

How to test a specific version of Linux on PC hardware?

AlpOSS 2025

Michael Opdenacker

Root Commit

Feb. 20, 2025



© 2024-2025 Root Commit. Licensed under CC BY-SA 4.0.

Embedded Linux consultant and trainer

- <https://rootcommit.com/about/michael-opdenacker/>
- Previously, founder at Bootlin
- New founder of Root Commit
- Offering embedded Linux training courses with a focus on practical activities, interactivity and learning techniques.
<https://rootcommit.com/training/>
- Free Software enthusiast and advocate
(member of April.org)



Introduction

- To get new features or bug fixes
- To test your system on future kernels that you will have sooner or later.
- To contribute to the Linux kernel (writing code, reporting or fixing bugs)
- Because tweaking the kernel is fun!

Real life example build: Raspberry Pi 5
Updating an SD card initially generated by Yocto

```
$ wget https://cdn.kernel.org/pub/linux/kernel/v6.x/linux-6.13.3.tar.xz
$ tar xf linux-6.13.3.tar.xz
$ cd linux-6.13.3
$ export ARCH=arm64
$ export LLVM=1
$ make defconfig
$ make menuconfig
$ make -j16
$ make INSTALL_MOD_PATH=/mnt/rootfs modules_install
$ cp arch/arm64/boot/dts/broadcom/bcm2712-rpi-5-b.dtb /mnt/boot/
$ cp arch/arm64/boot/Image /mnt/boot/kernel_2712.img
```



- Yes, we know how to compile kernels, manually or using a tool (Yocto, Buildroot)
- But many system makers don't provide timely kernel upgrades (*ship and run*)
- On x86_64 distros, the distributions are updated very quickly after vulnerabilities are found. However, these kernels can be quite outdated.
- Solution for people managing their own kernels: use a supported *stable* release from kernel.org.

The Linux Kernel Archives



[About](#) [Contact us](#) [FAQ](#) [Releases](#) [Signatures](#) [Site news](#)

Protocol	Location
HTTP	https://www.kernel.org/pub/
GIT	https://git.kernel.org/
RSYNC	rsync://rsync.kernel.org/pub/

Latest Release
6.13.2 

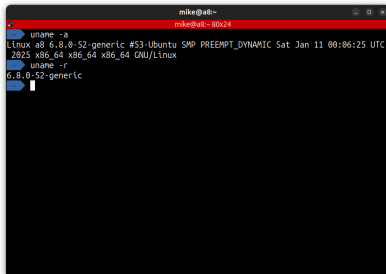
mainline:	6.14-rc2	2025-02-09	[tarball]	[patch]	[inc. patch]	[view diff]	[browse]		
stable:	6.13.2	2025-02-08	[tarball]	[pgp]	[patch]	[inc. patch]	[view diff]	[browse]	[changelog]
stable:	6.12.13	2025-02-08	[tarball]	[pgp]	[patch]	[inc. patch]	[view diff]	[browse]	[changelog]
longterm:	6.6.77	2025-02-11	[tarball]	[pgp]	[patch]	[inc. patch]	[view diff]	[browse]	[changelog]
longterm:	6.1.128	2025-02-01	[tarball]	[pgp]	[patch]	[inc. patch]	[view diff]	[browse]	[changelog]
longterm:	5.15.178	2025-02-01	[tarball]	[pgp]	[patch]	[inc. patch]	[view diff]	[browse]	[changelog]
longterm:	5.10.234	2025-02-01	[tarball]	[pgp]	[patch]	[inc. patch]	[view diff]	[browse]	[changelog]
longterm:	5.4.290	2025-02-01	[tarball]	[pgp]	[patch]	[inc. patch]	[view diff]	[browse]	[changelog]
linux-next:	next-20250212	2025-02-12						[browse]	

<https://kernel.org>

We will show how to boot a custom kernel on the latest releases of the most popular distros:

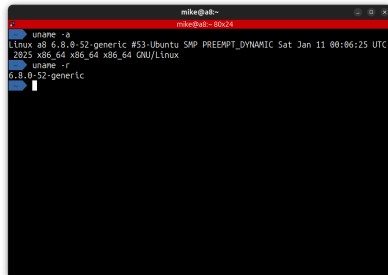
- Ubuntu 24.04: Linux 6.8.x
- Debian 12.9: Linux 6.1.x (LTS)
- Fedora 41: Linux 6.12.x (LTS)
- OpenSUSE Leap 15.6: Linux 6.4.x
- AlmaLinux 9.5 (Red Hat Enterprise Linux compatible): 5.14.x !

We chose the distros supported by the Yocto Project.
Missing: ArchLinux and Gentoo (rolling releases, their advanced users probably know how to update their kernels).

A terminal window titled 'mike@a8--' with a red header bar. The terminal shows the boot process of a custom kernel. The prompt is 'uiane -a', followed by the kernel version 'Linux a8 6.8.0-52-generic #53-Ubuntu SMP PREEMPT_DYNAMIC Sat Jan 11 00:06:25 UTC 2025 x86_64 x86_64 GNU/Linux'. The prompt changes to 'uiane -r' and then '6.8.0-52-generic', with a cursor at the end of the last line.

```
mike@a8--
uiane -a
Linux a8 6.8.0-52-generic #53-Ubuntu SMP PREEMPT_DYNAMIC Sat Jan 11 00:06:25 UTC
2025 x86_64 x86_64 GNU/Linux
uiane -r
6.8.0-52-generic
```

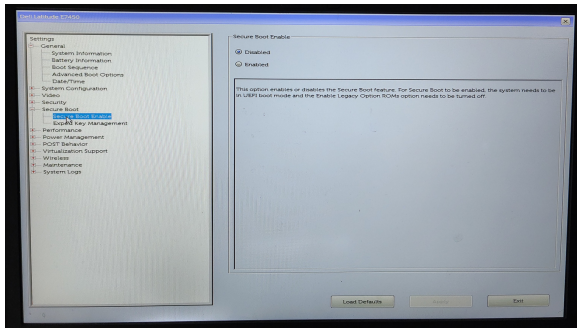
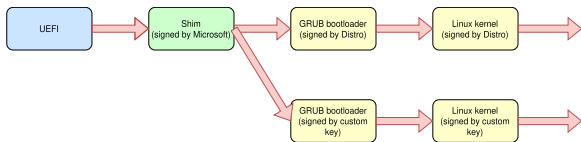
- My laptop has a suspend to RAM issue after Ubuntu 22.04.
- Issue fixed by compiling the kernel with a different configuration
- Originally re-compiling the kernel on my PC and installing it "manually".
- Learned how to do this through packages (better integration with the distribution custom scripts).

A terminal window with a red title bar. The terminal shows the output of the 'uname' command. The first command is 'uname -a', which outputs: 'Linux m8 6.8.0-52-generic #53-Ubuntu SMP PREEMPT_DYNAMIC Sat Jan 11 00:06:25 UTC 2025 x86_64 x86_64 x86_64 GNU/Linux'. The second command is 'uname -r', which outputs: '6.8.0-52-generic'.

```
mike@m8:~  
└─$ uname -a  
Linux m8 6.8.0-52-generic #53-Ubuntu SMP PREEMPT_DYNAMIC Sat Jan 11 00:06:25 UTC 2025 x86_64 x86_64 x86_64 GNU/Linux  
└─$ uname -r  
6.8.0-52-generic
```


Running a custom kernel

- Needed because only signed kernels from the distribution vendors will be accepted otherwise
- It's still possible to create Machine Owner Keys (MOK) to sign custom kernels, but the workflow is more complicated.
- See <https://wiki.debian.org/SecureBoot>



- Ubuntu, Debian

```
sudo apt install build-essential git flex bison libelf-dev libssl-dev \  
libncurses-dev rsync debhelper
```

- Fedora, AlmaLinux

```
sudo dnf install kernel-devel git-core ncurses-devel rpm-build dwarves openssl perl
```

- OpenSUSE

```
sudo zypper install git make gcc flex bison dwarves \  
libopenssl-devel libelf-devel bc ncurses-devel rpmbuild
```

Same for all distros!

```
git clone https://git.kernel.org/pub/scm/linux/kernel/git/stable/linux.git
cd linux
git tag --sort=-creatordate
git checkout v6.13.2
cp /boot/config-`uname -r` .config      # Configuration for running kernel
make olddefconfig                       # Applies default settings for new parameters
```

Did you know?

- The *stable* tree contains all the commits in *master* too, including the *-rc* tags. That's the best tree for production work.
- To reuse an older configuration, you could use `make oldconfig` too. This allows to choose a setting (default or not) for each parameter that is not defined yet.

Useful to add new features or to remove unnecessary drivers (smaller kernel, faster to build)

make menuconfig

or

make nconfig

Did you know?

- make nconfig is more modern than make menuconfig and automatically adapts to the colors of your terminal. At last, it also offers the possibility to set values directly from search results.

```
./config - Linux/x86 6.12.13 Kernel Configuration
Linux/x86 6.12.13 Kernel Configuration
  General setup --->
[*] 64-bit kernel
  Processor type and features --->
[*] Mitigations for CPU vulnerabilities --->
  Power management and ACPI options --->
  Bus options (PCI etc.) --->
  Binary Emulations --->
[*] Virtualization --->
  General architecture-dependent options --->
[*] Enable loadable module support --->
-* Enable the block layer --->
  Executable file formats --->
  Memory Management options --->
[*] Networking support --->
  Device Drivers --->
  File systems --->
  Security options --->
-* Cryptographic API --->
  Library routines --->
  Kernel hacking --->
F1 Help F2 SymInfo F3 Help 2 F4 ShowAll F5 Back F6 Save F7 Load F8 SymSearch F9 Exit
```

Distribution specific configuration tweaks (1)

Mainly needed to disable kernel and module signing, with distribution specific keys that we don't have.

- Ubuntu, AlmaLinux: open `.config` and empty the settings containing the `.pem` string.
Also disable `CONFIG_MODULE_SIG_ALL`.

```
#
# Certificates for signature checking
#
CONFIG_MODULE_SIG_KEY="certs/signing_key.pem"
CONFIG_MODULE_SIG_KEY_TYPE_RSA=y
# CONFIG_MODULE_SIG_KEY_TYPE_ECDSA is not set
CONFIG_SYSTEM_TRUSTED_KEYRING=y
CONFIG_SYSTEM_TRUSTED_KEYS="debian/canonical-certs.pem"
CONFIG_SYSTEM_EXTRA_CERTIFICATE=y
CONFIG_SYSTEM_EXTRA_CERTIFICATE_SIZE=4096
CONFIG_SECONDARY_TRUSTED_KEYRING=y
# CONFIG_SECONDARY_TRUSTED_KEYRING_SIGNED_BY_BUILTIN is not set
CONFIG_SYSTEM_BLACKLIST_KEYRING=y
CONFIG_SYSTEM_BLACKLIST_HASH_LIST=""
CONFIG_SYSTEM_REVOCATION_LIST=y
CONFIG_SYSTEM_REVOCATION_KEYS="debian/canonical-revoked-certs.pem"
# CONFIG_SYSTEM_BLACKLIST_AUTH_UPDATE is not set
# end of Certificates for signature checking
```



```
#
# Certificates for signature checking
#
CONFIG_MODULE_SIG_KEY=""
CONFIG_MODULE_SIG_KEY_TYPE_RSA=y
# CONFIG_MODULE_SIG_KEY_TYPE_ECDSA is not set
CONFIG_SYSTEM_TRUSTED_KEYRING=y
CONFIG_SYSTEM_TRUSTED_KEYS=""
CONFIG_SYSTEM_EXTRA_CERTIFICATE=y
CONFIG_SYSTEM_EXTRA_CERTIFICATE_SIZE=4096
CONFIG_SECONDARY_TRUSTED_KEYRING=y
# CONFIG_SECONDARY_TRUSTED_KEYRING_SIGNED_BY_BUILTIN is not set
CONFIG_SYSTEM_BLACKLIST_KEYRING=y
CONFIG_SYSTEM_BLACKLIST_HASH_LIST=""
CONFIG_SYSTEM_REVOCATION_LIST=y
CONFIG_SYSTEM_REVOCATION_KEYS=""
# CONFIG_SYSTEM_BLACKLIST_AUTH_UPDATE is not set
# end of Certificates for signature checking
```

- Debian, Fedora: nothing to change
- OpenSUSE Leap: had to address this issue:
 - Got this error: `lib/crypto/aesgcm.c:212:29: error: ptext1 causes a section type conflict with aesgcm_tv`
 - ⇒ Checked `lib/crypto/Makefile`:
`obj-$(CONFIG_CRYPTOLIB_AESGCM) += libaesgcm.o`
`libaesgcm-y := aesgcm.o`
 - ⇒ Disabled `CONFIG_SEV_GUEST` to disable `CONFIG_CRYPTOLIB_AESGCM`

- Debian, Ubuntu:

```
make -j16 bindeb-pkg
```

- Fedora, AlmaLinux, OpenSUSE

```
make -j16 binrpm-pkg
```


- Debian, Ubuntu: packages in ../

```
8.8M   linux-headers-6.13.2_6.13.2-4_amd64.deb    --> Headers for compiling out-of-tree modules
66M    linux-image-6.13.2_6.13.2-4_amd64.deb     --> Binary kernel and modules, stripped
872M   linux-image-6.13.2-dbg_6.13.2-4_amd64.deb --> Same but with debugging symbols
1.4M   linux-libc-dev_6.13.2-4_amd64.deb         --> Headers for userspace applications
```

- Fedora, AlmaLinux: packages in rpmbuild/RPMS/x86_64

```
1.1G   kernel-6.13.2-4.x86_64.rpm                --> Binary kernel and modules, stripped
9.6M   kernel-devel--6.13.2-4.x86_64.rpm        --> Headers for compiling out-of-tree modules
1.5M   kernel-headers--6.13.2-4.x86_64.rpm     --> Headers for userspace applications
```

- OpenSUSE Leap: packages in rpmbuild/RPMS/x86_64 too

```
1.2G   kernel-6.13.2_150600.23.33_default-8.x86_64.rpm
9.6M   kernel-devel-6.13.2_150600.23.33_default-8.x86_64.rpm
1.5M   kernel-headers-6.13.2_150600.23.33_default-8.x86_64.rpm
```

- Debian, Ubuntu:

```
sudo dpkg -i linux-headers-6.13.2_6.13.2-4_amd64.deb
sudo dpkg -i linux-image-6.13.2_6.13.2-4_amd64.deb
sudo dpkg -i linux-libc-dev_6.13.2-4_amd64.deb
```

- Fedora, AlmaLinux

```
sudo rpm -i kernel-6.13.2-4.x86_64.rpm
sudo rpm -i kernel-devel--6.13.2-4.x86_64.rpm
```

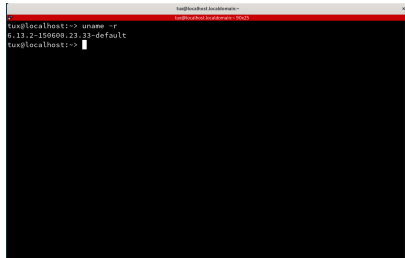
Note: couldn't install kernel headers (not critical as the default headers are usually sufficient):
kernel-headers < 6.13.2 is obsoleted by kernel-headers-6.13.2-4.x86_64

- OpenSUSE Leap: could only install 2 packages too:

```
sudo rpm -i kernel-6.13.2_150600.23.33_default-8.x86_64.rpm
sudi rpm -i kernel-devel-6.13.2_150600.23.33_default-8.x86_64.rpm
```

After rebooting

- Ubuntu, Debian, Fedora, AlmaLinux:
The new kernel is booted by default
- OpenSUSE Leap:
Had to run:
`sudo grub2-mkconfig -o /boot/grub2/grub.cfg`
to detect the new kernel

A terminal window with a black background and white text. The prompt is 'tux@localhost:~'. The user enters 'uname -r' and the output is '6.13.2-150608.23.33-default'. The prompt returns to 'tux@localhost:~'.

```
tux@localhost:~$ uname -r
6.13.2-150608.23.33-default
tux@localhost:~$
```

- Of course, you can remove the packages you installed before
- You can also make sure you get a menu to choose between kernel versions at boot time:
 - Ubuntu:
In `/etc/default/grub`, replace `GRUB_TIMEOUT_STYLE=hidden` by `GRUB_TIMEOUT_STYLE=menu` and then run `sudo update-grub`.
 - Debian, OpenSUSE Leaf:
The menu is already there
 - Fedora, AlmaLinux:
You need to run this command:
`sudo grub2-editenv - unset menu_auto_hide`



Bonus: testing full kernel preemption (merged in 6.12)



cyclictest: measuring worst case scheduling latency

```
sudo cyclictest --mlockall --smp --priority=99 --interval=100 --duration=1m --quiet
[sudo] password for mike:
# /dev/cpu_dma_latency set to 0us
T: 0 ( 5560) P:99 I:100 C: 599963 Min: 1 Act: 18 Avg: 5 Max: 1241
T: 1 ( 5561) P:99 I:600 C: 99990 Min: 1 Act: 20 Avg: 3 Max: 3513
T: 2 ( 5562) P:99 I:1100 C: 54541 Min: 1 Act: 25 Avg: 3 Max: 1154
T: 3 ( 5563) P:99 I:1600 C: 37496 Min: 1 Act: 2 Avg: 3 Max: 2442
T: 4 ( 5564) P:99 I:2100 C: 28568 Min: 1 Act: 34 Avg: 3 Max: 1693
T: 5 ( 5565) P:99 I:2600 C: 23074 Min: 1 Act: 2 Avg: 3 Max: 117
T: 6 ( 5566) P:99 I:3100 C: 19351 Min: 1 Act: 26 Avg: 3 Max: 172
T: 7 ( 5567) P:99 I:3600 C: 16663 Min: 1 Act: 23 Avg: 3 Max: 1256
T: 8 ( 5568) P:99 I:4100 C: 14631 Min: 1 Act: 3 Avg: 4 Max: 40
T: 9 ( 5569) P:99 I:4600 C: 13040 Min: 1 Act: 2 Avg: 3 Max: 802
T:10 ( 5570) P:99 I:5100 C: 11761 Min: 1 Act: 23 Avg: 3 Max: 981
T:11 ( 5571) P:99 I:5600 C: 10711 Min: 1 Act: 3 Avg: 4 Max: 1496
T:12 ( 5572) P:99 I:6100 C: 9833 Min: 1 Act: 3 Avg: 3 Max: 368
T:13 ( 5573) P:99 I:6600 C: 9088 Min: 1 Act: 2 Avg: 3 Max: 571
T:14 ( 5574) P:99 I:7100 C: 8447 Min: 1 Act: 24 Avg: 3 Max: 738
T:15 ( 5575) P:99 I:7600 C: 7891 Min: 1 Act: 25 Avg: 3 Max: 2114

sudo cyclictest --mlockall --smp --priority=99 --interval=100 --duration=1m --quiet
[sudo] password for mike:
# /dev/cpu_dma_latency set to 0us
T: 0 ( 6401) P:99 I:100 C: 599996 Min: 1 Act: 11 Avg: 4 Max: 220
T: 1 ( 6402) P:99 I:600 C: 99996 Min: 1 Act: 2 Avg: 2 Max: 545
T: 2 ( 6403) P:99 I:1100 C: 54541 Min: 1 Act: 2 Avg: 2 Max: 39
T: 3 ( 6404) P:99 I:1600 C: 37496 Min: 1 Act: 2 Avg: 2 Max: 72
T: 4 ( 6405) P:99 I:2100 C: 28567 Min: 1 Act: 2 Avg: 2 Max: 31
T: 5 ( 6406) P:99 I:2600 C: 23073 Min: 1 Act: 2 Avg: 2 Max: 34
T: 6 ( 6407) P:99 I:3100 C: 19350 Min: 1 Act: 2 Avg: 2 Max: 541
T: 7 ( 6408) P:99 I:3600 C: 16662 Min: 1 Act: 2 Avg: 2 Max: 29
T: 8 ( 6409) P:99 I:4100 C: 14630 Min: 1 Act: 2 Avg: 2 Max: 23
T: 9 ( 6410) P:99 I:4600 C: 13039 Min: 1 Act: 2 Avg: 2 Max: 29
T:10 ( 6411) P:99 I:5100 C: 11760 Min: 1 Act: 2 Avg: 2 Max: 32
T:11 ( 6412) P:99 I:5600 C: 10710 Min: 1 Act: 2 Avg: 2 Max: 743
T:12 ( 6413) P:99 I:6100 C: 9832 Min: 1 Act: 2 Avg: 2 Max: 29
T:13 ( 6414) P:99 I:6600 C: 9087 Min: 1 Act: 2 Avg: 2 Max: 36
T:14 ( 6415) P:99 I:7100 C: 8446 Min: 1 Act: 2 Avg: 2 Max: 28
T:15 ( 6416) P:99 I:7600 C: 7890 Min: 1 Act: 2 Avg: 2 Max: 26
```

Standard kernel for Ubuntu
(with CONFIG_PREEMPT_VOLUNTARY)

Standard kernel for Ubuntu
(with CONFIG_PREEMPT_RT)

- Didn't realize that the *stable* tree of Linux also contained everything in *master*
- Didn't know about `make olddefconfig`. Was using `make oldconfig` instead.
- Didn't realize how convenient the generation of packages was compared to the manual approach to build and install a kernel. I can now build my kernels on my bigger desktop machine and deploy them on my laptop through packages.
- Didn't know that custom keys can be added for secure boot with custom kernels. That's still secure as you have to load the custom keys in the UEFI settings (can be password protected).

- Use the Linux *stable* tree to get the source code
- Very similar ways to build the kernel:
 - make `bindeb-pkg` for distros with `.deb` packages (Debian, Ubuntu)
 - make `binrpm-pkg` for distros with `.rpm` packages (Fedora, Red Hat, OpenSUSE)
- Very easy to deploy compiled kernels on several systems, thanks to binary packages.
- However, that should be mostly for testing purposes. Sticking to the distribution vendor kernels is safer as you get (tested) updates in a timely fashion.
- Each distribution can have its own way to deploy the kernel, but that's transparent when the kernel is deployed through binary packages.

Questions? Comments?

- Slides available under the CC-BY-SA 4.0 license
<https://rootcommit.com/pub/conferences/2025/alposs/mainline-linux-on-pc/>
 - Sources (L^AT_EX): <https://gitlab.com/rootcommit/mainline-linux-on-pc/>
 - Blog post: <https://rootcommit.com/2025/mainline-linux-on-pc/>
-
- mo@rootcommit.com
 - XMPP: [omichael@conversations.im](xmpp:omichael@conversations.im)
 - Signal: [rootcommit.01](https://rootcommit.com)



<https://www.youtube.com/watch?v=zeDCODvmo14>