue code
Some things are hard to do in Rust
Conservatism

```
pub unsafe fn current() -> impl Deref<Target = Task> {
    struct TaskRef<'a> {
        task: &'a Task,
        _not_send: PhantomData<*mut ()>,
    }

    impl Deref for TaskRef<'_> {
        type Target = Task;

        fn deref(&self) -> &Self::Target {
            self.task
        }
    }
```

“There are possibly some well-designed and written parts which have not suffered a memory safety issue in many years. It's insulting to present this as an improvement over what was achieved by those doing all this hard work.”
— a longtime kernel developer

Status

Initial Rust support merged for 6.1

- A “hello world” module

More support code in subsequent kernels

- Access to existing types and functions
- ...but safer

Status

Lots more support code out of tree

Interesting new stuff:

- Apple M1 GPU driver
- PuzzleFS implementation
- Plan9 filesystem server (read/write)

Rust support was merged as
an **experiment**

When do we decide that the experiment is a success?

When do we decide that the experiment is a success?

→ when we merge the first feature that users depend on

The Rust decision point is coming soon

Threat models

Kernel security

...has gotten better!

- Improved APIs
- Adoption of hardening techniques
- Better patch management

...but it’s still awful

Security — against what?

Security — against what?

- Remote attackers?

Security — against what?

- Remote attackers?
- Local, unprivileged accounts?

Security — against what?

- Remote attackers?
- Local, unprivileged accounts?
- The root account?

Protecting against root

Run the kernel in lockdown mode
    Numerous features disabled
fs-verity / composefs / dm-verity …
Integrity measurement

Protecting against root

Run the kernel in lockdown mode
- Numerous features disabled

fs-verity / composefs / dm-verity …

Integrity measurement

But what about:
- Malicious filesystem images?
- Writing to mounted block devices?

Security — against what?

- Remote attackers?
- Local, unprivileged accounts?
- The root account?
- The computer itself?

Confidential computing

Even the host cannot be trusted

Thus:

- Require attestation from the CPU
- Disable every feature you can
- Harden device drivers against hostile input

> Talking about security models without having an agreed upon threat model is really a waste of time.
> — Ted Ts’o

The kernel does not have
an agreed-upon threat model

The maintainership crisis

Being maintainer feels like a punishment, and that cannot stand. We need help.
— Darrick Wong

Maintainers/longtime developers are burning out.
— Josef Bacik

What is going on?

Maintainer pain points

Increasing demands

Maintainer pain points

Increasing demands
Understaffing

Most of my friends work for small companies, nonprofits, and local governments. They report the same problems with overwork, pervasive fear and anger, and struggle to understand and adapt to new ideas that I observe here. They see the direct connection between their org's lack of revenue and the under resourcedness.

They /don't/ understand why the hell the same happens to me and my workplace proximity associates, when we all work for companies that each clear hundreds of billions of dollars in revenue per year.
— Darrick Wong

Maintainer pain points

Increasing demands
Understaffing
Lack of employer support

> But being a maintainer myself with a full-time job that is not to do my maintainership, I'm struggling to find time to work on this.
> — Steve Rostedt

Many maintainers are not paid to maintain

How does your company compare?

https://www.kernel.org/doc/html/latest/process/contribution-maturity-model.html

Maintainer pain points

Increasing demands
Understaffing
Lack of employer support
Fuzzers

Dark areas in the kernel

Documentation
Build system
Many core-kernel areas
Drivers for older hardware
...

Dark areas in the kernel

Documentation
Build system
Many core-kernel areas
Drivers for older hardware
…
Maintainers

> Open source is free like a puppy is free
> — Scott McNealy

How can we take better care of the puppy?

Questions?

(slides: https://lwn.net/talks/2023/kr-osseu.pdf)