Popping Kernels for Linux Distributions

Making Linux kernel builds and the life of a distro Linux kernel maintainer

Neal Gompa

Velocity Limitless

Who am I?

- Open Source Advocate
- Owner and Principal at Velocity Limitless
- Contributor to Fedora, CentOS, openSUSE, and Mageia
    - Member of Fedora Engineering Steering Committee (FESCo)
    - Member of Fedora Workstation, Fedora Cloud, and Fedora Server WGs
    - Chair of Fedora KDE SIG
    - Co-chair of CentOS Hyperscale SIG
    - Member of CentOS Alternative Images SIG
    - Member of the openSUSE Board
    - Member of the openSUSE Heroes (Infrastructure)
    - Member of the Mageia Council and Mageia.org Board
- Contributor to RPM, DNF, KIWI, Koji, Open Build Service, and various other similar projects
- KDE contributor
    - Member of KDE, e.V. and X.Org Foundation
- Co-host of the Sudo Show podcast

Velocity Limitless

The Linux kernel

Velocity
Limitless

The Linux kernel is (not?) special

The Linux kernel is simple:

- Written primarily in C and assembly
- Has no external runtime dependencies
- All subsystems and drivers are included in the kernel
- Is substantially well-documented

The Linux kernel is complex:

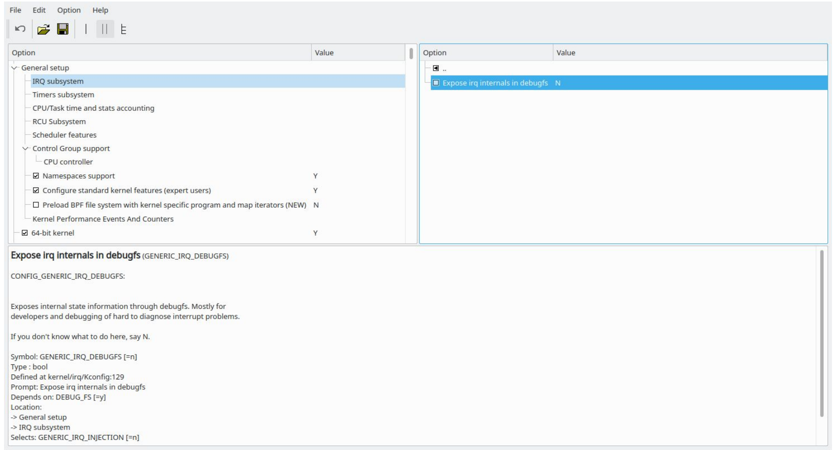
- Newly adding Rust to the mix of languages
- Builds user-space components that are tightly coupled to the kernel built
- All subsystems and drivers are included in the kernel
- The build tooling is sufficiently complicated to necessitate its own bespoke programs

Building the Linux kernel for yourself…

Typically, when building the kernel for yourself, there are a few reasonable choices:

- Seeding the build from your running kernel (e.g. `make localmodconfig`)
- Seeding the build from an existing kernel (e.g. `make olddefconfig`)
- Building the kernel with all the things (e.g. `make allmodconfig`)

And there's the slightly less reasonable choice of doing it interactively and selecting everything individually (e.g. `make xconfig`)



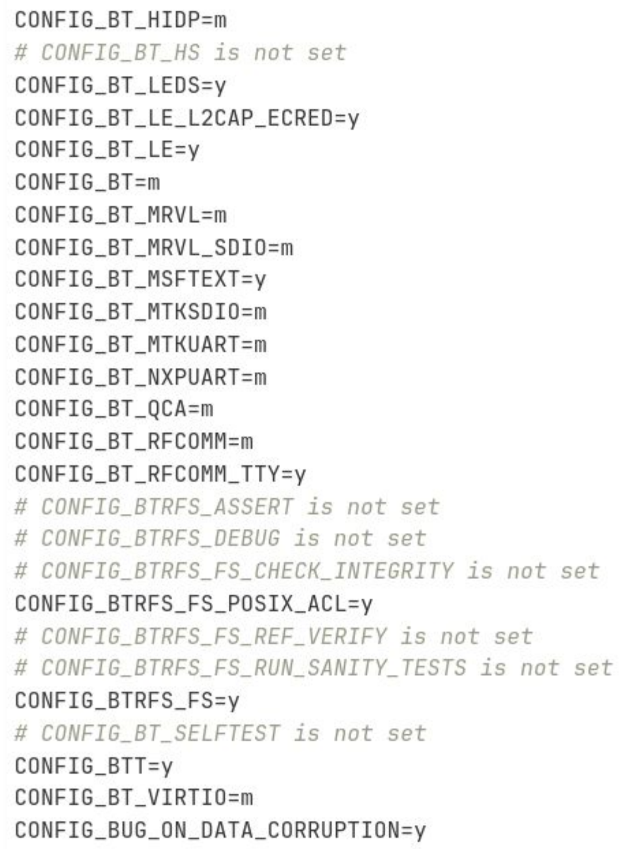
Option	Value
General setup	
IRQ subsystem	
Timers subsystem	
CPU/Task time and stats accounting	
RCU Subsystem	
Scheduler features	
Control Group support	
CPU controller	
☑ Namespaces support	Y
☑ Configure standard kernel features (expert users)	Y
☐ Preload BPF file system with kernel specific program and map iterators (NEW)	N
Kernel Performance Events And Counters	
☑ 64-bit kernel	Y

Option	Value
..	
☐ Expose irq internals in debugfs	N

```
Expose irq internals in debugfs (GENERIC_IRQ_DEBUGFS)

CONFIG_GENERIC_IRQ_DEBUGFS:


Exposes internal state information through debugfs. Mostly for
developers and debugging of hard to diagnose interrupt problems.

If you don't know what to do here, say N.

Symbol: GENERIC_IRQ_DEBUGFS [=n]
Type : bool
Defined at kernel/irq/Kconfig:129
Prompt: Expose irq internals in debugfs
Depends on: DEBUG_FS [=y]
Location:
-> General setup
-> IRQ subsystem
Selects: GENERIC_IRQ_INJECTION [=n]
```

Screenshot of `make xconfig` (credit: Wikipedia)

Building the Linux kernel for yourself… is a lot of work!



```
CONFIG_BT_HIDP=m
# CONFIG_BT_HS is not set
CONFIG_BT_LEDS=y
CONFIG_BT_LE_L2CAP_ECRED=y
CONFIG_BT_LE=y
CONFIG_BT=m
CONFIG_BT_MRVL=m
CONFIG_BT_MRVL_SDIO=m
CONFIG_BT_MSFTEXT=y
CONFIG_BT_MTKSDIO=m
CONFIG_BT_MTKUART=m
CONFIG_BT_NXPUART=m
CONFIG_BT_QCA=m
CONFIG_BT_RFCOMM=m
CONFIG_BT_RFCOMM_TTY=y
# CONFIG_BTRFS_ASSERT is not set
# CONFIG_BTRFS_DEBUG is not set
# CONFIG_BTRFS_FS_CHECK_INTEGRITY is not set
CONFIG_BTRFS_FS_POSIX_ACL=y
# CONFIG_BTRFS_FS_REF_VERIFY is not set
# CONFIG_BTRFS_FS_RUN_SANITY_TESTS is not set
CONFIG_BTRFS_FS=y
# CONFIG_BT_SELFTEST is not set
CONFIG_BTT=y
CONFIG_BT_VIRTIO=m
CONFIG_BUG_ON_DATA_CORRUPTION=y
```

Velocity Limitless

Building the Linux kernel for yourself… is a lot of work!

So far, this is only about the simple case of building a kernel once. Maintaining that kernel over time for *just yourself* means repeating this process ***every time*** a new release is made (provided you wish to update).

And because of how the kernel is developed, build options do not offer stability and need to be evaluated again each time a new kernel is being built. It can get more complex when managing patches without some kind of mechanism to track patches applied to separate upstream and personal changes.

And while the kernel can wrap the build into packages to install on your system (using e.g. `make binrpm-pkg`), that does not mean the packaging is necessarily correct for your distribution and integrates with everything else you have.

Velocity Limitless

The Linux kernel for distributions

Velocity
Limitless

Building the Linux kernel for distributions…

Linux distributions go through some of the same processes for building and maintaining kernel packages, but there are added duties that make things more complex:

- Tracking build options based on features, architectures, and distribution needs
- Executing kernel builds in an isolated and reproducible way
- Integrating the built kernels into the distribution machinery
- Backporting fixes and features to align the kernel to userspace needs and lifecycle requirements

These result in distributions implementing tooling on top of the Linux kernel build to enhance it for these needs.

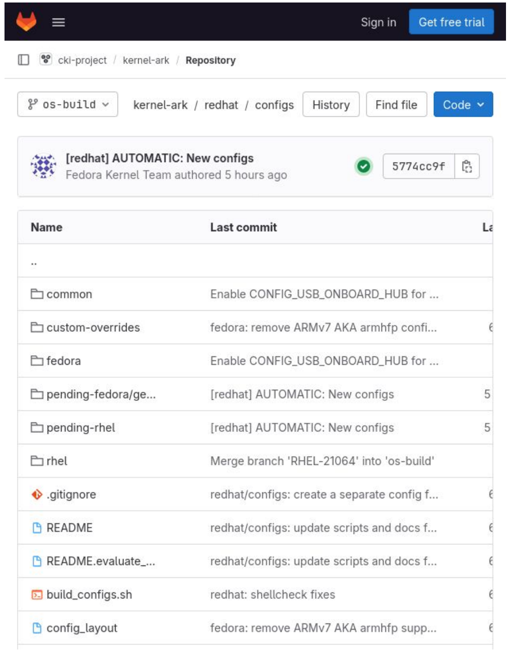
Velocity Limitless

Building the Linux kernel for Red Hat distributions

Velocity Limitless

Linux distributions in the Red Hat family have a central source repository for their Linux kernel sources that contains the merged trees of the Linux kernel and the packaging code for Red Hat distributions. This repository is known as the Always Ready Kernel (or ARK for short). This refers to the fact the tree contains an "always-ready" to release tree to create the RHEL kernel, which is generally a long-term maintained derivative of the mainline Linux kernel code.

The `redhat` folder contains all the distribution kernel build scripts and related files. A notable feature of this setup is that it contains its own processing logic for generating the kernel build configuration that includes a layering model to allow common, per-architecture, and per-distribution settings to be made and maintained easily.


Sign in
Get free trial
cki-project / kernel-ark / Repository
os-build
kernel-ark / redhat / configs
History
Find file
Code

[redhat] AUTOMATIC: New configs
Fedora Kernel Team authored 5 hours ago
5774cc9f

Name	Last commit	La
..		
common	Enable CONFIG_USB_ONBOARD_HUB for ...	
custom-overrides	fedora: remove ARMv7 AKA armhfp confi...	6
fedora	Enable CONFIG_USB_ONBOARD_HUB for ...	
pending-fedora/ge...	[redhat] AUTOMATIC: New configs	5
pending-rhel	[redhat] AUTOMATIC: New configs	5
rhel	Merge branch 'RHEL-21064' into 'os-build'	
.gitignore	redhat/configs: create a separate config f...	6
README	redhat/configs: update scripts and docs f...	6
README.evaluate_...	redhat/configs: update scripts and docs f...	6
build_configs.sh	redhat: shellcheck fixes	6
config_layout	fedora: remove ARMv7 AKA armhfp supp...	6

Velocity Limitless

Building the Linux kernel for Fedora Asahi Remix

The Fedora Asahi Remix is a derivative of Fedora Linux that is optimized for Apple Silicon systems, such as the Macs released from 2020 onward.

The kernel sources for this comprises of three major changes:

- A large patch set from AsahiLinux/linux targeting the major Linux kernel version (e.g. 6.6)
- Configuration changes to turn on Asahi Linux specific code
- Enabling the 16k flavor for AArch64 kernel builds

The ARK model helps us by allowing us to reliably maintain this in a way that minimizes any split-brain effect.

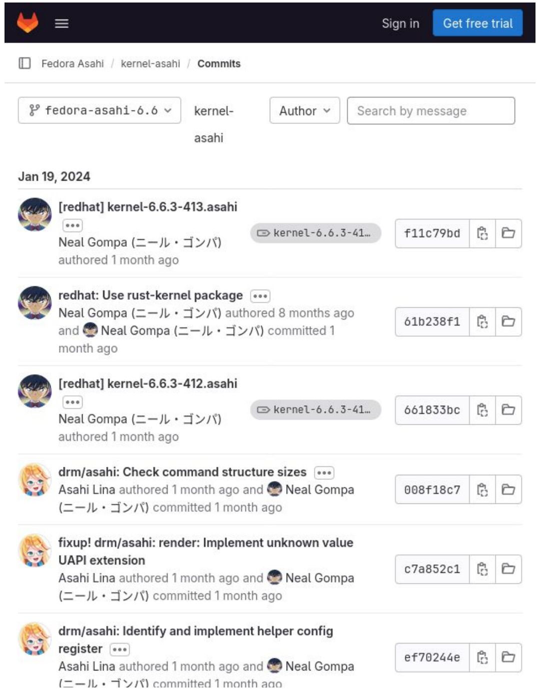
Velocity Limitless

Building the Linux kernel for Fedora Asahi Remix

The challenges in developing and maintaining this kernel do not start at the beginning. The problems come when moving to new kernel versions. Since Fedora Asahi Remix tracks Fedora Linux, it also inherits the policy of aggressively updating to new kernel releases.

Because the kernel maintains no stability in its own code, each cycle where we have to maintain our patch set for the kernel takes a lot of time to prepare as patches may need to be completely rewritten in some circumstances, and this is semi-common!

The result is that Fedora Asahi Remix has to lag a lot from Fedora Linux because it takes time and effort to rewrite and retest everything. Sometimes it means we skip a whole kernel series. It also can mean we observe difficult-to-triage bugs because of the intersection of mainline code and custom code interacting poorly.



Jan 19, 2024

[redhat] kernel-6.6.3-413.asahi
Neal Gompa (ニール・ゴンパ) authored 1 month ago
kernel-6.6.3-41…
f11c79bd

redhat: Use rust-kernel package
Neal Gompa (ニール・ゴンパ) authored 8 months ago and Neal Gompa (ニール・ゴンパ) committed 1 month ago
61b238f1

[redhat] kernel-6.6.3-412.asahi
Neal Gompa (ニール・ゴンパ) authored 1 month ago
kernel-6.6.3-41…
661833bc

drm/asahi: Check command structure sizes
Asahi Lina authored 1 month ago and Neal Gompa (ニール・ゴンパ) committed 1 month ago
008f18c7

fixup! drm/asahi: render: Implement unknown value UAPI extension
Asahi Lina authored 1 month ago and Neal Gompa (ニール・ゴンパ) committed 1 month ago
c7a852c1

drm/asahi: Identify and implement helper config register
Asahi Lina authored 1 month ago and Neal Gompa (ニール・ゴンパ) committed 1 month ago
ef70244e

Velocity Limitless

Building the Linux kernel for CentOS Hyperscale

CentOS Hyperscale is built on CentOS Stream, the immediate upstream for Red Hat Enterprise Linux. This means that by default CentOS Stream provides the kernel that is used by Red Hat Enterprise Linux (RHEL).

The RHEL kernel is derived from a specific Linux kernel version and features and fixes are backported in such a way that behavioral and API stability is somewhat preserved.

Building a variant with different features is very difficult in this environment as it becomes necessary to do the same. Each time it needs to be done, it becomes more difficult than the previous time. In the end, it became untenable.

CentOS
Community Enterprise
Operating System

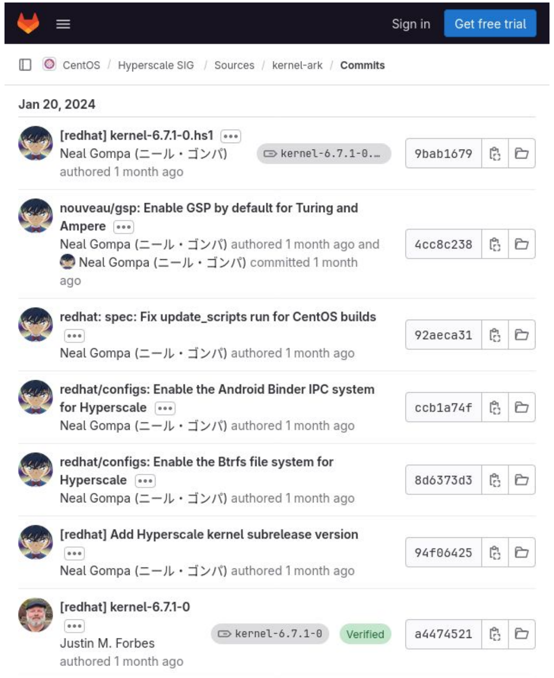
Velocity Limitless

Building the Linux kernel for CentOS Hyperscale

CentOS Hyperscale is built on CentOS Stream, the immediate upstream for Red Hat Enterprise Linux. This means that by default CentOS Stream provides the kernel that is used by Red Hat Enterprise Linux (RHEL).

But that does not mean CentOS Hyperscale has to use the RHEL kernel. Instead this is branched directly from the ARK repository to reduce the delta down to just configuration changes to turn on features.

With no significant patches and no non-merged patches, moving to new releases can happen very quickly.



Jan 20, 2024

[redhat] kernel-6.7.1-0.hs1
Neal Gompa (ニール・ゴンパ) authored 1 month ago — kernel-6.7.1-0.… — 9bab1679

nouveau/gsp: Enable GSP by default for Turing and Ampere
Neal Gompa (ニール・ゴンパ) authored 1 month ago and Neal Gompa (ニール・ゴンパ) committed 1 month ago — 4cc8c238

redhat: spec: Fix update_scripts run for CentOS builds
Neal Gompa (ニール・ゴンパ) authored 1 month ago — 92aeca31

redhat/configs: Enable the Android Binder IPC system for Hyperscale
Neal Gompa (ニール・ゴンパ) authored 1 month ago — ccb1a74f

redhat/configs: Enable the Btrfs file system for Hyperscale
Neal Gompa (ニール・ゴンパ) authored 1 month ago — 8d6373d3

[redhat] Add Hyperscale kernel subrelease version
Neal Gompa (ニール・ゴンパ) authored 1 month ago — 94f06425

[redhat] kernel-6.7.1-0
Justin M. Forbes authored 1 month ago — kernel-6.7.1-0 — Verified — a4474521

What about other Linux distributions?

We primarily discussed the Red Hat family of distributions and their processes, but other Linux distributions have their own approaches:

- SUSE distributions use a repository (confusingly called kernel-source) containing scripts and patch queues for generating kernel packaging based on a target mainline kernel release. This exports a "prepared" source tree and a submitted package to the kernel development project on the openSUSE Build Service.
- Debian uses a packaging repository that contains a Makefile that dynamically constructs the build flags and options based on the dpkg vendor, target architecture, and kernel image variants. To build the package, the sources must be downloaded and merged separately with the packaging repository.
- Ubuntu uses a variant of the Debian style, except with the sources merged for similar reasons to why ARK is done that way. Each Ubuntu version has its own repository named after its codename (e.g. Ubuntu 24.04 has the "noble" repository)
- Arch, Mageia, and OpenMandriva all have simpler packaging schemes that merely use a unified config file that is updated alongside kernel version updates. This is the closest to how an individual would build the kernel.

Velocity Limitless

Building the Linux kernel for distributions is about the long haul!

Building the Linux kernel for distributions is about the long haul

At the end of the day, Linux distribution package maintainers optimize for minimizing effort over the long-term, which leads to attempting to set up things to require as little thinking after the initial setup to support their needs.

This drives many Linux distributions have very complex packaging for their Linux kernel packages. After the initial setup of the packaging, every change needs to be easily tracked, testable, and revertable (if needed).

Despite the different ways distributions package the Linux kernel, these characteristics are often visible in their maintenance strategies.

Velocity Limitless

Velocity
Limitless

The End

www.velocitylimitless.com