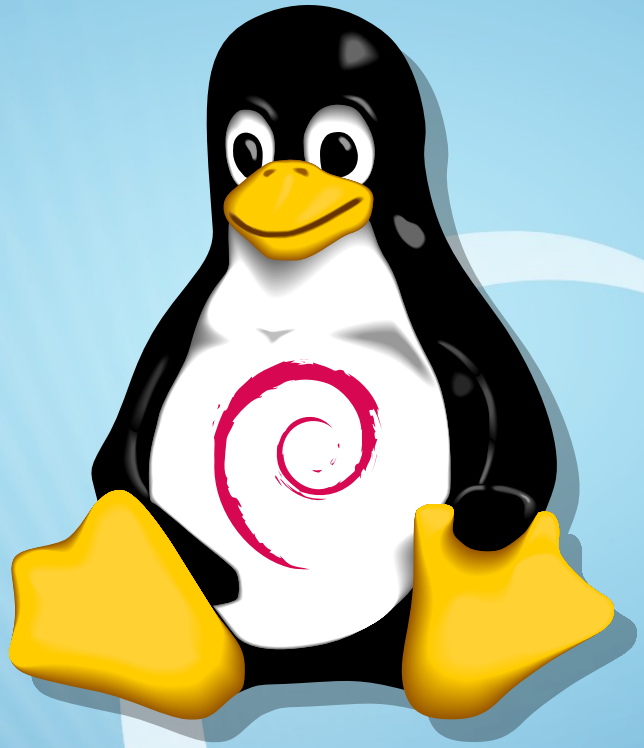
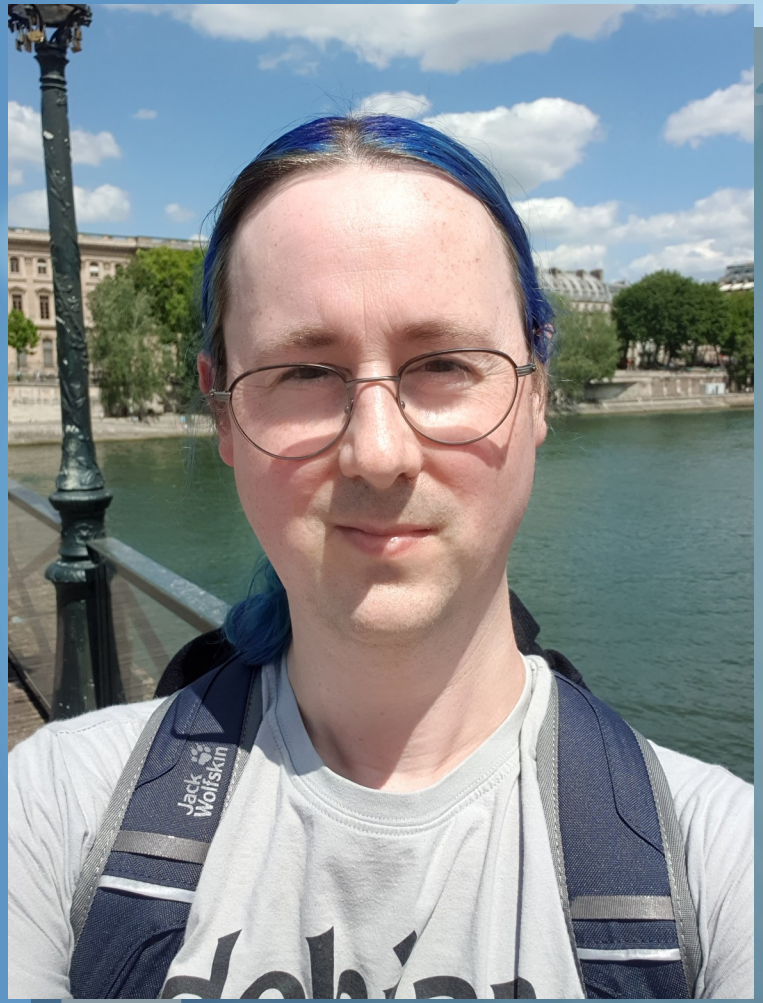
What’s new in the Linux kernel

and what’s missing in Debian



Ben Hutchings · DebConf 2025, Brest, France



Ben Hutchings

- Working on Linux kernel and related code for Debian and in paid jobs for about 15 years
- Debian kernel and LTS team member, doing various kernel packaging and backporting work

mainline:	6.16-rc6	2025-07-13
stable:	6.15.6	2025-07-10
longterm:	6.12.37	2025-07-10
longterm:	6.6.97	2025-07-10
longterm:	6.1.144	2025-07-10
longterm:	5.15.187	2025-07-10
longterm:	5.10.239	2025-06-27
longterm:	5.4.295	2025-06-27
linux-next:	next-20250714	2025-07-14

Linux keeps changing

- Linus makes a release with new features every 9-10 weeks
- Larger features may take multiple releases to become useful

- Some features need changes elsewhere to enable them:
  - New user-space management tool
  - New version of existing user-space tool
  - Applications and libraries using new API
  - Packaging or infrastructure changes
- I'll talk about new features in Linux 6.13 to 6.16 inclusive

Recap of previous years' features

- **io_uring:** Support for pipe create and zero-copy receive ops; also ring resizing, hybrid polling, ...
- **ID-mapped mounts:** Nested ID-mapping allowed
- **Rust:**
  - Many more bindings: filesystem, memory management, time and timers, PCI and platform devices, clock control, cpufreq, DRM, ...
  - Supported on 32-bit Arm (v7 only)
  - Still no large drivers upstream, but skeleton of **nova** driver for newer Nvidia GPUs is there
- **bcachefs:** Probably on the way out due to continuing conflicts between developers

Preemption options (1)

- Thread running in user-space can be preempted at any time by higher priority threads, or when their “time slice” ends
- For a thread in the kernel this is not so simple:
  - It may disable preemption while accessing per-CPU state
  - It may disable interrupts while accessing state shared with interrupt handler
  - Keeping track of preemptibility has some performance cost
- Longstanding compile-time options for when to allow preemption:
  - **PREEMPT_NONE:** At specific points if time slice expired (max throughput)
  - **PREEMPT_VOLUNTARY:** At specific points (compromise)
  - **PREEMPT:** Any time not holding spinlock or explicitly disabled (min latency)
  - **PREEMPT_RT:** Almost any time (min worst-case latency)

Preemption options (2)

- Debian chose to set PREEMPT_VOLUNTARY, with alternate builds setting PREEMPT_RT on some architectures
- But wait, there's more:
  - **PREEMPT_DYNAMIC:** Boot-time choice between all except PREEMPT_RT
    - Available for arm64, loongarch, riscv, x86, **s390** [6.13], **powerpc** [6.16]
    - Enabled for amd64 but now possible for all 64-bit release architectures
  - **PREEMPT_LAZY:** Similar to PREEMPT, but defers preemption by non-RT tasks up to 1 timer tick; meant to replace PREEMPT_{NONE,VOLUNTARY}
    - Impact on throughput is lower because victim task usually exits kernel within 1 tick and is preempted then anyway
    - Still provides fairly low latency for RT tasks
    - Available for all the same architectures and enabled in Debian

Large atomic writes (1)

- Relational database systems (RDBMSs) implement ACID transactions
  - **Atomic:** a transaction must be seen as having happened completely or not at all, even if the system crashes
- Writing a single logical block of storage (usually 4 kiB) is atomic, but most RDBMSs work with larger blocks (16 kiB for MySQL or Postgres)
  - RDBMS recovering from a crash must be able to handle the last write being “torn”, with the block now containing part old and part new data
- NVMe and some SCSI devices support atomically writing larger blocks, but they must be properly aligned
  - e.g. with 4 kiB logical block size, a 16 kiB atomic write must start at LBA divisible by 4

Large atomic writes (2)

- Some filesystems and stacked block devices now support this if underlying devices do:
  - ext4
  - XFS (requires setting `FORCEALIGN` flag on inode)
  - md-RAID: modes 0/1/10 only
  - device-mapper: dm-linear (LVM2), dm-stripe, dm-raid1 only
- UAPI extensions introduced for applications to take advantage of this:
  - `RWF_ATOMIC` flag for `pwritev2()` and io_uring write ops
  - `STATX_WRITE_ATOMIC` flag for `statx()` to query atomic write limits

➢ Large atomic writes allow for more efficient write sequence; LWN quoted Ted Ts'o claiming "60-100% improvement in [MySQL] performance"

- Resilient queued spinlocks (rqspinlocks) support deadlock detection and recovery
  - Nested acquisition of spinlocks not allowed, but with rqspinlocks this is OK
- Atomic load-acquire and store-release instructions
  - Atomic read-modify-write instructions already existed but are slow
  - Some lockless algorithms will benefit from using these instead
  - Supported by core interpreter and JITs for arm64, riscv, x86_64
- Red-black tree (rbtree) traversal without removing
- Improved list traversal: peek at first/last without removing

Script execution control

- Some systems have a security policy restricting which executables can run
  - Implemented through an LSM such as SELinux
  - Depends on `execve()` opening the file for execution
- This does not restrict scripts:
  - `execve("script", …)` does open `script` for execution
  - but this can be evaded with: `execve("/usr/bin/interp", &{"interp", "script", NULL}, …)`
- ➢ User-space script interpreters need to participate in policy enforcement
  - New API for this: `execveat(…, AT_EXECVE_CHECK)`
  - All script interpreters should call this for the script file descriptor (usually) or name (when searching a path)
  - Dynamic linker should also call this for each shared library

io_uring for FUSE (1)

FUSE allows Filesystem drivers to run in USErspace instead of the kernel, offering several benefits to implementers:

Filesystem type	Kernel	FUSE
Implementation language	C, Rust?	any
License	GPLv2 or compatible	any
Portability	Linux kAPI not available elsewhere	same/similar API supported on many other OSes
Impact of bugs	system crash, system-wide data loss, kernel compromise	filesystem loss, account compromise if not sandboxed
Performance	generally higher	generally lower

io_uring for FUSE (2)

- Performance is the weak point:
  - Communication through single fd per mount — can be a bottleneck
  - Operations generally require at least twice as many context switches
- ublk (user-space block drivers) API avoided some of this by using io_uring in reverse:
  1. Driver submits request for block I/O (D1)
  2. Kernel completes D1 with a block I/O request (K1)
  3. Driver submits request for block I/O (D2), containing response to K1
  - All of this can run asynchronously and using separate rings per CPU
- Linux now supports the same model for FUSE drivers
  - Setup, notifications and interrupts still done through `/dev/fuse`

Secure Boot Advanced Targetting (SBAT)

- UEFI Secure Boot includes block-list (dbx) of boot components that should no longer be trusted despite being properly signed
  - dbx is EFI variable containing hashes of certificates and binaries
  - …but variable storage is limited, while set of insecure binaries keeps growing
- SBAT introduced a “security generation” per component:
  - Security generation is incremented after each SB-relevant fix
  - Minimum generation per component stored in EFI variables
  - Boot loader checks both signatures and security generations
- Already implemented in shim and GRUB (and Windows)
- Linux starting to support embedding SBAT security generation:
  - [6.16] arm64, loongarch, riscv
  - [6.17] x86

Packaging changes (1)

- Architecture and flavour updates:
  - rt featureset replaced with rt flavours
    - Not yet available for armhf, and no i915 driver
  - [sh4] Removed broken sh7785lcr flavour
- Binary package updates:
  - **linux-misc-tools** added (in Git); will allow building firmware-free from source
  - **linux-support-*ABINAME*** removed; was only used by src:firmware-{free,nonfree} which now copy required Python package from src:linux
- Kernel compressed with zstd on most architectures instead of xz
- BTF generation works for external modules

Packaging changes (2)

- ABI name suffix configurable per suite, and will be “+deb*REL*” in Debian stable and unstable
- Compiler dependencies simplified to gcc-*VER*-for-host
- Support for kernel image hooks installed under `/usr/share/kernel` (for use in forky)

- `debian/bin/test-patches` now works without `$EMAIL` or `$DEBEMAIL` set
- Bug script includes `modprobe` configuration, after several bug reports caused by a third party adding a blacklist directive
- Git branches renamed to `debian/`*UPSTREAM-VER*`/`*CODENAME* and `debian/latest`, *approximately* following DEP-14

Packaging changes (3)

Configuration changes to add or improve support for:

[amd64] **AMD** “Renoir”, SEV; **Intel** “Whiskey Cove”, TDX

[amd64,arm64] **AWS** Nitro Enclaves

[arm64] Arm Confidential Compute Architecture; **Google** Pixel 6; **Lenovo** Thinkpad X13s; various **MediaTek** SoCs; **Qualcomm** SC7180

[armhf] **STMicro** STM32MP15x

[loong64] **Loongson** LS7A1000/2000

[riscv64] **Sophgo** SG204x; **SpacemiT** SoCs; **T-Head** C900, TH1520

Questions?

Credits & License

- Content by Ben Hutchings
  www.decadent.org.uk/ben/talks/
  License: GPL-2+
- Original OpenOffice.org template by Raphaël Hertzog
  raphaelhertzog.com/go/ooo-template
  License: GPL-2+
- Background based on “Serenity” theme by Edward Padilla
  wiki.debian.org/DebianArt/Themes/serenity
  License: GPL-2