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

AlpOSS 2025

Michael Opdenacker

Root Commit

Feb. 20, 2025

Michael Opdenacker

root commit

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)

April
promouvoir et défendre le logiciel libre

Introduction

Why test the mainline kernel?

root commit

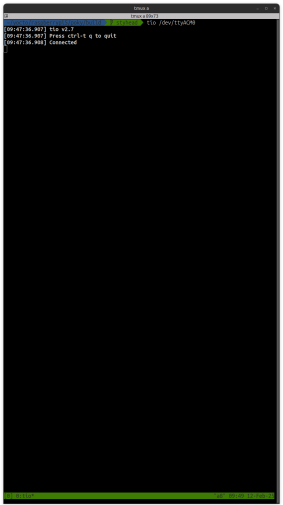
- To get new features or bug fixes
- To test your system on future kernels that you will have sooner or later.
- To contribute to the Linux kernel (writing code, reporting or fixing bugs)
- Because tweaking the kernel is fun!

We do it all the time in embedded

root commit

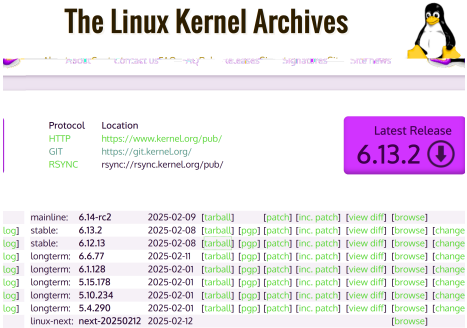
Real life example build: Raspberry Pi 5
Updating an SD card initially generated by Yocto

```
$ 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
```



Though the situation in embedded is worse

- 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 However, these kernels can be quite outdated.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.



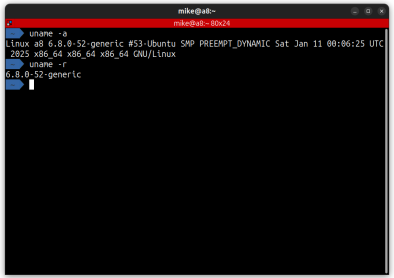
`https://kernel.org`

Goal: cover most useful distributions

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 Entreprise Linux compatible): 5.14.x !

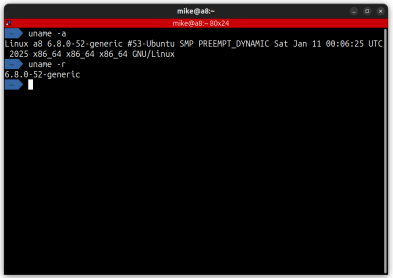
We chose the distros supported by the Yocto Project.
Missing: ArchLinux and Gentoo (rolling releases, their advanced users probably know how to update their kernels).



Motives for creating this presentation

root commit

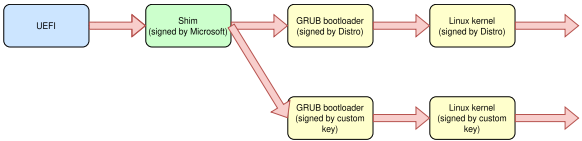
- My laptop has a suspend to RAM issue after Ubuntu 22.04.
- Issue fixed by compiling the kernel with a different configuration
- Originally re-compiling the kernel on my PC and installing it "manually".
- Learned how to do this through packages (better integration with the distribution custom scripts).

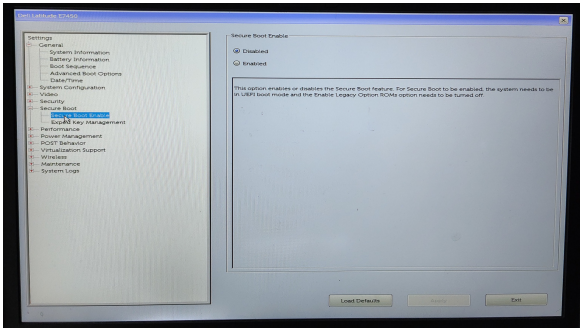


Running a custom kernel

Disable secure boot

- 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





Installing required development software

root commit

- 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
```

Get the Linux sources

root commit

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.

Customizing kernel configuration

root commit

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

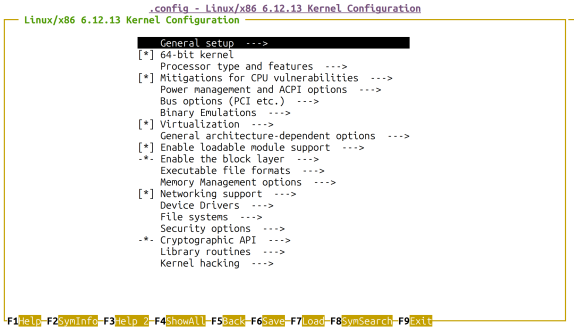
```
make menuconfig
```

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.



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
```

Distribution specific configuration tweaks (2)

root commit

- 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_CRYPTO_LIB_AESGCM) += libaesgcm.o`
    `libaesgcm-y := aesgcm.o`
  - ⇒Disabled `CONFIG_SEV_GUEST` to disable `CONFIG_CRYPTO_LIB_AESGCM`

Building the kernel

root commit

- Debian, Ubuntu:

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

- Fedora, AlmaLinux, OpenSUSE

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

Generated package files

root commit

- 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
```

Installing the new kernel

root commit

- 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:

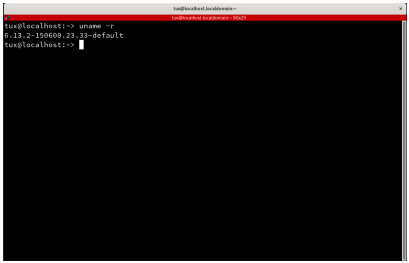
```
sudo rpm -i kernel-6.13.2_150600.23.33_default-8.x86_64.rpm
sudi rpm -i kernel-devel-6.13.2_150600.23.33_default-8.x86_64.rpm
```

Booting the newly installed kernel

root commit

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



Keeping the possibility to boot the default kernel

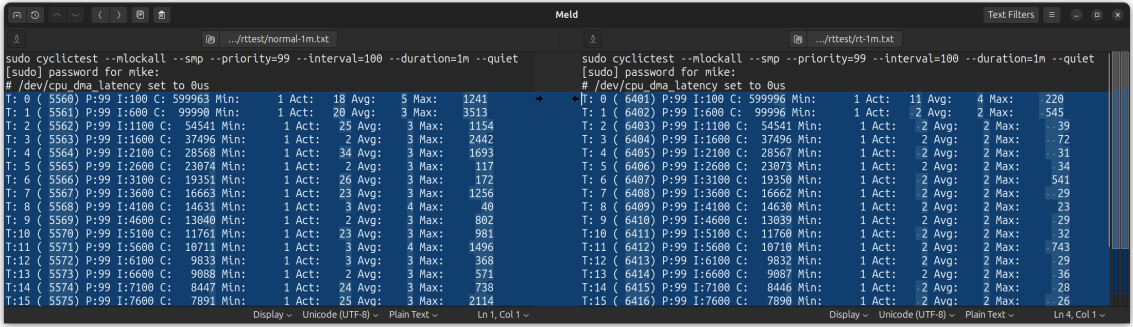
root commit

- 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)

root commit

cyclictest: measuring worst case scheduling latency



Standard kernel for Ubuntu
(with `CONFIG_PREEMPT_VOLUNTARY`)

Standard kernel for Ubuntu
(with `CONFIG_PREEMPT_RT`)

What I learned preparing this talk

root commit

- 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).

Things to remember

root commit

- 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 (LaTeX): `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
- Signal: rootcommit.01

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