



Reverse Engineer  
a Bluetooth device



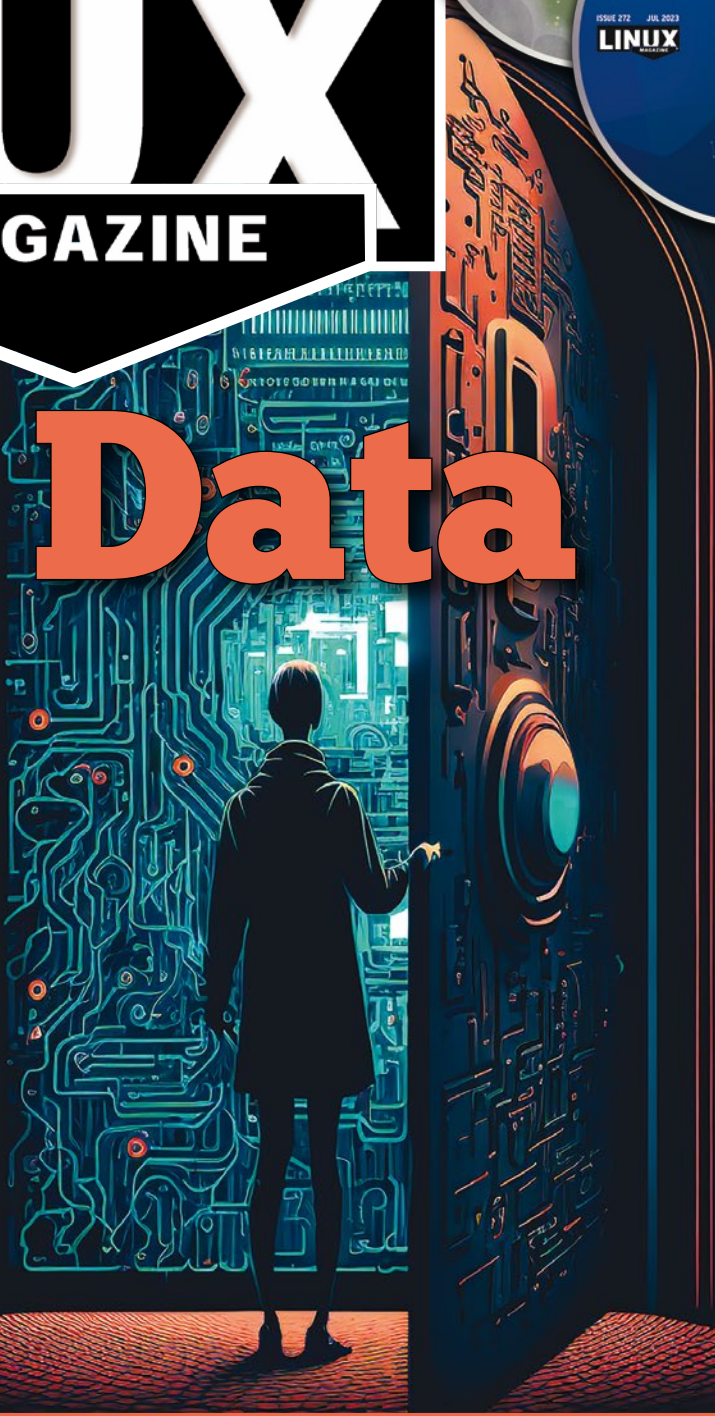
# LINUX

MAGAZINE

ISSUE 272 – JULY 2023

# Open Data

FOSS ideals meet  
the information  
revolution



**FPGAs:** Get started with programmable hardware

**Roll Your Own ISO Images**

**Espanso:** Save typing time with a text expander

**Lenticular Images**  
Make your pictures wiggle

**nnn:** Keep it simple with a GUI-free file manager

**10** STUPENDOUS FOSS TOOLS!





16:10-16 inch display und maximum sized 99 Wh battery. Compatible with the TUXEDO Aquaris.

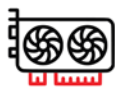


GeForce RTX 4090, Core i9-13900HX and a mechanical keyboard with Cherry MX switches.

## Interstellar performance in a compact form factor! **TUXEDO Stellaris 16 - Gen5**



CPU performance



GFX performance



Mobility



Battery life



100%  
Linux

5

Year  
Warranty



Lifetime  
Support



Built in  
Germany



Germany  
Privacy



Local  
Support

# TUXEDO

[tuxedocomputers.com](https://tuxedocomputers.com)

# WHO'S DRIVING?

Dear Reader,

I happen to be writing this column on a day when the US Senate is conducting hearings on artificial intelligence (AI) and, specifically, whether a need exists for greater regulation. One of the people testifying is Sam Altman, CEO of OpenAI, the company behind ChatGPT. CEOs of companies that are about to be the subject of regulation often come to Congress with dire warnings about how bad further regulation will be for their businesses. Is it refreshing, or is it alarming, that Altman is taking a different view and calling for more government oversight?

Altman says that his worst fear is that AI “could cause significant harm to the world,” adding “If this technology goes wrong, it can go quite wrong” [1]. Who better to warn us about these potential consequences than an industry insider who is directly involved with developing and marketing the technology? And yet, Altman is not a whistle-blower who is resigning because of his misgivings. He is one of the guys who is making it happen, and he isn't saying he wants to stop. He is just saying he wants government to set up some rules.

It is commendable that a CEO would call for more regulation of his industry, yet I can't help but feeling a little frustrated that all the onus is on the government and that individuals (as well as companies) working in this industry would not be expected to exercise some self-restraint about building a technology that they themselves feel “could cause significant harm to the world.” NYU professor Gary Marcus, who also testified, offered a more balanced perspective when he warned of AI becoming a “perfect storm of corporate irresponsibility, widespread deployment, lack of regulation, and inherent unreliability” [2].

The senators played to the cameras, looking for sound bites and attempting to appear august, but in this case, I can sympathize with the difficulties they seem to have with understanding this issue enough to know how to regulate it. In the old days, people thought they could get computers to “think” like a person by just defining the right rules, but modern generative AI systems find their own way to the answer, with no clear path that anyone can follow later to show how they got there other than that someone (who?) might know what data was used for training.

## Info

- [1] Sam Altman's opening statement:  
<https://www.judiciary.senate.gov/imo/media/doc/2023-05-16-%20-%20Bio%20&%20Testimony%20-%20Altman.pdf>
- [2] Gary Marcus' opening statement:  
<https://www.judiciary.senate.gov/imo/media/doc/2023-05-16-%20-%20Testimony%20-%20Marcus.pdf>
- [3] The AI Act: <https://artificialintelligenceact.eu/>
- [4] Christina Montgomery's opening statement:  
<https://www.ibm.com/policy/wp-content/uploads/2023/05/Christina-Montgomery-Senate-Judiciary-Testimony-5-16-23.pdf>

I have read several news stories and opinion columns on the importance of regulating AI, yet I have seen few details on what this regulation would look like. Rather than writing another one of those opinion columns, I'll tell you what I know. For Altman, regulation means setting requirements for testing to ensure that AI meets “safety requirements.” His concept of safety encompasses several parts, including privacy, accuracy, and disinformation prevention. In his opening statement before the Senate, he states, “it is vital that AI companies – especially those working on the most powerful models – adhere to an appropriate set of safety requirements, including internal and external testing prior to release and publication of evaluation results. To ensure this, the U.S. government should consider a combination of licensing or registration requirements for development and release of AI models above a crucial threshold of capabilities, alongside incentives for full compliance with these requirements” [1].

Many computer scientists have also talked about the need for transparency in disclosing the dataset that was used for training the AI, so that others can check it and search for potential bias. This step seems essential for ensuring accurate and non-discriminatory AI systems, but we'll need to develop new systems for checking these datasets that can sometimes include millions of pages of data.

The EU already has a proposed law on the table [3]. I am not a legal expert (or an EU expert), but part of the AI Act appears to regulate the behavior of the AI, as opposed to the development process, by prohibiting activities such as subliminal manipulation, social scoring, exploitation of children or the mentally disabled, and remote biometric identification by law enforcement. Beyond these prohibited activities, other uses are classified into three different risk categories with accompanying requirements for each category. The requirements address the need for training, testing, and documentation.

I applaud the EU for getting some proposed legislation out on the table. However, the act was written two years ago, and it already sounds a little anachronistic in the ChatGPT era. Things we are worrying about now weren't even imagined then, like what if an AI steals your copyright or deepfakes you into a porn movie?

Times are rapidly changing. We need to be careful, and governments need to be unified in addressing the problem. IBM Chief Privacy and Trust Officer Christina Montgomery, who also testified at the Senate hearing, put it best in summarizing the need for “clear, reasonable policy and guardrails.” Montgomery warns that “The era of AI cannot be another era of move fast and break things” [4].



Joe Casad,  
Editor in Chief

## ON THE COVER

### 30 Custom Bootable ISO Images

Get rid of the bloat and tailor your Linux for the task with a custom ISO image.

### 46 Reverse Engineering a Bluetooth Device

With a little ingenuity and some basic Python skills, you can talk to your Bluetooth clock.

### 58 DIY Lenticular Camera

A Raspberry Pi and the Camarray HAT are all you need to build a special-purpose camera.

### 64 FPGA Simulations

If you're ready to join the FPGA craze, you'll need to learn a little about hardware description languages.

### 75 Espanso Text Expander

Save your fingers and save some time with a tool that helps you with the typing.

### 80 nnn File Manager

Manage your files without the clutter of a glitzy graphic desktop.

## NEWS

### 8 News

- New Release of Br OS Includes ChatGPT Integration
- Command-Line Only Peropesis 2.1
- TUXEDO Computers Announces InfinityBook Pro 14
- Linux Kernel 6.3 Release Includes Interesting Features
- Arch-Based blendOS Features Cool Trick
- Fedora 38 Released with New Features
- LXQt 1.3 Released with Bug Fixes
- 4MLinux 42.0 Now Ready for Prime Time

## COVER STORIES

### 12 Open Data

The open data movement extends the ideals of open source to government data and, ultimately, all the world's knowledge.

### 16 Data Transfer Project

The Data Transfer Project wants to make it easier to move your data between social media sites.

### 20 Open Data with CKAN

CKAN, a versatile data management system, lets you build a portal to share your open data.

### 24 Interview – ODI's Lisa Allen

The Open Data Institute's Lisa Allen explains why open data matters and what it will take for more widespread adoption.

95 Back Issues

97 Call for Papers

96 Events

98 Coming Next Month

## REVIEW

### 28 Distro Walk – One-Stop Linux Distros

Linux users can now experience what Windows and macOS users have enjoyed for decades: hardware, software, and services bundled together. We look at six of these one-stop solutions for Linux.

## IN-DEPTH

### 30 Custom Bootable ISO Images

If you are looking to customize your Linux distribution, we show you three graphical front ends for creating bootable ISO images.

### 36 Custom Slackware Repository

If you deploy software packages to several computers, the standard Slackware tools lack efficiency. We show you how to create a custom repository to automatically install and upgrade software for multiple systems.

### 42 Command Line – coreboot

Coreboot lets you build your own custom firmware while learning more about Linux.

### 46 Reverse Engineering Bluetooth

What do you do when all your Bluetooth clocks show slightly different times? With some reverse engineering, you can write a Python program to synchronize your clocks.

### 52 Programming Snapshot – Remote Backup

To be able to power up and shut down his NAS and check the current status without getting out of his chair, Mike Schilli programs a graphical interface that sends a Magic Packet in this month's column.

## Open Data

As long as governments have kept data, there have been people who have wanted to see it and people who have wanted to control it. A new generation of tools, policies, and advocates seeks to keep the data free, available, and in accessible formats. This month we bring you snapshots from the quest for open data.

## MakerSpace

### 58 DIY Lenticular Camera

You can take lenticular images with a home-made camera to recreate the “wiggle” pictures of your childhood.

### 64 Digital IC Simulation on Linux

Designing field-programmable gate arrays is only half the job: The hardest part is the simulation, but Linux is the best place to tackle certain challenges.

 @linux\_pro

 @linuxpromagazine

 Linux Magazine

 @linuxmagazine



**TWO TERRIFIC DISTROS  
DOUBLE-SIDED DVD!**

**SEE PAGE 6 FOR DETAILS**

## LINUXVOICE

### 73 Welcome

This month in Linux Voice.

### 74 Doghouse – Databases

There are many FOSS databases available inexpensively today, and they might serve new projects well.

### 75 Espanso

Espanso is a cross-platform text expander that can do far more than simply replace text modules.

### 80 nnn

If you're a Linux lover, you'll know the command line is the slickest and most efficient way to interact with the system. Free yourself from point-and-click with developer jarun's nnn command-line file manager.

### 84 FOSSPicks

This month Graham looks at Seamly2D, Arianna, X-Pipe, Nosey Parker, IEM Plug-in Suite, Open Hexagon, and more!

### 90 Tutorial – Carbonyl Graphical Web Browser

This Chromium port can run inside any console, with minimal resources, and is a great tool for making old computers really useful – and learning programming along the way.

# Xubuntu 23.04 and Fedora Workstation 38

## Two Terrific Distros on a Double-Sided DVD!

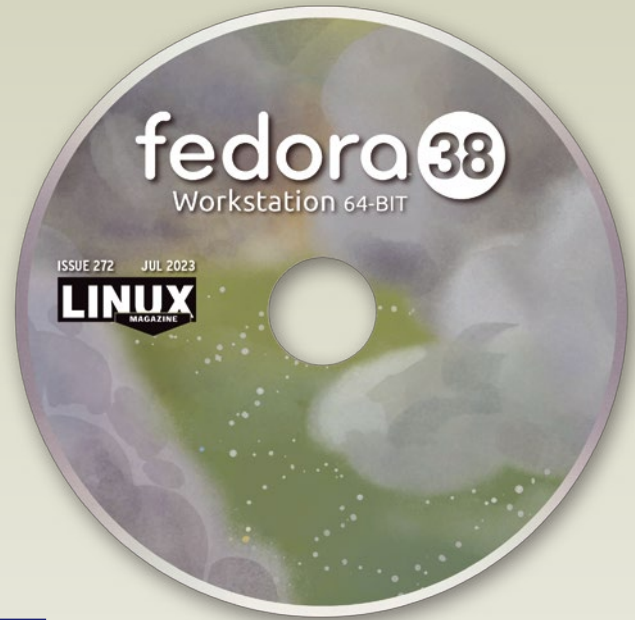


**Xubuntu 23.04**  
64-bit

Xubuntu 23.04, codenamed Lunar Lobster, is the latest version of the Ubuntu flavor that uses Xfce 4.18 as its default desktop. It will be supported until January 2024. Like all Xubuntu releases, this release is ideal for those who prefer a middleweight desktop that does not skimp on features.

Traditionally, Ubuntu's short-term releases do not include many major improvements, and Lunar Lobster is no exception. However, it still has several innovations that are worth checking out. To start with, it is the first version of Xubuntu to use PipeWire and WirePlumber, which together offer improved sound and sound management. The default terminal font is now 10 points, up from nine, a small but noticeable gain for easy reading. In addition, the image now includes Xubuntu Minimal for those who prefer to control the contents of their installation rather than rely on default curated applications.

With Xfce 4.18 also comes a series of other tweaks. As usual, many are to the Thunar file manager, such as an image preview, an editable toolbar and custom highlighting of selected files, and undo/redo support for a larger number of operations. The desktop in general includes numerous small tweaks, as well as improved Wayland support.



**Fedora Workstation 38**  
64-bit

Fedora Workstation is the Fedora release for general desktop users. It is the first appearance of many features that will eventually find their way into distributions such as Rocky Linux, AlmaLinux, and Red Hat Enterprise Linux (RHEL).

New features include redesigns of mouse and touchpad, sound, and accessibility settings; clearer language for device security; and the ability to connect to WiFi via QR Codes. Management of third-party package repositories is also improved. Other changes include the use of GNOME 44, including icon views and previews in the file chooser and added Bluetooth support in the Quick Settings dialog. The general tendency of these changes is to a more pictorial design, as it has been for well over a decade. The result is a well-organized user experience that does not sacrifice functionality and is accessible to all levels of users.

*Defective discs will be replaced. Please send an email to [subs@linux-magazine.com](mailto:subs@linux-magazine.com).*

*Although this Linux Magazine disc has been tested and is to the best of our knowledge free of malicious software and defects, Linux Magazine cannot be held responsible and is not liable for any disruption, loss, or damage to data and computer systems related to the use of this disc.*

# Attention Newsstand Readers

**Welcome to the latest issue of *Linux Magazine*!**

We hope you enjoy this month's selection of technical articles, tutorials, and projects.

You may have noticed an increase in the cover price this month. Unfortunately, the rising costs of selling *Linux Magazine* on the newsstand required us to raise our rates.

**You can save money and get *Linux Magazine* faster if you buy direct from us!** You will always get the best price, and your direct support lets us continue to produce *Linux Magazine* every month.



US/Canada Customers  
[linuxnewmedia.square.site](http://linuxnewmedia.square.site)



Customers in All Other Countries  
[sparkhaus-shop.com](http://sparkhaus-shop.com)

**Thank you for your continued support!**



# NEWS

Updates on technologies, trends, and tools

## THIS MONTH'S NEWS

- 08 • New Release of Br OS Includes ChatGPT Integration
- Command-Line Only Peropesis 2.1
- 09 • TUXEDO Computers Announces InfinityBook Pro 14
- Linux Kernel 6.3 Release Includes Interesting Features
- More Online
- 10 • Arch-Based blendOS Features Cool Trick
- Fedora 38 Released with New Features
- 11 • LXQt 1.3 Released with Bug Fixes
- 4MLinux 42.0 Now Ready for Prime Time

### New Release of Br OS Includes ChatGPT Integration

If you're a web content creator, Br OS might be a great option for you. Only in its third year, this Brazilian-born Linux distribution, created by Anderson Marques, focuses on having an operating system ready to use from the moment of installation.

With Br OS, users will enjoy image- and video-editing tools, security tools, trace hiding, and tools for publishing content even in countries with authoritarian governments.

However, it's the integration of ChatGPT that might garner the attention of a lot of Linux users. The developer has integrated ChatGPT in such a way that it's easily accessible from the desktop menu and doesn't have access to local data.

ChatGPT runs isolated within the application. By clicking the ChatGPT icon, you can log in with your OpenAI account and start using the service. As you use ChatGPT, the only data the service can collect is what you type into it.

This latest release of Br OS includes Linux kernel 6.2, KDE Plasma 5.27.4, and QT 5.15.8.

You can read all about the latest version from the release notes (<https://br-os.com/blog?p=390>) (in Portuguese) and download an ISO for installation from the official Br OS site (<https://br-os.com/download/>).

### Command-Line Only Peropesis 2.1

Peropesis might not be for everyone but if you're looking for a small-scale, minimal, GUI-less operating system to use as the base for something special, this might be what you're looking for.

Peropesis includes only free software, such as console tools for email, web browsing, data recovery, and development. You can use Peropesis for educational purposes, fixing a broken OS, data recovery, development, and more.

Peropesis 2.1 adds automake and autoconf into the mix and also includes tools such as Perl, lzip, zstd, GNU GCC and g++, bash, binutils, bzip2, pkg-config, tar, wget, and more. You can find the complete package list here (<https://peropesis.org/bundle/>).

New updates for version 2.1 include coreutils 9.3, e2fsprogs 1.47.0, elfutils 0.189, glibc 2.37, grep 3.10, guile 3.0.9, iana-etc 20230418, krb5 1.20.1, libcap 2.68, Links 2.29, Linux kernel 6.3.0, linux-firmware 20230404, lzlib 1.13, Make 4.4.1, OpenSSL 3.1.0, Procps-ng 4.0.3, sqlite 3.41.2, wireless-regdb-master 2023-02-13, xz 5.4.2.

You can download a live ISO (<https://peropesis.org/>), which isn't intended to be written to a drive but run directly from memory. The ISO is very small (205MB) and should run on aging hardware without a problem.



## TUXEDO Computers Announces InfinityBook Pro 14

TUXEDO Computers has refreshed their popular InfinityBook Pro 14 with a 16:10 3K display, a 99 Wh battery, and full Linux support.

According to TUXEDO, “The unique combination of an extremely light and slim magnesium body with an exceptionally powerful high-end processor for this form factor and the maximum 99 Wh battery capacity allowed for airplane carry-on combines maximum mobility with strong performance for business, multimedia, and even image and video editing on a premium 14-inch high-resolution display.”

The 99 Wh battery delivers a reported 16 hours of idle time and is charged, via the USB-C (Power Delivery DC-in) or the 90-watt power supply.

As far as CPU goes, the refreshed InfinityBook Pro 14 includes an Intel Core i7-13700 (14 cores and 20 threads) that can be driven with 40 watts (instead of the weaker 15 watts of energy-saving CPUs in competing ultrabooks), thereby turning this laptop into a beast of a mobile workstation. And with dual-fan cooling, there shouldn't be any concern about heat.

TUXEDO also includes the pre-installed TUXEDO Control Center, so users can easily adjust the power limits for the CPU (between 5 and 45 watts).

The display is 3K and the chassis is a slim 17mm thick with a 1.3-kg magnesium case. You can equip the TUXEDO InfinityBook Pro with up to 2 32 GB DDR5-6400 RAM and 1 M.2 SSD (PCI-Express 4.0) with up to 4 TB of internal storage.

The TUXEDO InfinityBook Pro 14 can be pre-ordered today (<https://www.tuxedo-computers.com/en/TUXEDO-InfinityBook-Pro-14-Gen8>); at the time of writing, delivery is scheduled to start at the end of May.

The base configuration of the InfinityBook Pro 14 - Gen8 with Intel Core i7-13700H, a 14-inch Omnia display, 2 x 8 GB DDR5 RAM, and a 250 GB Samsung 980 SSD, as well as TUXEDO OS preinstalled, has an entry-level price of EUR1,427.73 (excl. VAT).

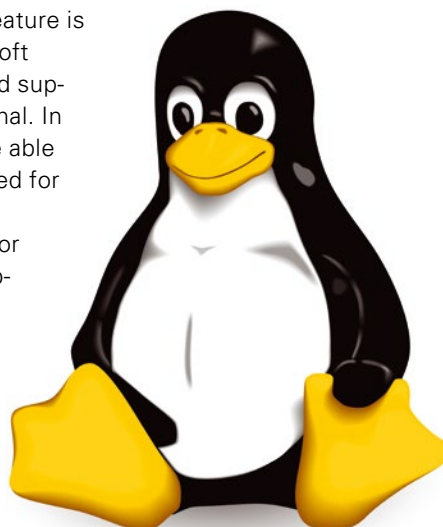
## Linux Kernel 6.3 Release Includes Interesting Features

For those who like to use the latest/greatest Linux kernel, version 6.3 has a few features that just might excite you.

On the top of that list is support for multi-actuator hard disk drives. Such drives are being more widely adopted for hyper-scale cloud deployments because of their ability to enable faster performance by way of a second set of read/write heads.

As well, Linux kernel 6.3 adds support for Kernel Address Space Layout Randomization, but this security feature is only available to Loongson RISC-V CPUs. Microsoft added nested hypervisor support for Hyper-V, and support for Intel's Meteor Lake CPUs is fully functional. In addition, with the 6.3 kernel, Intel GB NIC will be able to break the 60-percent throttle speed it's suffered for the past few years.

Other features/improvements include support for the upcoming AMD CPUs and graphics cards, support for AMD's Indirect Branch Restricted Speculation (to help mitigate Spectre vulnerabilities), support for the Scalable Matrix Extension 2 instructions for ARM architecture, support for AES-SHA2-based encryption with NFS, EXT4 optimizations, a faster Btrfs driver, and a native Steam Deck controller interface in HID.



## MORE ONLINE

### Linux Magazine

[www.linux-magazine.com](http://www.linux-magazine.com)

### ADMIN HPC

<http://www.admin-magazine.com/HPC/>

#### Updates and Upgrades in HPC

• Jeff Layton

What is the difference between an update and an upgrade? We'll look at why, once an HPC distribution is installed, you are likely to see one or two minor release updates, but rarely an upgrade.

### ADMIN Online

<http://www.admin-magazine.com/>

#### Network Monitoring with Zeek

• Matthias Wübbeling

Zeek offers an arsenal of scripts for monitoring popular network protocols and comes with its own policy scripting language for customization.

#### Analysis Tour with Binary Ninja

• Matthias Wübbeling

Binary analysis is an advanced technique used to work through cyberattacks and malware infestations and is also known as reverse engineering. We show you how to statically analyze binary programs with Binary Ninja, an interactive binary analysis platform.

#### IAM for Midmarket Companies

• Martin Kuppinger

We look at the role of identity and access management in midmarket organizations.

Although Linux kernel 6.3 has yet to make it into the standard repositories, you can download it from kernel.org (<https://kernel.org/>) and compile it yourself. Eventually, this kernel will arrive in your distribution's repositories, and it's generally best to wait.

For more information on the new kernel, check out part 1 (<https://lwn.net/Articles/923846/>) and part 2 (<https://lwn.net/Articles/924384/>) of the merge window.

## Arch-Based blendOS Features Cool Trick

Imagine being able to install apps from Arch, Fedora, Ubuntu, Android, and even the web onto a single distribution. That's what blendOS (<https://blendos.co/>) does.

Created by the same maintainer as Ubuntu Unity, blendOS allows you to use the native Arch package manager (pacman), as well as DNF (from Fedora) and Apt (from Ubuntu). Both DNF and Apt are made available as containers by way of the Podman container runtime engine (with a bit of help from Distrobox for init and NVIDIA driver support).

Even though blendOS is based on Arch Linux, the developers have made it such that you can use any app from a supported distro. On top of that, you can choose between either the Gnome or KDE Plasma desktop.

When you install applications using the containerized versions of DNF or Apt, those applications will be available from the base OS (so you don't have to run them from the command line as typical Podman containers).

blendOS is also a rolling and immutable operating system that allows you to install system packages as you normally would, while still being able to roll back to an existing snapshot.

You can also use the blendOS build scripts (<https://github.com/blend-os/blendiso>) to create and submit your own blendOS remix with the desktop environment of your choice.

You can read more about the new release of blendOS here (<https://blendos.co/blend-os-v2/>).

## Fedora 38 Released with New Features

The developers of Fedora Linux have gifted us with yet another stellar release, by way of number 38. It's a surprise, given the team rarely delivers early but it's certainly an early gift from a dedicated team.

Fedora 38 stars Gnome 44 and Linux kernel 6.2. The version of Gnome used in Fedora is straight-up standard, so there are zero changes made by the Fedora team.

Some of the new features found in Gnome 44 include expandable folders in the Files file manager as well as the icon/thumbnaill mode for the file picker.

You'll also find full access to Flathub in the Software application, which means you'll find even more flatpak apps available for installation through third-party repositories.

Other additions include Mesa 23.0, Ruby 3.2, gcc 13, LLVM 16, Golang 1.20, and PHP 8.2.

With the release of Fedora 38, the team has made some significant changes to the website (<https://fedoraproject.org/>) and has included a few new spins, including one for Budgie, Sway, and one for mobile devices called Phosh.

You can read more about the latest release from the official blog (<https://fedora-magazine.org/announcing-fedora-38/>) and download an ISO from the download page (<https://fedoraproject.org/workstation/download/>).



**Get the latest news  
in your inbox every  
week**

**Subscribe FREE  
to Linux Update  
[bit.ly/Linux-Update](https://bit.ly/Linux-Update)**



## LXQt 1.3 Released with Bug Fixes

---

LXQt is a fan-favorite Linux desktop environment for systems with meager resources. Suitable for older systems (but it will run like an absolute champ on new hardware), LXQt is not just easy on hardware, but easy to use.

With the latest release, version 1.3, the developers have remained with Qt 5.15 (although work is being done for initial Qt6 support) and have focused on adding new features.

For example, the LXQt Panel now defaults to include the DOM plugin and fixes a bug in the clock widget, as well as a bug discovered when switching between the dark and light themes. Additionally, the panel will correctly position the context menu when used with Wayland.

The default file manager, PCManFM-Qt has fixed a bug that prevented desktop items from “shaking” on configuration changes. Also, the latest iteration of the file manager adds a title for Desktop to help set window manager rules for certain Wayland compositors. You’ll also find that smooth scrolling can now be disabled in all view modes.

There is also now support for procs-ng  $\geq$  4.0.0 in the LXQt session tool for better detection of window managers and system trays.

Although the new feature and bug list isn’t exactly extensive, there were a lot of minor bug fixes added across the entire desktop.

Currently, LXQt is only available via source code. Eventually, however, rolling releases (such as Arch Linux, Gentoo, and openSUSE Tumbleweed) will pick up the latest version in their respective repos.

You can read more about this new release in the official LXQt announcement (<https://lxqt-project.org/release/2023/04/15/release-lxqt-1-3-0/>).

## 4MLinux 42.0 Now Ready for Prime Time

---

4MLinux is a distribution aimed at systems with fewer resources that can run on machines with as little as 128 MB of RAM, turning them into viable workstations or servers. 4MLinux also can be used as a rescue disc to recover data from a malfunctioning system.

The latest stable version is 42.0 and includes software such as LibreOffice 7.5.2, AbiWord 3.0.5, GIMP 2.10.34, Firefox 111.0, Chromium 106.0.5249, Thunderbird 102.8.0, Audacious 4.3, VLC 3.0.18, Mesa 22.2.3, Wine 8.3, ALSAPlayer, Baka MPlayer, Gnome MPlayer, Gnome MPV, mp3blaster, and XMS.

You’ll also find updated toolchains, including PHP (versions 5.6.40, 7.4.33, and 8.1.17), Perl 5.36.0, Python (versions 2.7.18 and 3.10.8), and Ruby 3.1.3. On the server side, you’ll find Apache 2.4.56 and MariaDB 10.6.12. The kernel shipped with 4MLinux 42.0 is version 6.1.10. You can view the entire package list included with 4MLinux (<http://4mlinux.com/addons-42.0.txt>).

4MLinux isn’t just ideal for reviving old hardware, because it also includes support for UEFI-enabled machines. With this distribution, you can create a desktop multimedia player, a productivity machine, and a mini server.

Download an ISO of 4MLinux (<http://4mlinux.com/index.php?page=download>) now and bring new life to old hardware or use a new machine to create a lightning-fast desktop.



## Prying the lid off of government data

# Open Up

The open data movement extends the ideals of open source to government data and, ultimately, all the world's knowledge.

By Joe Casad

For as long as governments have kept records, citizens have debated over access to those records. When the ancient Mesopotamians started recording data on clay tablets, I have no doubt that lively discussions erupted over who should or should not get to see those tablets. Information, after all, is power, and data is the first step on the path of information.

But data has a funny way of becoming more powerful with more eyes looking at it. Enlightened government eventually came to understand that the whole society could benefit from giving access to outside viewers who might find insights. The rise of democratic governments, where the people (at least in theory) set the rules, and the concept of press freedom brought more attention to the need for public access, leading to the development of massive national archives, as well as university libraries, presidential libraries, and other portals that led to data acquired through government funding.

Then came the information age, and the volume of data, as well as the tools for accessing it, became so vast and complicated that the system quite lost its moorings. Back when data was primarily stored and communicated through paper (or clay tablets), the “format” as we use the term today was much less critical. When government data came to be stored electronically, format became a point of control, and we are still trying to work out what that means and what to do about it.

Governments and research institutions funded by governments were early users of computers. In those early days, development teams tended to work independently – often without a long-term view of how to prepare data for future access. Many government agencies didn't have the staff or budget to do their own programming and, instead, out-sourced it to contractors. The contractors were well aware of the power of their position, and some became adept at positioning themselves as gatekeepers to the data. The government would commission a study, and the contractor or grant recipient would provide a final report. But the status of the data after

that point was often ill-defined. In theory, it was government data, but the knowledge and tools for how to access it belonged to the contractor, and the only way to gain additional insights and value from the data was to pay the contractor for another study.

It took several years for elected officials to understand the full scope of this problem and to start to address it. If you think of source code as “data” describing how a program works, the Free Software movement was an early effort at addressing the problems of secrecy and too many gatekeepers in the software development industry. And certainly, the ideals of Free Software and Open Source software have had a strong influence on recent open data initiatives.

When the smoke cleared from the first computer revolution, officials gradually began to see that public interest was not well served by throwing government money into data that was stored in closed formats. In 2003, the European Union approved the Directive on the Re-Use of Public Sector Information [1], which came to be known as the PSI Directive. The PSI Directive laid out a legislative framework for member states to establish policies that promote data re-use, stating that public information should be “open by design and by default.” The PSI Directive has been amended twice, most recently in 2019, and

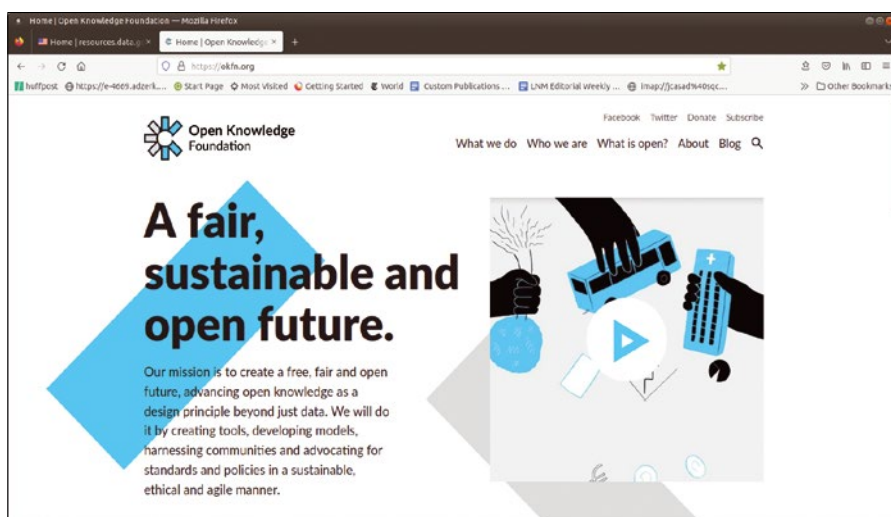


Figure 1: The Open Knowledge Foundation was an early advocate for open data and creator of the Open Definition.



it is now called the Open Data Directive [2]. The PSI and Open Data directives stake out the position that keeping data in a reusable state has value, regardless of whether a predefined plan exists for how and when it will be reused.

In the UK, the Open Knowledge Foundation (OKF) [3] launched in 2004 to provide services, tools, training, and community campaigns in support of open knowledge (Figure 1). One of the first accomplishments of the OKF was the Open Definition, a proposed reference for what it means to be “open.” The full Open Definition is available online with later amendments [4]. The OKF summarizes the definition as follows, “Open means anyone can freely access, use, modify, and share for any purpose (subject, at most, to requirements that preserve provenance and openness).”

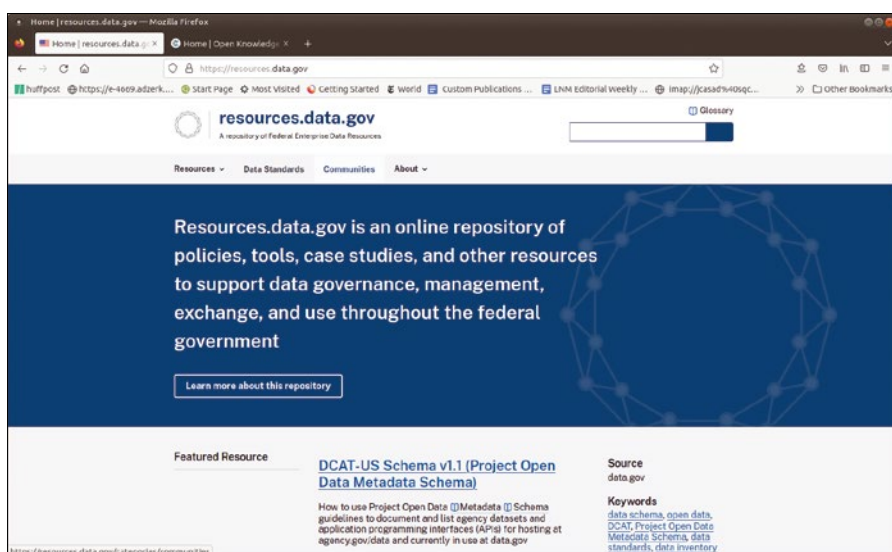
Similar forces were at play across the ocean, although in the US, with its long history of skepticism about government, it took a little longer to raise awareness. In 2007, a group of 30 visionaries met in Sebastopol, California, for a summit that would lead to the launch of the Open Government Data initiative [5]. Unsurprisingly, many in the group were familiar figures in FOSS circles, including Tim O’Reilly, founder of O’Reilly Media, and Stanford Law Professor Lawrence Lessig, founder of the Creative Commons license collection. One of the youngest members of the group was the late Aaron Schwartz, a FOSS and free data activist and creator of RSS. The Sebastopol group established eight principles for defining when government data is “open”:

- complete – all public data “that is not subject to valid privacy, security, or privilege limitations” is made available, without missing pieces that could limit its usefulness or obscure its meaning.
- primary – data is saved as collected at the source, not aggregate or processed into a modified form.
- timely – data is made available as quickly possible to preserve the potential value.

- accessible – data is available on the Internet and stored in a form that promotes the widest possible range of uses.
- machine processable – data should be “reasonably structured” to allow automated processing.
- non-discriminatory – anyone can access the data, with no requirement for registration.
- non-proprietary – data should be stored in a format “over which no entity has exclusive control.”
- license-free – data is not subject to any “copyright, patent, trademark, or trade secret regulation.”

The Open Definition of the OKF and the eight principles of the Open Government Data initiative were important for establishing a clear vision for what the term “open data” meant.

In 2013, President Barack Obama signed an executive order making open and machine-readable data the default for government data sets [6]. One of the initiatives launched during the Obama administration was Project Open Data [7], a collection of tools and resources for promoting open data. Project Open Data was replaced by the online repository resources.data.gov (Figure 2) with the passage of the Open Government Data act in 2019. As of this writing, the US



**Figure 2:** resources.data.gov is the US government’s repository for open data tools and resources.

government has posted more than 255,000 open datasets to the data.gov website [8].

## The Details: What Is It Really

Once you get organized, it doesn't really cost more money to pursue open data policies (and it might cost less in the long run). In another sense, it does take a certain amount of energy and diligence to ensure that things are done in an open way. Government directives and executive orders are useful for ensuring that project managers and bureaucrats stay focused on the goal. Down at the level of implementation, the open data movement depends primarily on three things:

- open formats
- specifications and contract language for requiring open data practices
- tools for supporting and maintaining open data

A number of tools exist for facilitating open data. Many of these tools are focused on bringing existing data into a form that is easy to publish on the web. The tools range from full-scale data management systems to special-purpose data transformation utilities. Project Open Data, for instance, included tools for converting a database or a CSV file into a RESTful API or directly into HTML (Figure 3). These kinds of tools make it very easy to take data stored in an arbitrary form of the past and make it available as open data through a web portal. CKAN, which you'll learn more about later in this issue, is an open source data management tool created by the OKF that is used throughout the world as a tool for publishing and aggregating open data.

## Where It Stands

Governments have made much progress on open data in the past 20 years, but the quest continues. The EU Open Data Directive, for instance, is quite promising, but the Directive itself does not really have the force of law and is, instead, a directive to the member states to pass their own laws facilitating open data. According to Wikipedia, almost half the EU member states were still out of compliance with the Open Data Directive as of July 2022.

In the US, executive orders are not exactly like laws either and are subject to changes and reinterpretation by future administrations. A law like the Open Government Data Act has more teeth than an executive order, however, the law only applies at the federal level. States, localities, and research projects that do not use federal dollars can set their own rules. Different states have varying levels of support for open data. It is worth noting, however, that even in today's hyper-partisan political culture, Open Government Data does have

bi-partisan support. The Open Government Data Act had four sponsors – two Republicans and two Democrats.

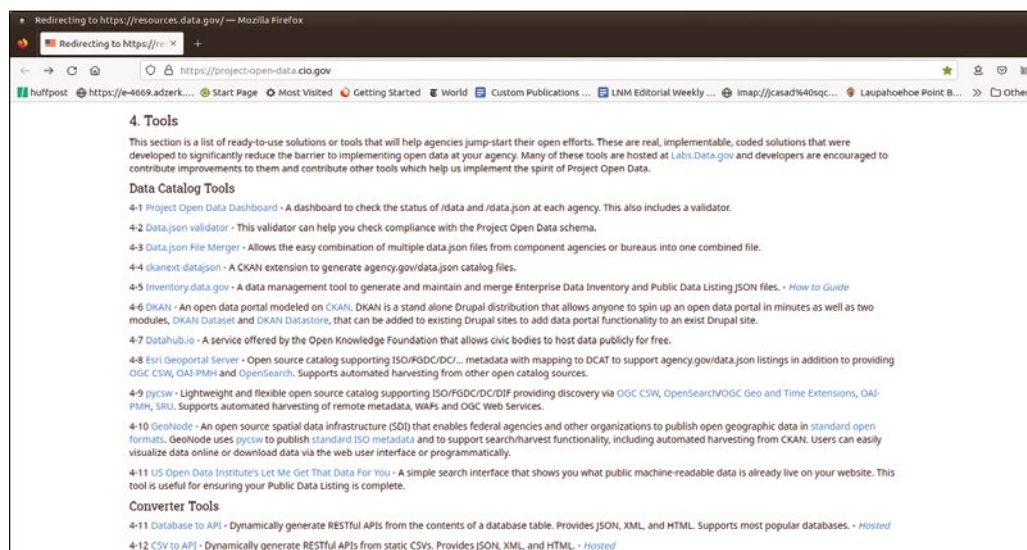
## Conclusion

Open data is a big topic that touches on laws, privacy matters, and politics, as well as programming, data storage, and web development topics. We can't cover all of it, but we'll give you a look at some things going on right now that are advancing the cause of open data. We'll introduce you to the CKAN document management system, and we'll talk to Lisa Allen of the Open Data Institute.

The topic of open data, as it is commonly used today, applies to government data, but we are all well aware that control of other forms of data is also a problem. I don't think anyone would want private user data to be "open" in the sense that anyone could access it, however, many users at least want access to *their own* data in order to migrate it to other platforms. The need for more control over personal social media data has led to the rise of open source platforms such as Mastodon and Diaspora. However, big tech vendors like Google and Microsoft have also heard the call for easier migration. Also in this month's issue, we take a look at the Data Transfer Project, an effort to arrive at an open form for migrating closed source social media data. ■■■

## Info

- [1] Directive on the Re-Use of Public Sector Information: [https://en.wikipedia.org/wiki/Directive\\_on\\_the\\_re-use\\_of\\_public\\_sector\\_information](https://en.wikipedia.org/wiki/Directive_on_the_re-use_of_public_sector_information)
- [2] Open Data Directive: <https://data.gov.ie/pages/open-data-directive>
- [3] Open Knowledge Foundation: <https://okfn.org/>
- [4] Open Definition: <http://opendefinition.org/od/2.1/en/>
- [5] Open Government Data: <https://opengovdata.org/>
- [6] Obama's Executive Order: <https://obamawhitehouse.archives.gov/the-press-office/2013/05/09/executive-order-making-open-and-machine-readable-new-default-government>
- [7] Project Open Data: <https://obamawhitehouse.archives.gov/blog/2013/05/16/introducing-project-open-data>
- [8] data.gov Website: <https://data.gov/>



**Figure 3:** Project Open Data developed a large collection of tools for cataloging and converting data. Many of these tools have been replaced by newer tools at resources.data.gov.

PRODUCED BY  TILDE



2 0 2 3  
**Rust**  
**CONF**

Meet and learn from hundreds of  
the world's top Rust developers.  
**Grab your tickets at [rustconf.com](https://rustconf.com)**



**September 12-15, 2023**  
Albuquerque, NM and Live Online

# New Address

The Data Transfer Project wants to make it easier to move your data between social media sites. *By Nate Drake*

**D**ata portability and transparency are ongoing issues that plague all major social media giants. Who owns the data you post to your social media accounts? Can you get a copy of the data if you ask for it? If you had a copy, what could you do with it?

Many leading social media companies have APIs that let you extract and upload data, but the data formats tend to be dissimilar and proprietary, which means if you obtained the data, you couldn't do much with it unless you are a programmer yourself and have plenty of time for personal coding.

Back in 2018, a few leading social media companies pledged to make an effort at addressing this problem. The result is the Data Transfer Project, which was recently rebranded and expanded as the Data Transfer Initiative [1]. The mission of the Data Transfer Project (and Initiative) is to support a common neutral format for social media data as it passes from one platform to another, as well as to provide the tools necessary for transforming data in and out of that format.

In the long term, the goal is for the user to be free from direct involvement in the migration. A user who wishes to move data from one platform (say Twitter) to another platform (say Facebook) will simply choose an option in the Facebook user interface, and the migration will happen automatically. According to the Data Transfer Project developers, the purpose of the project is to “Extend data portability beyond a user’s ability to download a copy of their data from their service provider (“provider”), to providing the user the ability to initiate direct transfer of their data into and out of any participating provider” [2].

The project was originally started by Google, Apple, Meta, Microsoft, Twitter, and SmugMug (Figure 1), but other companies are invited to join in. The Data Transfer Project is still



a work in progress, but some of the code is available on GitHub [3] (Figure 2), and the developers provide API keys for the participating services for those who are interested in testing [4].

## How It Works

According to Meta’s Engineering Blog [5], The Data Transfer Project consists of three main components:

- a set of shared data models to represent each vertical (i.e., photos, contacts, playlists),
- adapters, which handle the authentication of a user to a service (normally OAuth) and the transformation of data to and from the shared data models (importers and exporters),
- and a task management framework, which puts all the pieces together and handles the life cycle of a transfer job, including job creation and running the transfer.

The data models provide the neutral format needed for transferring data to or from the participating platforms. One of the guiding principles of the project is that the vendors should not have to rewrite their APIs. Instead, the vendor provides an *adapter* to transform the data from the platform’s own format to the neutral format – and also to transform the neutral data to a format needed for data import. The adapters basically serve as extensions of the API. Without something like DTP, a vendor would have to create a different solution for migration to every different platform. With DTP, the vendor just has to write an adapter for



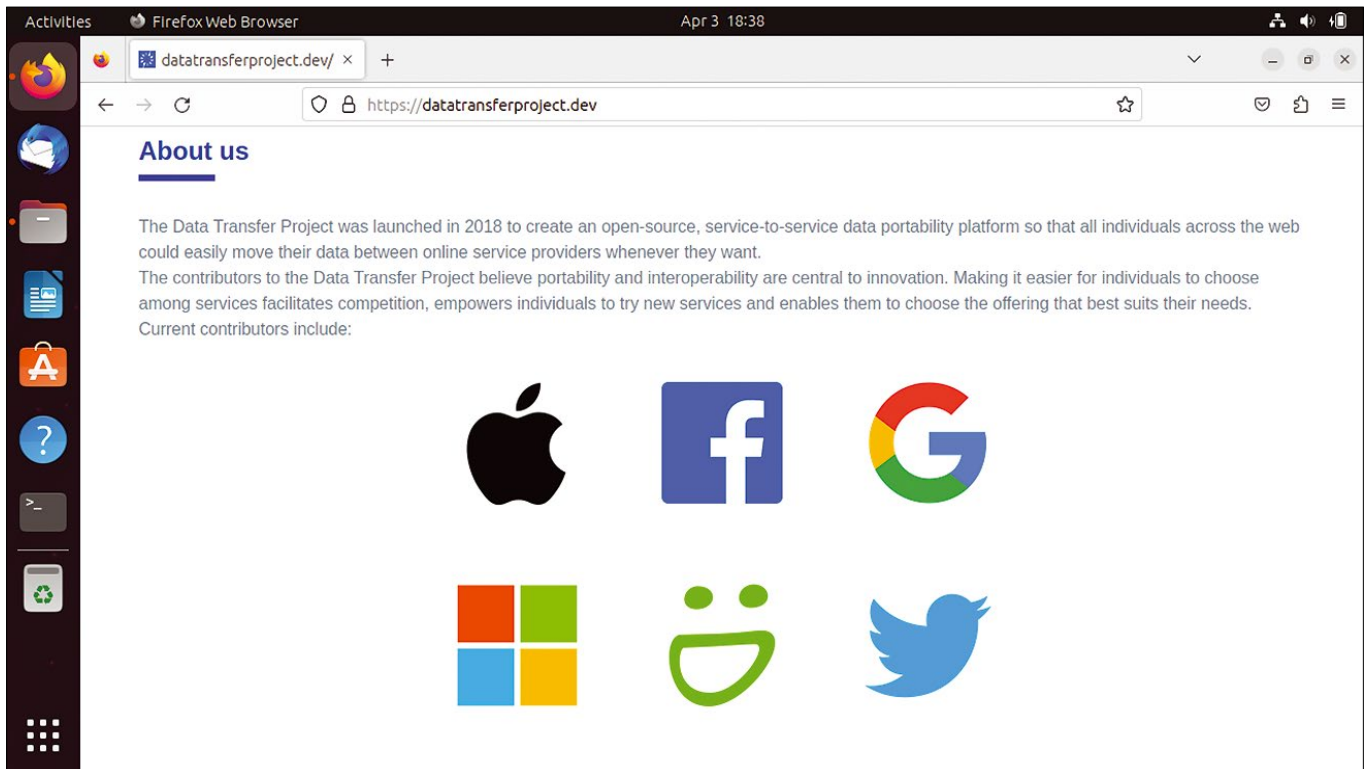
importing data from the neutral format and one to export code into the neutral format.

Of course, you can't migrate the data unless you have access to it. In addition to the data adapters are *authentication adapters* that allow the requesting service to access the originating service. According to the DTP documentation, "OAuth is likely

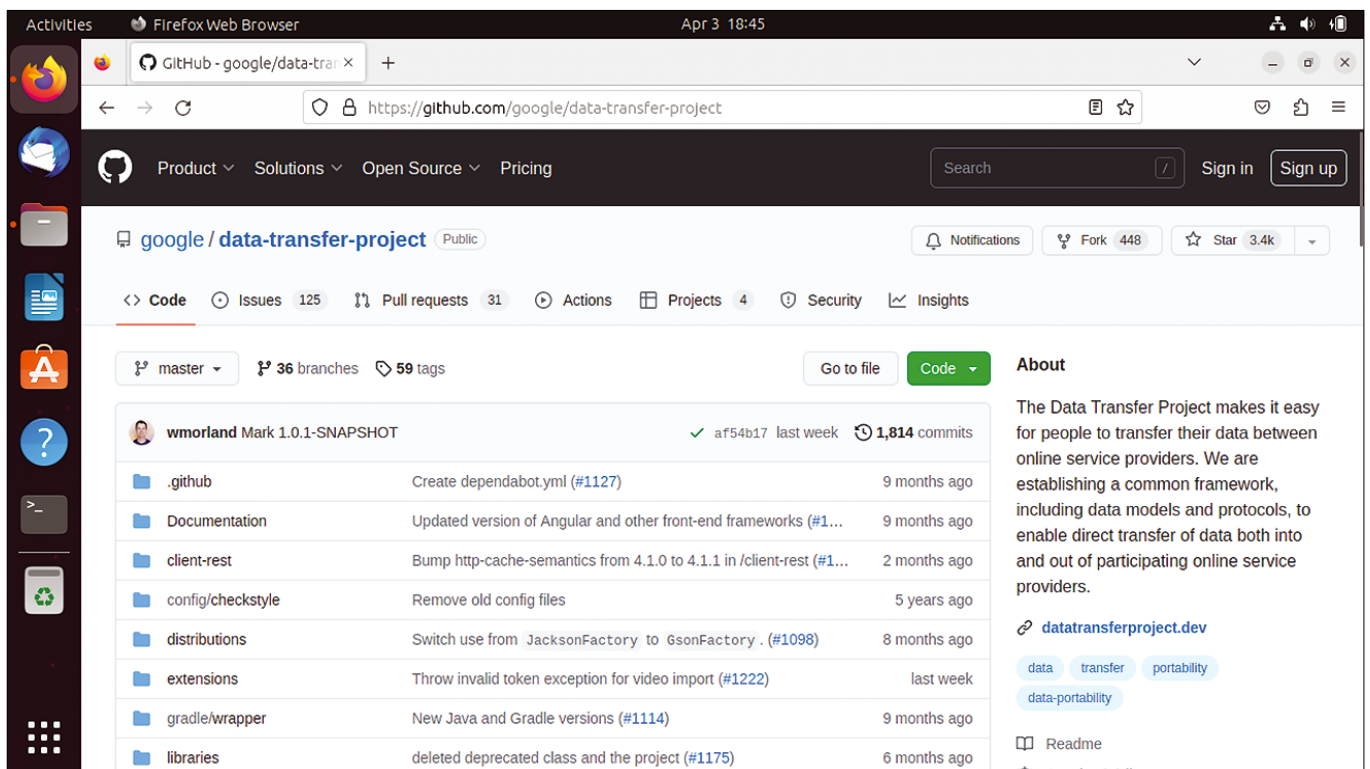
to be the choice for most providers, however the DTP is agnostic to the type of authentication."

## Why and Why Now?

If you've ever tried to download your information from major providers like Facebook, you'll know that they already



**Figure 1:** There are some big names behind DTP, though most of the coding is done by Google.



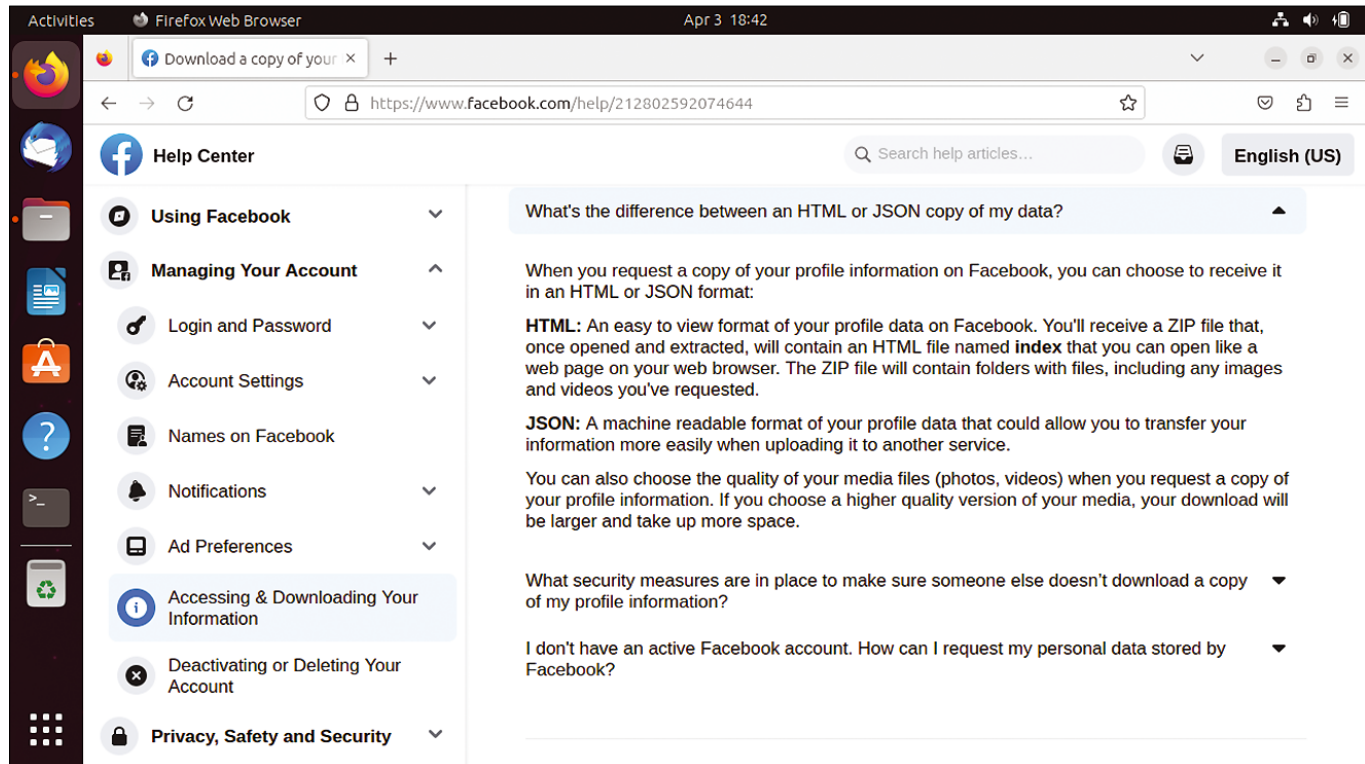
**Figure 2:** An impressive array of DTP open source tools are available, but it is not clear how exactly they're used on each platform.

have an online tool to download your account data, so you might wonder why the DTP is necessary.

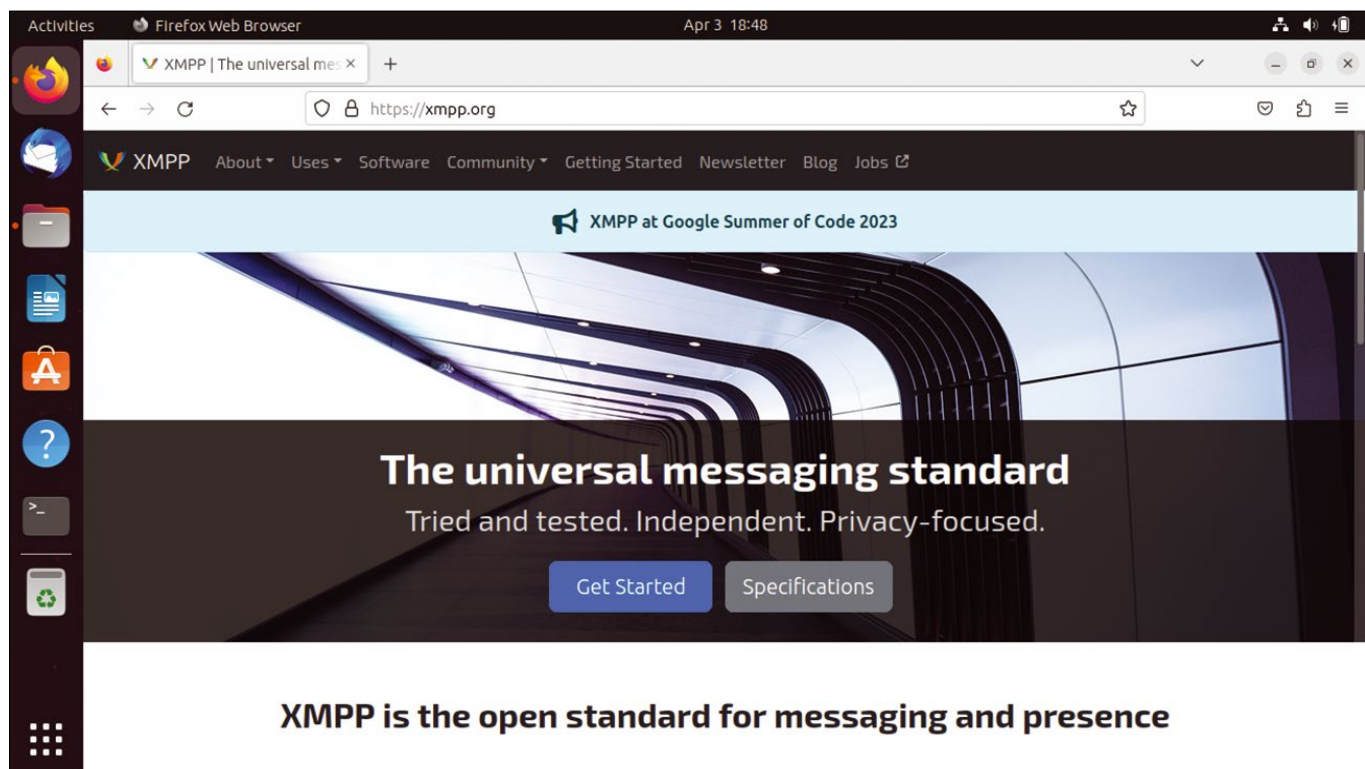
The reason is that the data you download isn't specifically designed to be interoperable with other services. Facebook has rectified this to some extent by allowing you to download your

account data in JSON rather than HTML format to make it easier to reupload to another service [6] but stills caution that the files are for your personal use (Figure 3).

New regulations like GDPR (General Data Protection Regulation) in Europe also require companies to provide all data they



**Figure 3:** Some websites like Facebook allow you to download your account information in JSON format for easier reupload, but these files don't necessarily hold all your data.



**Figure 4:** XMPP (formerly Jabber) is an open messaging standard that tech giants could use to enable messaging between platforms.

hold on their customers. In the case of websites like Facebook, the data download doesn't necessarily include information like location data, facial recognition, links to friend's profiles, and so on.

As Congressman David Cicilline pointed out around the time of the Cambridge Analytica scandal, smaller companies will only become competitive with Facebook if it's easy for existing users to transfer all their data elsewhere, such as html links to friends' profiles [7].

## Open Data and Open Source

In their announcements, both Facebook and Google play up the open source nature of the data transfer tools. There are public, open source extensions that allow the Data Transfer Project to be run on the Google Cloud Platform and Microsoft Azure. But the overall goal of the project is to support open data – not necessarily open source. An open source tool that supports migration to a proprietary format or a closed source service running in the cloud is not an ideal case study in free software. Because the software running on these platforms is proprietary, there's no way to be certain every part of your user data has been copied to your destination platform in a secure way.

Kevin Bankston, Director of the Open Technology Institute, has urged tech companies to go further [8]. Although he is optimistic about companies like Twitter and Facebook who offer downloads of account data in JSON format, he pointed out in 2018 that "Social networks should consider using the Activity Streams 2.0 open standard [9], a particular JSON-based format for exporting social media items. Facebook helped develop the standard at the World Wide Web Consortium, but right now only decentralized social network tools like Mastodon use it." (Since the time of writing Twitter has also adopted the Activity Streams format) [10].

Bankston mentions that tech giants could segue around the data transfer issue by making their platforms more interoperable. He points out that Meta's Developer Policy [11], e.g., makes it extremely difficult to create an app that makes full use of the API and replicates Facebook's core functionality like instant messaging.

Facebook also famously dropped support for the open XMPP standard for messaging in 2015 in favor of their own proprietary standards [12]. If they and other platforms were to agree to adopt XMPP (Figure 4), this would allow a Facebook user to view contacts and message and video-call users of other services like Skype or Google Chat without moving their data.

## The Data Dilemma

Openness, data transparency, and portability are very laudable goals, and we can only welcome attempts to allow users greater access and control over their data. Still, adding open data extensions to proprietary platforms like Facebook and Twitter is a far cry from implementing a fully open source, decentralized social networking platform like Diaspora or Mastodon. The Data Transfer Project is an incremental step in breaking down the walls of Big Tech, but no one should declare victory yet. ■■■

### Info

- [1] Data Transfer Initiative/Project: <https://dtinit.org/>
- [2] Data Transfer Project Overview and Fundamentals: <https://datatransferproject.dev/dtp-overview.pdf>
- [3] Data Transfer Project on Google's GitHub: <https://github.com/google/data-transfer-project>
- [4] Data Transfer Project Documentation: <https://dtinit.org/documentation>
- [5] Data Transfer Project – Enabling portability of photos and videos between services: <https://engineering.fb.com/2019/12/02/security/data-transfer-project/>
- [6] Download a Copy of Your Information on Facebook: <https://www.facebook.com/help/212802592074644>
- [7] Competition Is at the Heart of Facebook's Privacy Problem: <https://www.wired.com/story/competition-is-at-the-heart-of-facebooks-privacy-problem/>
- [8] "How We Can 'Free' Our Facebook Friends": <https://www.techdirt.com/user/kevin-bankston/>
- [9] Activity Streams 2.0: <https://www.w3.org/TR/activitystreams-core/#introduction>
- [10] Data Dictionary Activity Object: <https://developer.twitter.com/en/docs/twitter-api/enterprise/data-dictionary/activity-streams-objects/activity>
- [11] Developer Policies: <https://developers.facebook.com/devpolicy>
- [12] Facebook Chat will stop working in Ubuntu this week: <https://www.omgubuntu.co.uk/2015/04/facebook-chat-api-empathy-pidgin-stop-working>

### Author

**Nate Drake** is a tech journalist specializing in cybersecurity and retro tech.



## Managing open data with CKAN

# Fast Track

CKAN, a versatile data management system, lets you build a portal to share your open data. *By Marco Fioretti*

Open data would be of little use if it wasn't easy to automatically publish and update and subsequently find, download, and integrate with other open data, straight from the primary sources.

The Comprehensive Knowledge Archive Network (CKAN) [1], a key component of the Open Knowledge Foundation's mission to promote open data [2], is a collection of tools for implementing open data hubs and portals. If you have a dataset that you'd like to make available as open data, CKAN gives you a head start on implementing the necessary infrastructure.

The CKAN tools help you manage and federate the data. It also comes with advanced geospatial features, search capabilities, metadata, and visualization options. Using CKAN tools and best practices, you can build a portal for your open data with the least possible effort.

### How CKAN Works

CKAN is based on open standards and free/open source software. At its core, most of CKAN is Python code, released under an aGPL license. Other main components are the PostgreSQL database engine and the Apache Solr search server [3].

CKAN websites make it easy to find, browse, and analyze open data manually. CKAN's real value, however, is its consistent support for creating federated catalogs of open data and, importantly, finding and reusing the same data automatically.

CKAN does this through application programming interfaces (APIs) [4] that support REST-style data exchange and announce and export entire datasets according to the Data Catalogue Vocabulary Application Profile (DCAT-AP) [5]. DCAT-AP is an open standard for describing public sector datasets,

developed in Europe specifically to make those datasets easier to search. DCAT-AP rules standardize the description of datasets, defining which metadata should be published and the corresponding machine-readable formats.

For this reason, many organizations (mainly, but not exclusively, public administrations) use CKAN to collaboratively build a worldwide, coherent catalog of open data. CKAN's tools and methods completely automate not just the initial online publication of open data with all its metadata, but also its constant updating, straight from the original sources, whether from databases or spreadsheet files.

When loading one or more sets of open data onto a CKAN website, its administrator may enable its end users (both human and other software) to immediately download current dataset snapshots in JSON format or browse and visualize the raw data directly in several ways.

Automatic usage is possible thanks to other CKAN functions that, once a dataset is loaded, announce its existence with Resource Description Framework (RDF) declarations. Reading the RDF declarations, other portals may then expose the corresponding datasets, as well as datasets of many other CKAN-compatible sources, as part of one single catalog of their choosing. Even better, that catalog will always automatically contain complete, current versions of all the datasets it lists.

In Italy, for example, some regional administrations' CKAN catalogs automatically feed into the national data portal [6], which then automatically feeds into the European portal [7]. Worth noting here, the national data portal is not a CKAN portal, but it can still federate successfully by supporting and using the DCAT-AP standards and software tools. That's the beauty of open standards.

**Figure 1:** With CKAN, each dataset gets its own web page, containing all the related information, as well as several ways to access the data.

**Figure 2:** A tabular view of the restaurants, transit services, and other tourist services, which users may sort and search in several ways.

**Figure 3:** If properly labeled, CKAN can automatically show the location of each entry. Here, CKAN doesn't recognize the *Latitudine* and *Longitudine* labels, so the map is blank.

For this to work in practice, two conditions must be met. First, a website must announce that it's running CKAN (or at least following the compatible interfaces and standards) and be configured to make it discoverable by other CKAN (or CKAN-compatible) websites (which I am told is not always the case). Second, each dataset's metadata must always exactly correspond to its actual scope, freshness, and content. Otherwise, the whole thing either breaks or becomes practically useless. While this synchronization requires some effort, CKAN remains the fastest path to standardized, collaborative publishing of open data.

## CKAN in Action!

To demonstrate CKAN's usefulness, I interviewed two of the main Italian CKAN experts: Maurizio Napolitano [8] and Vincenzo Patruno [9]. Napolitano works for Fondazione Bruno Kessler (FBK, Bruno Kessler Foundation) in Trento, Italy, as coordinator of the FBK Digital Commons Lab. Patruno is the IT/data manager for the Italian National Institute of Statistics and the vice president of onData APS [10] (a nonprofit association that promotes open data). Both Napolitano and Patruno got involved with open data because they believe "the most powerful form of revolution we have today [is] to give more opportunities to everybody with good ideas to improve our society."

## Trento Tourist Spots

Napolitano provides the first example of CKAN's power with a dataset that lists the tourist points of interest in the Valle dei Laghi (Valley of Lakes) near Trento in Northern Italy. To access this dataset (shown in Figure 1), you can use the federated OPENdata Trentino portal to search for "Valle dei Laghi" [11].

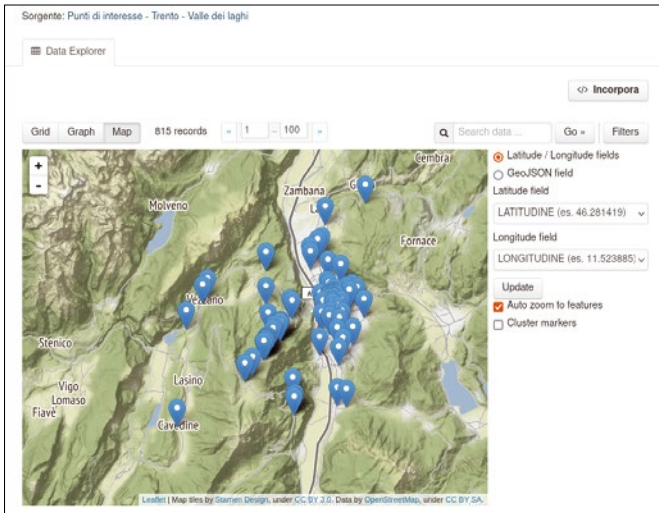
On the right, each *Esplora* (explore) drop-down menu has two options: *Scaricare* (download the raw dataset) and *Anteprima* (preview). Choosing *Anteprima* opens the interface shown in Figure 2.

The default *Grid* view lists the several entries in columns, allowing users to filter or sort them as needed. Clicking on the *Map* tab shows each entry on the map. However, in Figure 3, the map is empty. The reason for this is because the latitude and longitude columns have not been labeled correctly in the

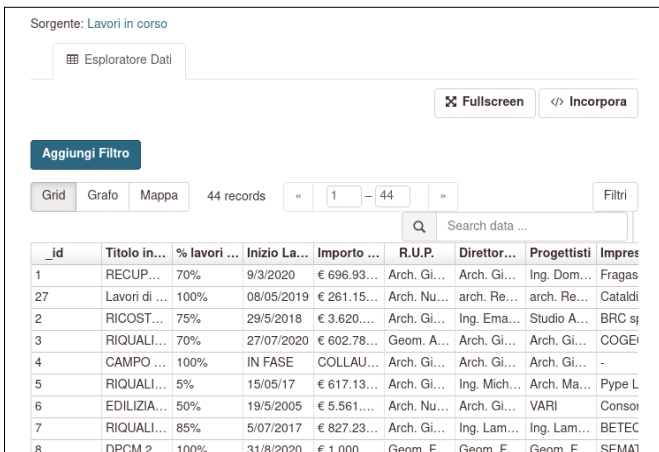
**Figure 4:** When coordinates are not labeled as CKAN expects, you can still help CKAN recognize them.

*Grid* view in Figure 2. CKAN expects these columns to be named *Latitude* and *Longitude* to recognize and use the geographic coordinates, but Figure 2 uses the Italian column names *Latitudine* and *Longitudine*. To get around this problem, you can tell CKAN explicitly where to find those values as shown in Figure 4. Once you have

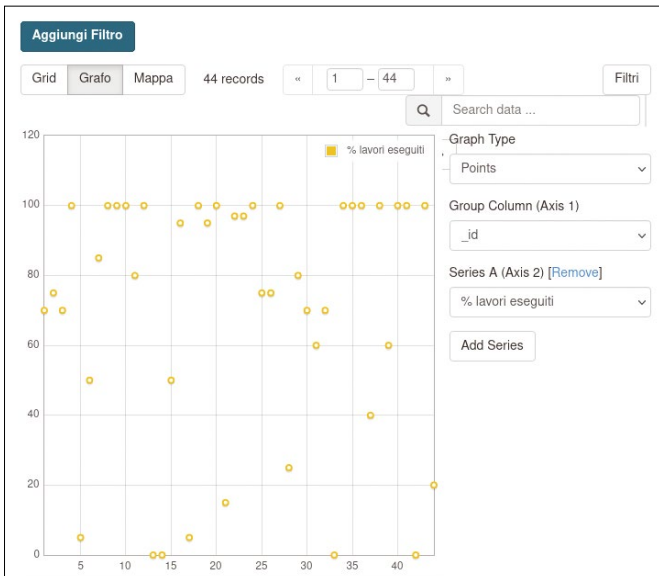
assigned the latitude and longitude fields, all the entries appear on the map (Figure 5).



**Figure 5:** After assigning the correct fields for latitude and longitude, CKAN now displays a map with the points of interest.



**Figure 6:** A detailed list of public works in Matera.

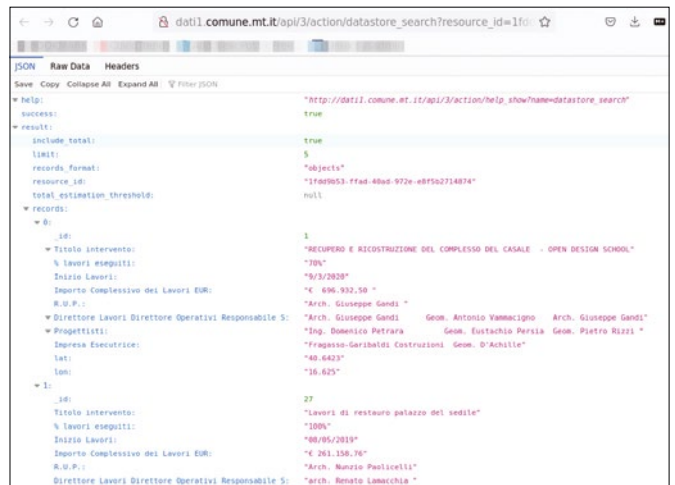


**Figure 7:** Besides mapping, CKAN supports data visualization, much like ordinary spreadsheets.

### Public Works in Matera

The second example of CKAN's power, courtesy of Patruno, comes from the ancient city of Matera, filming location for *The Passion of the Christ* and the 2019 European Capital of Culture. Because a place with so much history requires more maintenance than most, Patruno and others prepared a CKAN demo [12] that, among other things, contains the database of all the public works in progress (*Lavori in corso*), shown in Figure 6.

Despite the difference in location and data, Matera's interface looks the same as Trento's interface (Figure 2), which is exactly the point of CKAN. Localization may be different (e.g., *Data Explorer* in the Trento version versus the Italian



**Figure 8:** Clicking on the **Download** button, you can get the entire dataset, either raw or in JSON format, or just its headers.

### CKAN Data API

Accesso alle informazioni di risorsa via web utilizzando un'ambiente API o informazioni in the [main CKAN Data API and DataStore documentation](#).

#### Endpoints >

L'interfaccia Data API può essere consultata attraverso le azioni seguenti tra que

- Crea** [http://datil.comune.mt.it/api/3/action/dastore\\_create](http://datil.comune.mt.it/api/3/action/dastore_create)
- Aggiorna / Inserisci** [http://datil.comune.mt.it/api/3/action/dastore\\_upsert](http://datil.comune.mt.it/api/3/action/dastore_upsert)
- Query** [http://datil.comune.mt.it/api/3/action/dastore\\_search](http://datil.comune.mt.it/api/3/action/dastore_search)
- Query (via SQL)** [http://datil.comune.mt.it/api/3/action/dastore\\_search\\_sql](http://datil.comune.mt.it/api/3/action/dastore_search_sql)

#### Querying >

##### Esempio di query (primi 5 risultati)

[http://datil.comune.mt.it/api/3/action/dastore\\_search?resource\\_id=1fdd9b53-ffad-40ad-972e-](http://datil.comune.mt.it/api/3/action/dastore_search?resource_id=1fdd9b53-ffad-40ad-972e-)

##### Esempio di query (risultati che includono 'jones')

[http://datil.comune.mt.it/api/3/action/dastore\\_search?resource\\_id=1fdd9b53-ffad-40ad-972e-](http://datil.comune.mt.it/api/3/action/dastore_search?resource_id=1fdd9b53-ffad-40ad-972e-)

##### Esempio di query (via SQL statement)

[http://datil.comune.mt.it/api/3/action/dastore\\_search\\_sql?sql=SELECT \\* from '1fdd9b53-ffad-40ad-972e-' 'jones'](http://datil.comune.mt.it/api/3/action/dastore_search_sql?sql=SELECT * from '1fdd9b53-ffad-40ad-972e-' 'jones')

#### Esempio: Javascript >

Una richiesta ajax semplice (JSONP) verso l'API dati utilizzando jQuery.

```
var data = {
  resource_id: '1fdd9b53-ffad-40ad-972e-e8f5b2714874', // the resource id
  limit: 5, // get 5 results
  q: 'jones' // query for 'jones'
};
$.ajax({
  url: 'http://datil.comune.mt.it/api/3/action/dastore_search',
  data: data,
  dataType: 'jsonp',
  success: function(data) {
    alert('Total results found: ' + data.result.total)
  }
});
```

**Figure 9:** CKAN's Data API web pages list all the ways in which third-party software can access each dataset.

translation *Esploratore Dati* in Matera's interface), but a CKAN user in either city would have no problems using either portal.

Each entry in Figure 6 provides the project name, budget, managers, contract holder, and percentage of work completed for public works planned, ongoing, or completed in Matera.

Like the geographical coordinates used in Figure 5, CKAN lets you plot other numbers. The *Grafo* tab in Figure 6 (or the *Graph* tab in Figure 2) lets you quickly create all sorts of charts to visualize the raw data, just as you would in spreadsheet software. In Figure 7, for example, I plotted each project's work progress (when available).

You can also create a downloadable versions of the dataset in JSON format (Figure 8). For each dataset, CKAN also automatically creates a web page (Figure 9) listing all the access links (endpoints) to that dataset, as well database query samples in several programming languages. To generate the Data API web page, click on the *Data API* button (not shown in the figures).

## The Best for Last

I saved the coolest part of what Napolitano and Patruno shared for last. The *Incorpora* (embed) button in Figures 2

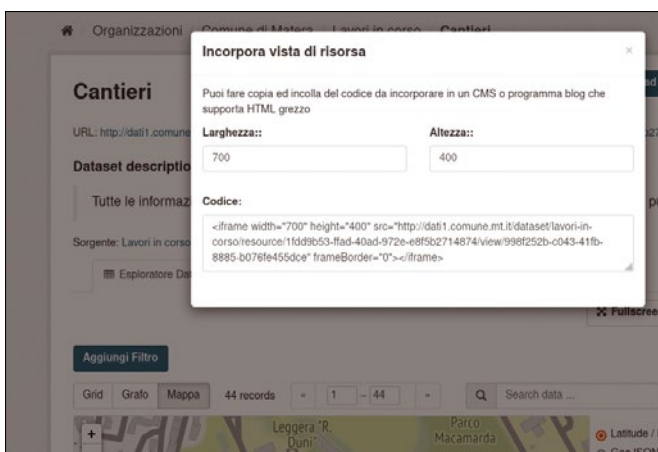
### Listing 1: Sample HTML Code

```
<html>
<body>
<h1>Hello Linux Magazine Readers!</h1>

<p>Here is a demo of how everybody may embed
<strong>dynamic</strong> CKAN data visualizations into any
Web page:</p>

<iframe width="700" height="400" src="http://dati1.comune.
mt.it/dataset/lavori-in-corso/resource/1fdd9b53-ffad-40ad-97
2e-e8f5b2714874/view/998f252b-c043-41fb-8885-b076fe455dce"
frameBorder="0"></iframe>

<p>All I had to do to get the same functionality of the
original portal, with <strong>clickable</strong> tabs for
graphs, maps, filters and so on, was to copy and paste here
the text I got from the "Incorpora" function!</p>
</body>
</html>
```

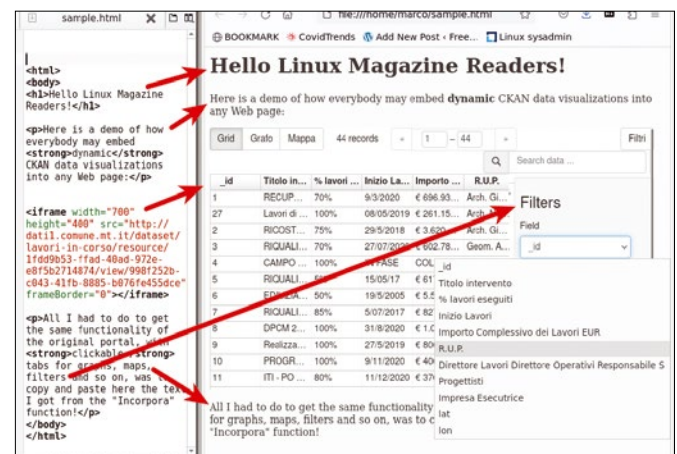


**Figure 10:** You can copy and paste the embed code shown here into any program that supports HTML.

and 6 lets another website offer all the services I just explained! Clicking on that button opens the panel in Figure 10, where you can define the size of an HTML iframe for the dataset and then copy the corresponding HTML code.

I wrote Listing 1 in less than three minutes, by pasting the code in Figure 10 into an empty file and wrapping it with the bare minimum of HTML markup, a title, and two short explanation paragraphs.

That's all I had to write to create the visually bare, but perfectly working dynamic web page shown in Figure 11. The web page in Figure 11 will let you sort or search the dataset, map its content, or make charts with it, just like on the original website. Openness is really cool, isn't it? ■■■



**Figure 11:** CKAN services are easy to embed in other websites. Just grab the code from the original server and paste it wherever you need it.

### Info

- [1] CKAN: <https://ckan.org/>
- [2] Open Knowledge Foundation: <https://okfn.org/>
- [3] Apache Solr: <https://solr.apache.org/>
- [4] REST APIs: <https://restfulapi.net/>
- [5] DCAT-AP: <https://joinup.ec.europa.eu/collection/semantic-interoperability-community-semic/solution/dcat-application-profile-data-portals-europe/release/11>
- [6] Italian national open data portal: <https://dati.gov.it> [In Italian]
- [7] European open data portal: <https://data.europa.eu/en>
- [8] Maurizio Napolitano: <https://twitter.com/napo>
- [9] Vincenzo Patruno: <https://twitter.com/vincpatruno>
- [10] onData APS: <https://www.ondata.it/> [In Italian]
- [11] OPENdata Trentino: <https://dati.trentino.it/> [In Italian]
- [12] Matero open data: <http://dati1.comune.mt.it> [In Italian]

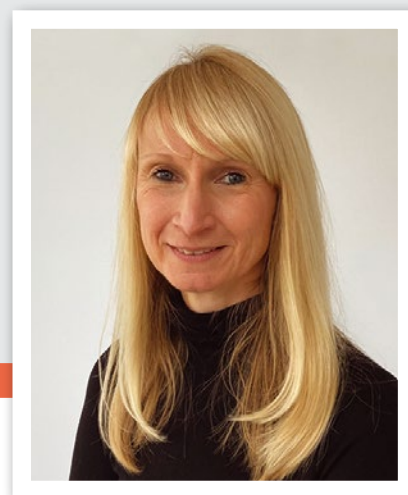
### Author

Marco Fioretti (<http://mfioretti.substack.com>) is a freelance author, trainer, and researcher based in Rome, Italy, who has been working with free/open source software since 1995 and on open digital standards since 2005. Marco also is a board member of the Free Knowledge Institute (<http://freeknowledge.eu>).



Meet Open Data Institute’s Lisa Allen

# Unlocking the Benefits



The Open Data Institute’s Lisa Allen explains why open data matters and what it will take for more widespread adoption. *By Joe Casad*

**W**orld Wide Web creator Tim Berners-Lee cofounded the Open Data Institute (ODI) to help foster a freer Internet. We asked Lisa Allen, Director of Data and Services for ODI, to talk about the group’s mission and the path ahead.

**Linux Magazine (LM):** How about if we start with a little history? What is the Open Data Institute? How did it get started and why?

**Lisa Allen (LA):** The Open Data Institute is a nonprofit company that was founded in 2012 (we celebrated our 10th anniversary last year!) by Sir Tim Berners-Lee and Sir Nigel Shadbolt. We operate as an institute and a data services provider, collaborating with businesses, governments, and civil society to create a world where data works for everyone.

We were originally established to champion the value of open data and to advocate for its innovative use in bringing about positive change. In the early days, we argued that social, economic, and environmental change would come about through the wide-scale adoption and understanding of open data. The global open banking movement began with a working group co-chaired by the ODI, and now over six million consumers in the UK alone benefit. We also incubated startup companies early on, companies that went on to create 1,000 jobs and generate £100 million (~\$125 million) in revenues. Today we work across the data spectrum, helping organizations share data across this spectrum.

**LM:** Your mission is so diverse. On one hand, you develop tools for open data. On the other hand, you have an advocacy role. Then you have this role that seems to be built around defining what open data is and developing guidelines for responsible data stewardship. How does all this fit together? How would you describe the priorities? Where does your organization spend most of its time?

**LA:** Our vision is to create a world where data works for everyone, and we aim to achieve this by building an open,

trustworthy data ecosystem through our work with businesses, governments, and civil society. For us we believe that data is a critical part of the national infrastructure, as important as the roads, railways, and the electricity networks on which we all rely. Therefore improving the data practices of organizations is important so they can build and manage effective data infrastructure. We do this by providing a broad range of services including training, expert consultancy, and applied research, to equip organizations with the ability to generate social and economic value. We also develop products and services, run a membership program, and work to inform, and build an understanding of, public policy development. While it is a broad range of services, they are vital components to building an open and trustworthy data ecosystem.

**LM:** The term “open data” gets so abstract. Are there specific formats or frameworks you recommend? Can you point to some specific practices you most want users to adopt?

**LA:** We define open data as data that anyone can access, use, and share. So rather than specific formats or frameworks, we believe in Sir Tim Berners-Lee’s 5 Star Linked Data system, so that data is available on the web, machine-readable, in a nonproprietary format, published using open standards, and links to other data. This is the journey that we all need to go through to unlock the value of data. Licensing is key and making use of open licenses such as Creative Commons can be transformational. For instance, introducing the Open Government License in the UK had a dramatic effect on the availability and usability of public data.

**LM:** Our readership of developers and Linux power users find your tools to be of particular interest. Tell us more about your Data Toolkit for Business and Data and Public Services Toolkit, as well as other tools you’re proud of.

**LA:** We produce many free tools and guides to help people manage, use, and publish data. The Data Toolkit for Business and the Data and Public Services Toolkit have much in





common – ultimately, they both aim to help people overcome barriers to using data effectively and bring together a range of tools to help in different settings. Two elements they have in common are the Data Ecosystem Mapping tool and the Data Ethics Canvas, and those are things we’re really proud of. We recently learned that both of these tools were incorporated into a toolkit created by the United Nations Development Programme and the GIZ (Germany’s international development office), who join the likes of Microsoft, OCHA (United Nations Office for the Coordination of Humanitarian Affairs), and Arup, alongside government departments, academic institutions, and health authorities in the UK and globally, in using them.

**LM:** It seems to be part of ODI’s DNA to evolve with the times. Recently, you’ve been raising awareness about open data issues surrounding artificial intelligence. How does AI relate to open data? What are your goals in the AI arena?

**LA:** It’s hard to miss all the attention that generative AI has received recently, and the headlines that have accompanied it. But the one thing the different technologies have in common is that they are all dependent on vast amounts of data. The technology will continue to evolve but data will remain an essential prerequisite. We like to think about what are you feeding your AI.

That’s why building open and trustworthy data ecosystems is so important. We need more openness and understanding about data sources, the biases contained within, and the impact this can have both negative and positive. We need greater data skills and knowledge to help people understand where AI is at work and for those working with AI.

Well-curated open data is vital here and so is data assurance. We define data assurance as the process, or set of processes, that increase confidence that data will meet a specific need, and that organizations collecting, accessing, using, and sharing data are doing so in trustworthy ways. Data assurance relates to several factors including its quality, supply, and ethics. When considering ethics, users need assurance that the data is appropriately collected, used, and shared. All of these factors

can affect the confidence that data is fit for purpose and will meet user requirements. Good, trusted data really matters, and open data is the best foundation for data infrastructure ensuring transparency, accountability, and understanding.

**LM:** What do you see as the biggest impediments to open data adoption? What is standing in the way right now?

**LA:** One of the biggest barriers is that people perceive sharing data, especially open data, as high risk. We think this is because people don’t necessarily understand the risks that are present in their data, and the risks associated with sharing it, and, importantly, how to mitigate those risks. Recently we have worked on this issue to develop a tool to help people to understand these issues and risks. We have identified five key areas to assess when sharing data: regulatory, security, ethical, reputational, and commercial risk. The biggest thing organizations can do is to understand their data, the risk and mitigation, and, most importantly, the benefits that open data can unlock.

**LM:** What are some specific examples of things that will get easier if open data is adopted on a massive scale?

**LA:** The impact of open data in banking and transport is undeniable, benefiting millions of people. Our own work with OpenActive (OA) working with the OA community has had a significant impact, making it easier for people to access physical activity opportunities through open data that were previously locked away. The benefits of these innovations extend far beyond what was initially anticipated. The Environment Agency’s open LIDAR data, for example, has had unexpected benefits for archaeology, for example, finding lost Roman roads. The potential benefits of open data are endless. This is unlocked through innovation, as others find creative and innovative ways to use the data; efficiencies, as the open data removes friction and unlocks value; and all the way to building trust by creating transparency. These are all benefits of open data – so the more that is released, then the more that can be unlocked. ■■■

# Hone your skills with special editions!

Get to know Shell, LibreOffice, Linux, and more from our Special Edition library.

The *Linux Magazine* team has created a series of single volumes that give you a deep-dive into the topics you want.

Available in print or digital format

**Check out the full library!**  
[sparkhaus-shop.com/specials](https://sparkhaus-shop.com/specials)



FREE DVD! JOIN THE LINUX REVOLUTION! ALL THE SOFTWARE YOU NEED!

# GETTING STARTED WITH LINUX

MORE POWERFUL • MORE SECURE • MORE FUN

LEARN HOW TO SET UP A LINUX SYSTEM TO:  
• Listen to Music • Play Games • Process Photos  
• Surf the Web • and Much More!

ARCHIVE DVD! RASPBERRY PI GEEK THE COMPLETE ARCHIVE 2,000 pages of maker projects and more! FREE DVD \$39.90 VALUE!

# MakerSpace

## HANDS-ON PROJECTS FOR MAKERS

BUILD A RASP PI RADIO!  
**MakerSpace #02**

## HANDS-ON PROJECTS FOR MAKERS

RASP PI TRICKS FEED YOUR FISH WHILE YOU'RE AWAY  
**MakerSpace #03**

## HANDS-ON PROJECTS FOR MAKERS

RETRO GAMING WITH FPGA

**Solar-Powered IoT**  
Avoid the battery dance with a sun-powered Pi Pico

**Irrigation Innovation**  
Water your plants without getting wet

**Xonish**  
Mix a little Bash in your Python scripts

TUNE YOUR LINUX SYSTEM  
2022 EDITION

# Cool Linux Hacks

84 HACKS

Tricks and shortcuts for Linux geeks

- Speed up downloads with Xtreme Download Manager
- Search text, PDF, and Office files with one tool
- Run VMs on a Proxmox server without installing

301 BEST BASH COMMANDS

# LINUX SHELL HANDBOOK

## SUPERCHARGE YOUR LINUX SKILLS

Power at Your Fingertips

- Pipe and redirect output
- Monitor processes
- Create custom scripts

Keep this guide as a permanent reference!

ALL NEW! INTERNET TOOLS TERMINAL UTILITIES FOR EMAIL AND GOOGLE SEARCH

FREE DVD! LibreOffice Full Version

Dive deep into the world's greatest free office suite

# LibreOffice Expert

Edit and Save MS Office Files

Write Your Own LO Macros  
Save time and automate common tasks

Create Professional:  
• Text Documents  
• Spreadsheets  
• Presentations  
• Databases

Replace MS Office and Google Docs!

LibreOffice® includes full versions for Windows, macOS, and Linux



One-stop solutions for Linux

# Coming of Age

Linux users can now experience what Windows and macOS users have enjoyed for decades: hardware, software, and services bundled together. We look at six of these one-stop solutions for Linux. *By Bruce Byfield*

**B**y traditional definition, distributions are software. Yet increasingly, companies are offering distributions as part of a bundle that includes hardware, support, and services – a kind of one-stop solution often called vertical integration in commercial jargon. What these solutions offer, though, can vary considerably.

**Author**

**Bruce Byfield** is a computer journalist and a freelance writer and editor specializing in free and open source software. In addition to his writing projects, he also teaches live and e-learning courses. In his spare time, Bruce writes about Northwest Coast art (<http://brucebyfield.wordpress.com>). He is also co-founder of Prentice Pieces, a blog about writing and fantasy at <https://prenticepieces.com/>.

The advantages of one-stop solutions are obvious. Vendors can offer software tailored to their hardware and win customer loyalty. For buyers, one-stop solutions offer convenience and ease of purchase, an end to the pre-purchase research that until recently has been standard in setting up a Linux computer.

Until recently, attempts at one-stop solutions have floundered. For instance, in 2013, KDE failed to launch the Spark tablet, with its unique variant of the Plasma desktop environment. Ubuntu did release the Touch tablet with its own desktop, but discontinued it in 2017. The reasons for these failures have never been thoroughly analyzed, but likely reasons are a lack of business experience in hardware and a relatively small market for commercial Linux.

Whatever the reason, one-stop solutions have only recently gained recognition. Many are derivatives of Ubuntu Long-Term Support (LTS) releases, with only minor tweaks visible on the desktop. Following the lead of large commercial vendors' Linux products, these one-stop solutions offer detailed options for each basic hardware model. Their distributions are usually distinguished by small administrative utilities, which are designed for bundled hardware and may not work with all distributions. Most offer coreboot [1], the free software BIOS and UEFI replacement, or plan to do so in the near future, which opens up the option of frequent and easy firmware updates. The distribution remains essential, of course, but the differences in one-stop

Photo by Harry Cunningham on Unsplash

solutions tend to lie in the rest of the bundle. Below are summaries of how six leading one-stop solutions differentiate themselves.

## TUXEDO Computers

TUXEDO [2], a rapidly expanding company just starting to become well-known in North America, offers an extensive product line. TUXEDO also does customization on request. One of the most security-conscious of the one-stop solutions, TUXEDO provides disk encryption and the ability to turn off webcams, microphones, radio technology, and the Intel Management Engine. As I write, TUXEDO is exploring the addition of coreboot. Its TUXEDO OS includes some well-designed admin utilities, which may not work on other distributions.

## Purism

Purism [3] immediately made headlines in 2014 when its distro, PureOS, became one of the few distributions to win the Free Software Foundation's Respects Your Freedom certificate for its security and privacy features. Manufactured in the United States, its computers include kill switches to physically disconnect the camera, microphone, wireless, and Bluetooth. Purism's computers can be run with the Librem key for security. In addition, when shipped to and from the company, unspecified anti-interdiction services are available. Both the hardware and distribution appear not to have been updated for at least a year. Whether this is because security features take longer to implement or because it is focusing on its Librem 5 phone is uncertain. Purism did not reply to a request for information.

## MX Linux

MX Linux [4] has had the most page views on DistroWatch since 2019. Its popularity partly may be due to the fact that is a collaboration between two popular distributions, antiX and the discontinued MEPIS. However, more importantly, MX Linux has created a

community version of a one-stop solution through partnerships with commercial companies. Cloud services are supplied through Shells and hardware through Star Labs Systems (see below). MX also offers thorough documentation, both embedded in its windows and on a separate web page.

## Star Labs Systems

Alone among the solutions listed here, Star Labs [5] does not have its own distribution, choosing instead to offer other user-friendly distributions: MX Linux, Ubuntu, Linux Mint, elementary OS, Manjaro, and Zorin. Star Labs emphasizes its hardware, including high-resolution screens, and a large trackpad with a built-in fingerprint reader. Its security features include a kill switch for peripherals and a camera that can be removed and stored in the hardware case. While Star Lab's website does not allow for easy browsing, it does have the option to display prices in dozens of currencies.

## Slimbook

Slimbook OS [6] uses Plasma by default – a combination that is not yet common – and includes the option for a tiled desktop. Alone among the choices summarized here, the hardware can include custom branding, an option that both the KDE and Manjaro distributions have exercised. Slimbook tested coreboot a few years ago, but discarded it due to unspecified problems.

## System76

System76 [7] manufactures all its workstations and keyboards in the United States, with plans to do the same for its laptops. While all of its laptops use coreboot, none of its workstations, minis, or servers do at this time. System76's in-house distro, Pop!\_OS, offers a user-friendly tiling desktop, but you can also choose Ubuntu as an alternative, although users may want download System76's packages for working with firmware. System76 provides lifelong person-to-person support and detailed, clear

documentation. An advocate of the right to repair, System76 provides detailed instructions on how to fix and upgrade its products.

## Vendor Lock-In?

With one-stop solutions, Linux users can finally enjoy what Windows and macOS users have enjoyed for decades. True, buyers may still have to research these one-stop solutions, because many are too recent to have much of a reputation, and solutions vary considerably. However, at least you no longer need to research each component's Linux compatibility.

Long-time Linux users might worry that one-stop solutions may also introduce vendor lock-in through the back door – something that many users turned to Linux to avoid. This possibility is not some malicious conspiracy, just businesses pursuing their normal efforts for an advantage. But is it likely to happen? The danger is always there, but probably unlikely. Already, enough one-stop solutions are available that avoiding one is easy enough if it becomes necessary.

Moreover, there is a simple way to judge the solution providers: Ask how they interact with the community. Do they participate on user forums? Are their products based on user feedback? Although they might focus on their own solutions, do they try to work with alternatives? In other words, are the providers good open source citizens, not just executives? As long as they are, then we can celebrate that Linux has finally come of age at last. ■■■

## Info

- [1] coreboot: <https://coreboot.org>
- [2] TUXEDO Computers: <https://www.tuxedocomputers.com>
- [3] Purism: <https://puri.sm>
- [4] MX Linux: <https://mxlinux.org>
- [5] Star Labs Systems: <https://us.starlabs.systems>
- [6] Slimbook: <https://slimbook.es/en/essential-en>
- [7] System76: <https://system76.com>

Creating custom ISO images

# DIY Imager

If you are looking to customize your Linux distribution, we show you three graphical front ends for creating bootable ISO images. *By Erik Bärwaldt*

**S**pecialized Linux distributions exist for virtually any imaginable use case. However, much like the typical all-rounders for daily use, these distributions often come with many superfluous applications. These apps consume disk space and – if they happen to run in the background – CPU resources as well. Some users want a lean basic system without additional software, which they can customize with the programs they actually need. This article looks at graphical front ends that give users a DIY Linux image quickly without too much overhead.

## Strategies

Customizable Linux distributions are usually based on a conventional ISO image. Ideally, the image will already contain a graphical user interface or offer a simple approach to installing a graphical desktop at the prompt. You also need an integrated package manager. A customizable system image often comes with a choice of multiple kernel versions. Standard applications like LibreOffice, Firefox, Gimp, or VLC may be missing, but they can be installed via the package manager if necessary. To be able to use a system like this on several

computers later, you need the ability to create ISO images of the system by deploying a handy tool, one that is easy to use and not just for Linux gurus.

## Approach

There are two approaches to generating a custom ISO image of a distribution. The first approach involves working with a live system. You first install and run the desired distribution on your computer and remove all unwanted applications. To this trunk system, you then add all the applications you actually need for your individual Linux system. Then, using a tool for generating ISO images gleaned from the running system, you create an image into which you bundle the entire running system, including the newly installed additional applications.

The second approach uses an existing ISO image obtained from the Internet (without prior installation) as the basis for a custom system. The tool generates the individual system from the standard image by loading it and temporarily unpacking it for editing. It then creates the custom image from the image you modified.

Depending on whether you want your custom Linux derivative to boot from optical media or a removable USB, you may need different tools to generate the ISO image. If possible, chose a tool that lets you generate hybrid images (i.e., ones that can boot from optical or flash media as needed).

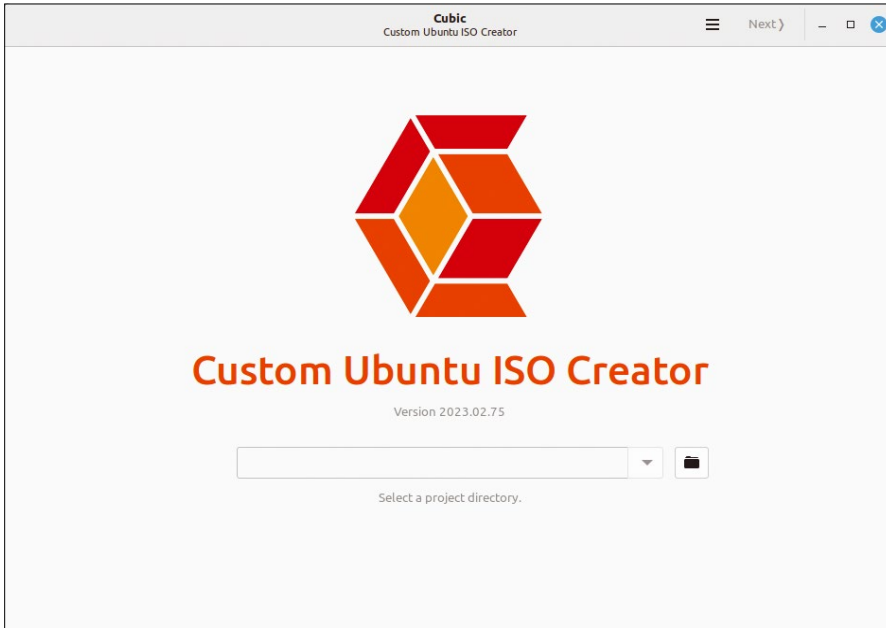
## Cubic

The Custom Ubuntu ISO Creator, or Cubic [1] for short, can be used on Ubuntu, Debian, and their derivatives to create individual ISO images. You can pick up the software from a separate repository. To add the archive to your package management system, first open a terminal and update your system. Then include the Cubic repo in your running system and install the software (Listing 1). After that, you can launch the tool from the system menu to open a virtually empty window (Figure 1).

When you get there, start by selecting a project directory where the data for the

### Listing 1: Cubic Setup

```
$ sudo apt-add-repository ppa:cubic-wizard/release
$ sudo apt update
$ sudo apt install --no-install-recommends cubic
```



**Figure 1:** Cubic's startup screen only requires you to specify a working directory.

ISO image to be generated will be stored. To do this, click on the folder icon bottom right next to the empty path display, and then select the desired directory in the file manager. You can apply the path by clicking the green *Select* button top right in the file manager. Then, in the main window, click *Next* top right.

In the next dialog box, you define the source and the target images. Cubic does not generate the new image from a running system, but from an existing ISO image. You can use any image file. Ideally, you will want to use an image that is as lightweight as possible.

After selecting the source image, Cubic fills out most of the fields in the dialog automatically. It checks the boxes to the left of each completely and correctly filled out field. If problems occur, or Cubic cannot determine certain data from the source image, it places a question mark after the respective field. You can ignore optional information such as the release URL and leave the related fields empty.

After clicking *Next*, Cubic analyzes the source image. If problems occur (e.g., the image is not an Ubuntu or Debian derivative), the program displays an error message. For successfully identified images, Cubic opens and extracts the data. You then click *Customize* at the top right of the window to enter a virtual terminal, with Cubic executing a *chroot*.

You can now work with the extracted ISO image at the prompt (Figure 2).

Because you are working as the admin in the virtual terminal, you can enter commands directly without prefixing *sudo*. As a first step, you should update the ISO image by typing:

```
apt update && apt upgrade
```

Depending on the image's age, this can generate a considerable download volume.

After updating the system, use

```
apt purge APPLICATION
```

to remove any unwanted applications included in the standardized ISO image. Vice versa, use

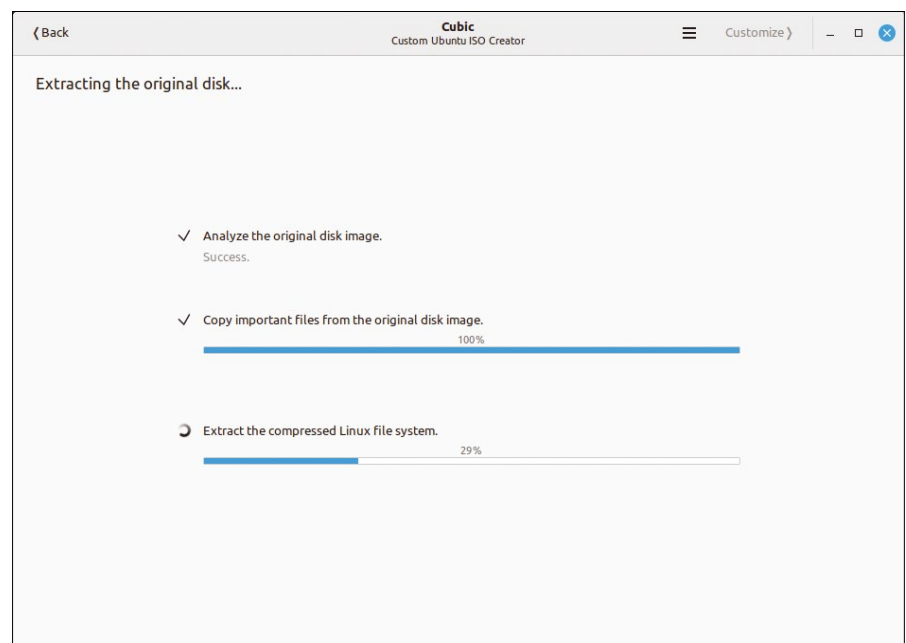
```
apt install APPLICATION
```

to add any desired applications to the custom ISO image.

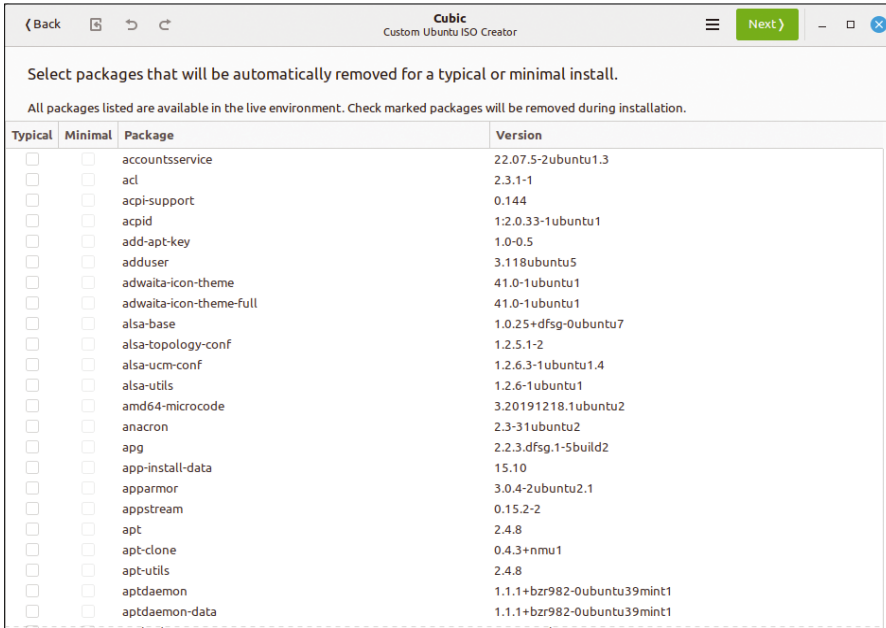
If you do not know exactly which applications the source ISO includes, you can also install the new applications first after completing the update and then press *Next* in the top right corner of the terminal. Cubic then analyzes the updated image and directs you to a window with a list of available applications (Figure 3). In the window, check the boxes of the listed applications that you want to delete. Typical candidates include, for example, the many additional packages for localizing LibreOffice.

Now, switch to the next dialog where you can choose between different kernels for the new image. In the following and final dialog, you need to specify the compression for the new filesystem. You have a choice of several methods here, with Cubic displaying their compression efficiency and the relative size of the resulting image in a graph (Figure 4). Choose the desired compression method by pressing the radio button to the left of the option.

Then click *Generate* to start creating the new ISO image. Cubic goes through several steps, which it displays individually and documents with progress bars.



**Figure 2:** You need to unpack the source image before editing it.



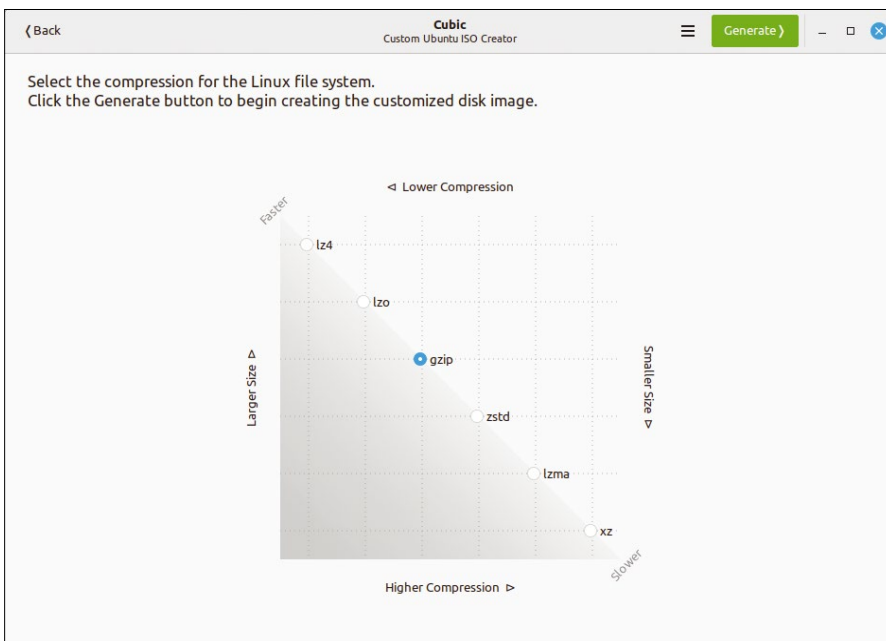
**Figure 3:** Cubic lists all applications present in an image.

Depending on the size of the image to be created and your computer's resources, this can take some time. After successfully completing each step, Cubic will show you some data about the new image in an overview window.

For an initial test, open a Qemu virtual machine (VM) by clicking the green *Test* button at top left; Cubic will start the new live system in the VM. After closing the VM, you are taken back to the last dialog in Cubic and can exit the application by pressing *Close*. This completes the process of generating the new image.

### MX Snapshot

MX Linux, based on Debian and AntiX Linux, offers several desktop environments, including a visually updated version of the lean Xfce desktop. One of MX Linux's many in-house tools, MX Snapshot [2], lets you compile an individual image from the running system. You will find MX Snapshot in each MX Linux desktop environment's menu. MX Linux generates a snapshot that can be transferred to a USB memory stick or optical disc, which you can then use like a live system, unlike a conventional system backup.



**Figure 4:** Cubic shows you which compression method is the most efficient.

After launching MX Snapshot, you are taken to a dialog box that tells you whether there is enough free space on the disc to generate the ISO image. If needed, change to another directory to create the snapshot. In the same dialog, you need to specify the ISO image name; you can also define some boot options for the GRUB boot manager in this dialog. Once you have checked all the information and modified it if necessary, press *Next* (bottom right).

The next dialog lets you exclude directories (if desired) from integration into the snapshot by checking their boxes. The directories to exclude will typically be subfolders below the home directory. You also can skip some optional folders in the snapshot by clicking the *Edit Exclusion File* button. The `/etc/mx-snapshot-exclude.list` exclude file already contains an extensive list of directories to exclude, including hidden files and folders. You can easily add more content to this text file – also in the form of placeholders – that you do not want to include.

Once you have completed the exclude list, select the compression method for creating the ISO image in the dialog under *Options* (Figure 5).

Under *Type of snapshot*, you can modify the type of ISO image. Select *Preserving accounts (for personal backup)* to keep user accounts, which is especially useful if you need to back up user data as well. Select *Resetting accounts (for distribution to others)* to generate an image without user accounts, which is mainly useful for creating an image for installation on different users' computers.

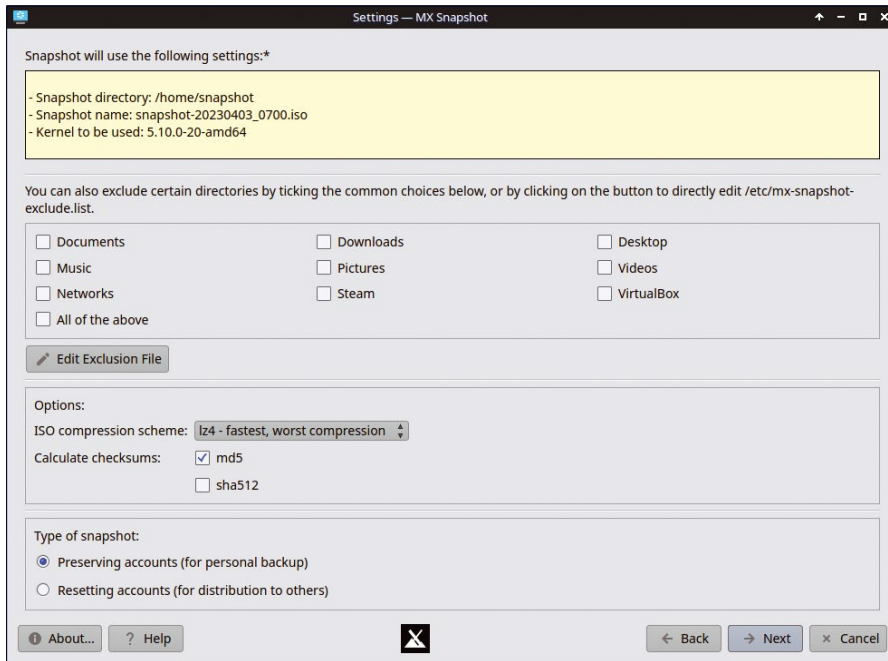
After clicking *Next* again, MX Snapshot warns that creating the image file can take some time. Press *OK* to start generating the ISO image. A separate window now opens with a progress bar and log entries (Figure 6).

After completing the image, a small window with a success message appears; you can close this and exit MX Snapshot. Next, transfer the ISO image you generated to a USB stick as a live system using the MX Live USB Maker tool, for example.

### MyLiveGTK

PCLinuxOS, which originated in the USA, has been around for many years in





**Figure 5:** MX Snapshot supports detailed editing of the new ISO image's contents.

a wide variety of variants with diverse working environments. Originally derived from Mandrake Linux, PCLinuxOS exclusively runs on 64-bit hardware and offers all common applications for everyday office use. In addition to tools specific to each desktop, PCLinuxOS comes with many of the original Mandrake Linux graphical tools.

With APT-RPM as its package management system, PCLinuxOS uses Synaptic as its graphical front end, which supports convenient installation of additional software. To create individual images, PCLinuxOS comes with the `mylivecd` command-line program preinstalled. In the repositories, you will find MyLiveUSB, a graphical tool for creating a multibootable USB removable disk, and MyLiveGTK, the graphical front end for creating bootable ISO images [3]. Unlike Cubic on Ubuntu or Debian, the tools on PCLinuxOS do not require an external ISO image as a working basis. Instead, the tools generate the individual image directly from a running system.

You need to run Synaptic to delete obsolete programs and then individually install the desired applications into the running system. Don't forget to include the `mylivegtk` and `myliveusb` packages on the running system. To generate an image of the running system, open the MyLiveGTK tool via your current working environment's menu.

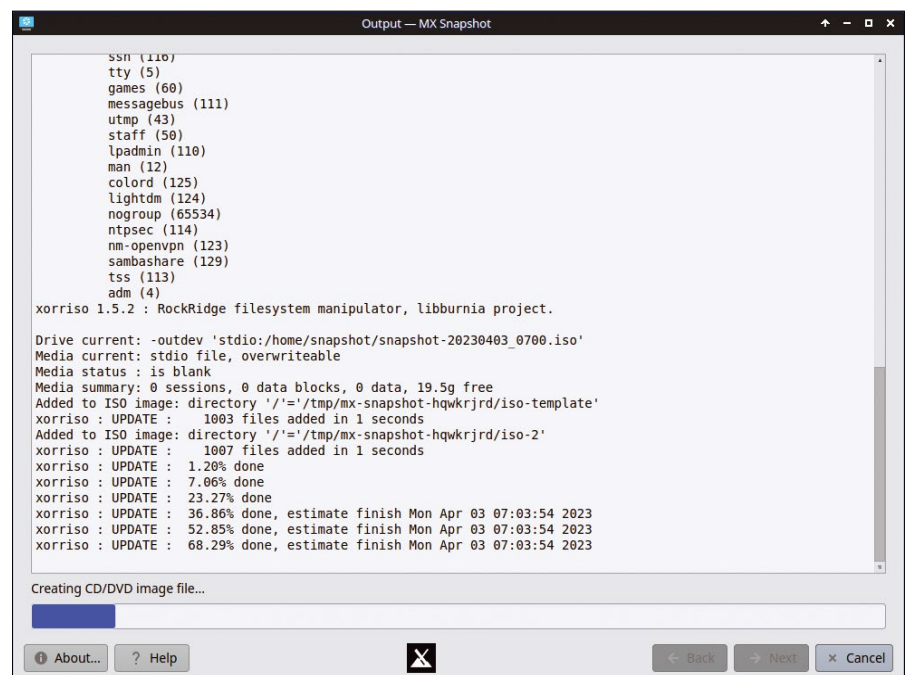
Although confusing at first, a dialog now opens that is actually quite easy to use on closer inspection. At the top of the dialog, you specify the name of the ISO image to be generated and the path for the temporary files. At the center of the window, define the paths to the individual directories and files to be excluded from your ISO image by using the *Add Directory* and *Add File* buttons below the two list areas for the folders

and files. Clicking one of these buttons opens a file manager where you can specify the complete paths for the files and folders to be excluded.

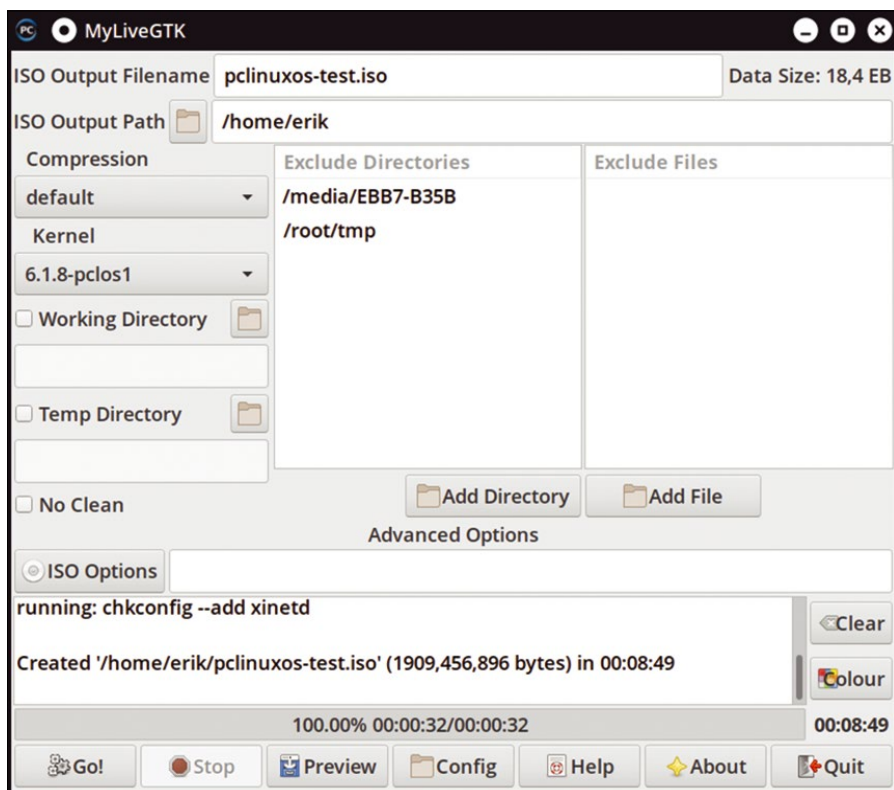
Once you have entered all the data, specify the compression for the new image and the kernel version to use (if there is a choice). By default, MyLiveGTK adopts the current kernel into the ISO image. When done, press *Go!* bottom left and the tool will generate the ISO image based on your settings. At the bottom of the dialog, MyLiveGTK displays messages from the current actions (Figure 7).

Completing the custom image can take more than an hour depending on the software content and your computer's resources. MyLiveGTK uses a progress bar to show you the progress of the action. To transfer the new ISO image to a USB stick, you then need the MyLiveUSB program, which you can access from the desktop menu.

After launching MyLiveUSB and authenticating as a system administrator, the tool prompts you to plug a memory stick into the computer. If you comply with this request, it automatically detects the stick and displays it in a separate dialog. The application does not identify USB sticks that have been plugged in previously, and detection also fails if the stick does not have a FAT32 partition.



**Figure 6:** MX Snapshot provides accurate logs of each step in the process of creating the new ISO image.



**Figure 7:** MyLiveGTK lets you define all the settings for creating a new ISO image in a single dialog.

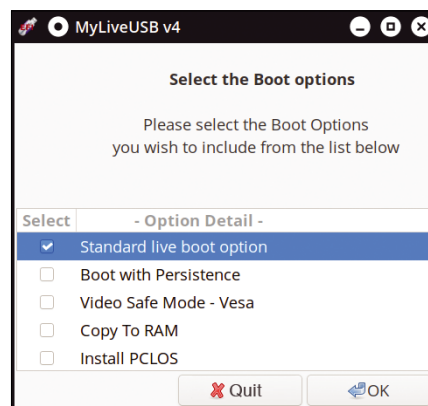
Once MyLiveUSB displays the memory stick, click *Restructure* to delete the existing filesystem and create a new partition scheme on the medium. MyLiveUSB adds an EFI boot system to the stick and creates several partitions. At the same time, a persistent area is created for backing up the user data later. In the next dialog, specify the ISO image's content you want to transfer to the USB flash drive. To help you here, MyLiveUSB displays a selection field that you can populate using the file manager.

In the final dialog, define the USB memory stick boot options you want the tool to add for the GRUB boot manager. Enable the desired settings by checking the boxes

(Figure 8). When done, MyLiveUSB transfers the ISO image to the memory stick. To transfer the custom ISO image to an optical medium, use one of the conventional graphical front ends for creating optical media, such as Brasero or K3b.

### Conclusions

Graphical front ends for customizing bootable ISO images are surprisingly rare on Linux (see the “Not in the Running” box for tools we didn't consider). Unlike the matching command-line tools, graphical front ends support intuitive operation and leave virtually nothing to be desired. However, the tools usually originate from a specific distribution family



**Figure 8:** MyLiveUSB also lets you define multiple boot options.

and only work within its framework. As of writing, AppImage or Flatpak packages for the applications presented here do not exist. However, users of Debian and Ubuntu (and their derivatives), PCLinuxOS, and some other distributions with the RPM package manager, can easily create custom ISO images with the help of their graphical front ends (see Table 1 for a comparison). ■■■

### Not in the Running


While Linux offers various graphical tools for creating individual ISO images, many of these projects have since been discontinued. Customizer [4], a program for remastering a live Ubuntu system, has been orphaned for four years, and Bodhibuilder [5], a program integrated into Bodhi Linux, has failed on recent Ubuntu and Linux Mint variants due to dependencies that can no longer be resolved. Programs like Penguins' Eggs [6] or the Linux Live Kit [7] were not considered for this article because they lack a graphical user interface.

### Info

- [1] Cubic: <https://github.com/PJ-Singh-001/Cubic>
- [2] MX Snapshot: <https://github.com/MX-Linux/mx-snapshot>
- [3] Guide for MyLiveCD and MyLiveGTK: [https://pclinuxohelp.com/index.php/LiveCD,\\_Create\\_your\\_own](https://pclinuxohelp.com/index.php/LiveCD,_Create_your_own)
- [4] Customizer: <https://github.com/kamilion/customizer>
- [5] Bodhibuilder: <https://sourceforge.net/projects/bodhibuilder/>
- [6] Penguins' Eggs: <https://penguins-eggs.net>
- [7] Linux Live Kit: <https://www.linux-live.org>

**Table 1:** Apps for Creating Individual ISO Images

	Cubic	MX Snapshot	MyLiveGTK
License	GPLv3	GPLv3	GPLv3
<b>Functions</b>			
Architecture	64 bit	64 bit	64 bit
ISO image from ISO image	Yes	No	No
ISO image from running system	No	Yes	Yes
ISO image with defined users	No	Yes	No
ISO image without users	Yes	Yes	Yes
Kernel selection	Yes	No	Yes
Select compression method	Yes	Yes	Yes
Specify GRUB boot parameters	No	Yes	No

The  **GNOME**<sup>™</sup> Conference

# GUADEC

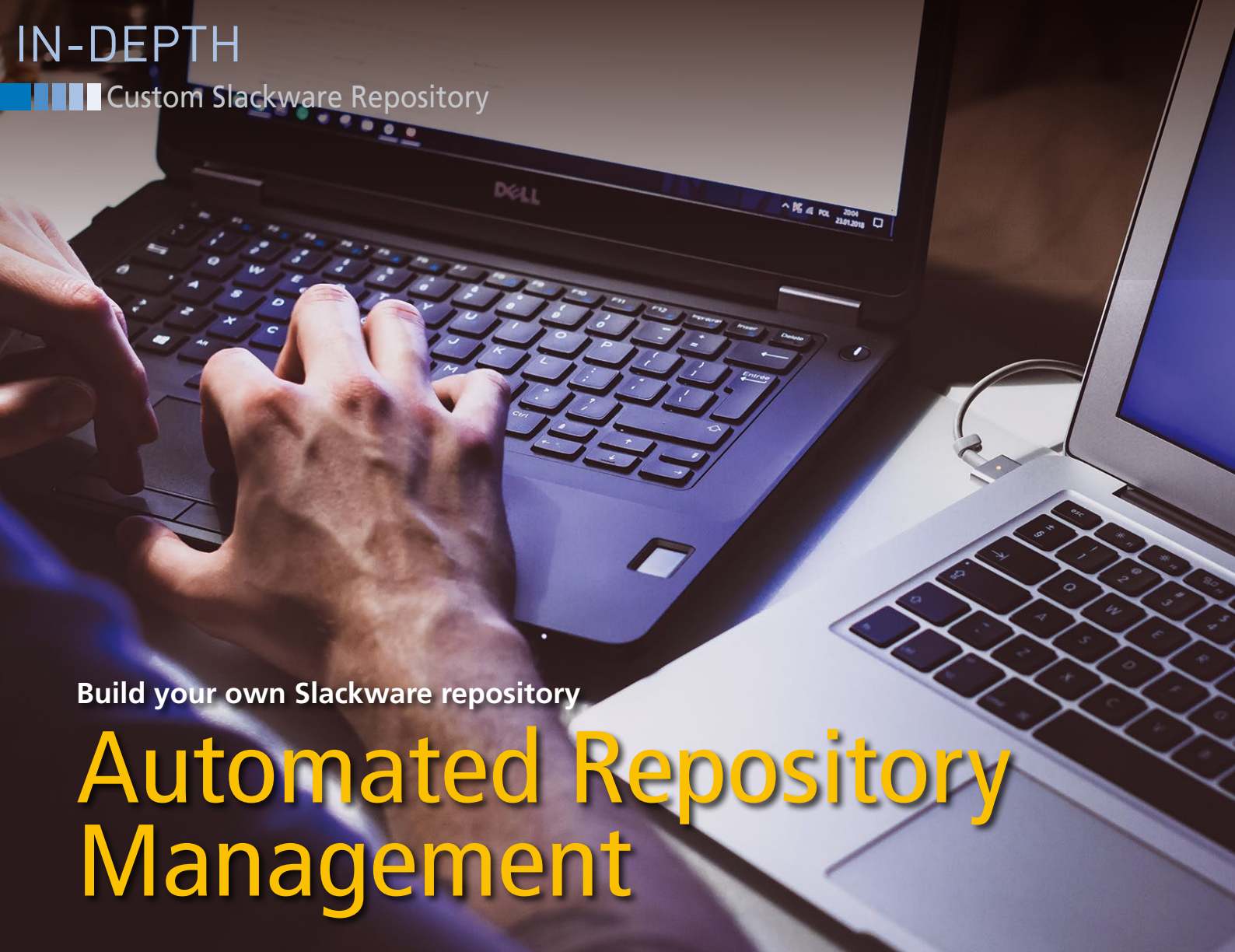
Riga, Latvia  
July 26–31, 2023

GUADEC is GNOME's main annual event. Held since 2000, it brings together Free Software enthusiasts and professionals from all over the world.

Join us at GUADEC 2023, in Riga or online, to hear about the latest technical developments, attend talks, participate in workshops, and celebrate GNOME!

Learn More  
**GUADEC.ORG**





Build your own Slackware repository

# Automated Repository Management

If you deploy software packages to several computers, the standard Slackware tools lack efficiency. We show you how to create a custom repository to automatically install and upgrade software for multiple systems. *By Rubén Llorente*

Slackware [1] has a bad reputation when it comes to package management. Many new users experience Linux through distributions that include advanced package managers. Tools such as APT or Discover [2] set expectations and, for a new user, define how a package manager functions as well as its capabilities. Modern package managers can track software dependencies, purge software no longer in use, search and find applications in the available repositories, and perform system upgrades. When users come in contact with Slackware, they find Slackware's approach too different from what they are used to and wonder why would anybody like it.

Intended to be installed as a monolithic system, you are expected to install the whole Slackware distribution

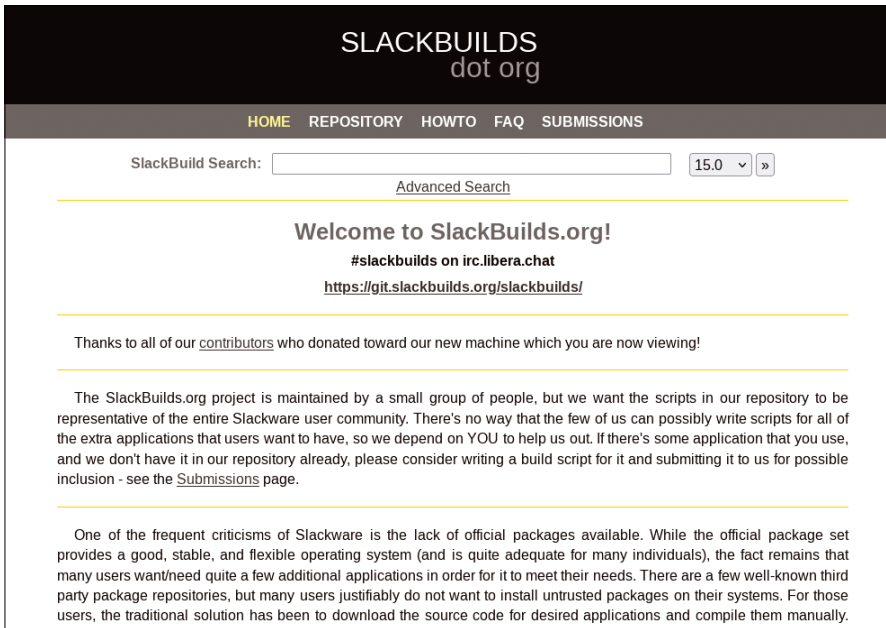
on a machine. Slackware includes lots of libraries, desktop environments, web browsers and, ultimately, most software needed for a personal workstation or server. You can partially install Slackware (e.g., if you plan to run a web server, you may skip installing a graphical desktop environment), but partial installs are not officially supported.

In practical terms, what you get on the DVD is all there is to official Slackware. Unlike APT, there are no official repositories for extra software. Official Slackware repositories exist only to serve upgrades to existing installs. The only official Slackware tool that interacts with the repositories directly, `slackpkg` [3], is designed specifically to fetch security updates and keep your system in sync with the official package tree.

When a Slackware administrator needs to install software not included in the official distribution, they must do so using unofficial means. Traditionally, users would build their own packages from source. Slackware packages have a simple format, and it is very easy to create packages from a project's source code using build scripts, commonly known as SlackBuild scripts. Eventually, the Slackware community created the SlackBuilds.org [4] website (Figure 1), which stores a large stockpile of build scripts that administrators may use to build any required package manually.

SlackBuild scripts still have shortcomings. In order to get a complex application installed, the user must track down the application's dependencies and get the build scripts and source code for each application. SlackBuild scripts

Photo by freestocks on Unsplash



**Figure 1:** SlackBuilds.org hosts an amazing number of SlackBuild scripts that allow administrators to package applications from source code.

make the process easier, but they don't automate it. To solve this, the Slackware community has developed some useful third-party tools that make use of SlackBuilds.org. Programs such as sbopkg [5] and sbtools [6] can replicate

### Ports-Based Systems

Ports are scripted mechanisms designed to build programs from source. The user issues a command to install a specific piece of software, and the ports system builds and installs it alongside its dependencies.

Gentoo is probably the most popular source-based Linux distribution. It makes use of a complex (if powerful) ports system. Gentoo supports flags, which let you configure Gentoo to build itself according to your specifications. With flags, you can build your distribution without components you might not want, such as PAM, PulseAudio, or systemd.

The BSD operating system family also makes use of ports. FreeBSD, OpenBSD, NetBSD, and DragonflyBSD can use either their own ports systems or regular repositories with binary packages (with some skill, a combination of both in some cases). This approach is quite powerful and offers the best of both worlds.

NetBSD's port system, pkgsrc [7], is designed to be portable and may be used on other operating systems (MINIX 3, illumos, and others) to some degree.

the Apt experience: If you instruct these tools to install a package, they will build, package, and install the requested program alongside its dependencies automatically. In particular, sbtools replicates using a ports system like the ones used by the BSD family of operating systems.

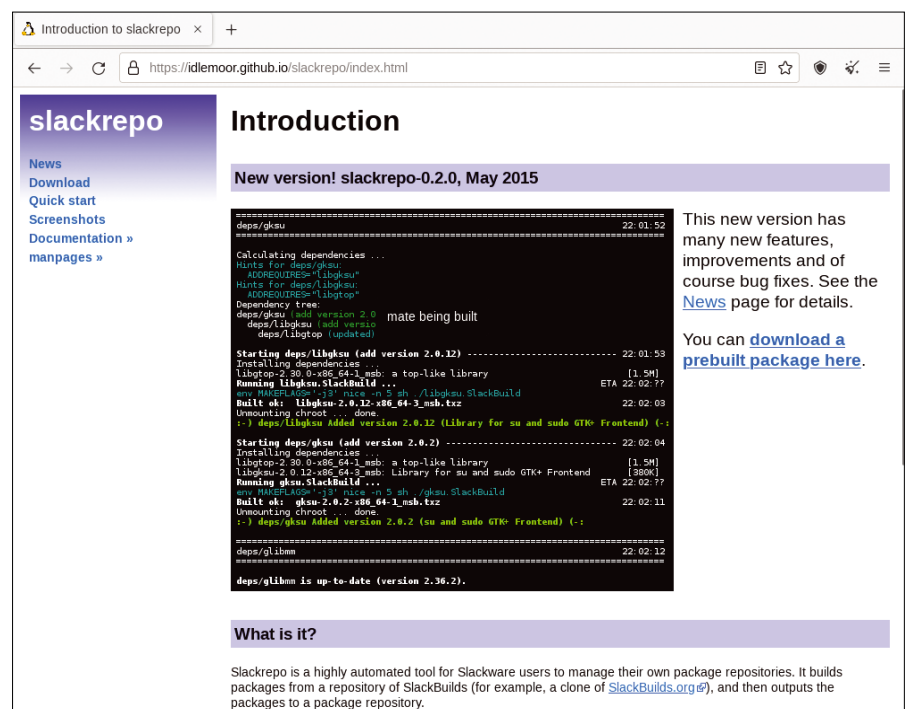
Unfortunately, these tools are not very helpful if you intend to manage a fleet of

Slackware computers. Issuing an `install` command using one of these pseudo-ports systems on every computer would cause all the software to be downloaded once for each machine, and then every computer would compile and install its own packages. This is very inefficient. Instead, if you are responsible for a sizable number of Slackware machines, a custom repository allows you to compile each package just once and distribute it to an indefinite number of machines with minimal effort.

### Why Bother with Slackware?

You may be wondering why you would bother using the Slackware package system. Instead of going to the trouble of using a ports system or building a custom repository, you could use a distribution such as Devuan that offers ready-made repositories.

It comes down to flexibility. With Devuan, if you use the stable version (Devuan Chimaera) and find out you need a more recent version of a library, upgrading the library might prove challenging. Just picking up the version you need from a development branch may not work, because the package manager may refuse to install it unless you bring along a whole set of



**Figure 2:** Slackrepo's original homepage (shown here) has been abandoned, but the program lives on under the maintainership of Andrew Clemons.

dependencies. Changing a library may also cause you to break the applications that use it. While you can upgrade some isolated components in binary distributions, the process can become complex.

When you use source-based software provisioning, you are provided with an *automated* way of solving these conflicts out of the box. If you need to apply a patch to one of the installed programs, or upgrade it to a newer version, the only thing you need to do is to bump a number in a configuration file in your build system. The computer will build your upgraded package specifically for your environment and deploy it with (usually) no conflicts.

To learn more about other popular distributions that use a software provisioning approach, see the “Ports-Based Systems” box.

## slackrepo

Slackrepo [8] (Figure 2), created by David Spencer and maintained by Andrew Clemons [9], offers the best way to create a custom Slackware repository.

Slackrepo is a set of tools capable of taking a set of build scripts and

compiling any packages you may need distributed to your systems. You can load packages created this way onto a web server, which may be used as a formal repository.

This article provides precise instructions on how to create a custom repository with slackrepo and then configure your Slackware computers to use it. The slackrepo server will be in our LAN with the IP address *192.168.3.40* in this example, while our other Slackware machines will have arbitrary addresses in network *192.168.3.0/24*.

In order to get started, you first need to install a full Slackware system on the server. Slackrepo is intended to run on a fresh install with all the official software installed. Listing 1 summarizes building slackrepo using a build script from SlackBuilds.org along with its “hintfiles,” which are instructions for slackrepo to build some tricky software. Slackrepo can use fakeroot as an optional dependency, which allows it to prevent the repository building engine from interfering with the host operating system.

By default, slackrepo is configured to download the whole repository of

SlackBuild scripts from SlackBuilds.org, which is what most administrators will want. Slackrepo updates the local SlackBuilds tree every time it is used and if more than one day has passed since it was last executed.

In order to create a formal repository, you must create GPG keys to sign the packages. Package signatures are used by the clients to verify the integrity of the packages downloaded from your repository. Listing 2 shows you how to create signing keys on the slackrepo server and then output the file GPG-KEY, which will be distributed to the clients. Clients will use this file to validate the packages. Beware: The instructions in Listing 2 aren’t particularly safe, because the OpenPGP key is left unencrypted, but I am choosing this method for the sake of clarity.

You then must configure slackrepo to use this key. Edit the `/etc/slackrepo/slackrepo_580.conf` file and perform the changes shown in Listing 3.

To ensure your values are correct, generate a valid R55\_UUID with the command `uuidgen -t`.

You are now ready to start deploying your packages. As a demonstration, I’ll

### Listing 1: Install Slackrepo on the Server

```

01 # Download the SlackBuild script for fakeroot, unpack it,
    download fakeroot and
02 # build it using the script.
03
04 cd /tmp
05 wget https://slackbuilds.org/slackbuilds/15.0/system/
    fakeroot.tar.gz
06 tar -xzf fakeroot.tar.gz
07 cd fakeroot
08 wget http://ponce.cc/slackware/sources/repo/
    fakeroot_1.25.3.orig.tar.gz
09 bash ./fakeroot.SlackBuild
10 installpkg /tmp/fakeroot*.tgz
11
12 # Download the SlackBuild script for slackrepo, unpack
    it, download slackrepo and
13 # build it using the script.
14
15 cd /tmp
16 wget https://slackbuilds.org/slackbuilds/15.0/system/
    slackrepo.tar.gz
17 tar -xzf slackrepo.tar.gz
18 cd slackrepo
19 wget https://github.com/aclemmons/slackrepo/archive/
    v20230107/slackrepo-20230107.tar.gz
20 bash ./slackrepo.SlackBuild
21 installpkg /tmp/slackrepo*.tgz
22
23 # Download the SlackBuild script for the hintfiles,
    unpack it, download the hintfiles and
24 # build them using the script.
25
26 cd /tmp
27 wget https://slackbuilds.org/slackbuilds/15.0/system/
    slackrepo-hints.tar.gz
28 tar -xzf slackrepo-hints.tar.gz
29 cd slackrepo-hints
30 wget https://github.com/aclemmons/slackrepo-hints/archive/
    v20230128/slackrepo-hints-20230128.tar.gz
31 bash ./slackrepo-hints.SlackBuild
32 installpkg /tmp/slackrepo-hints*.tgz

```

### Listing 2: Creating Keys for Package Signatures

```

01 gpg2 --batch --yes --passphrase="" --quick-generate-key "Repomaster <repomaster@example.org>" dsa
02 gpg2 --export -a Repomaster > GPG-KEY

```

build the Hedgewars package [10] along with its dependencies:

```
# slackrepo build hedgewars
```

A minimal repository will be placed in `/var/lib/slackrepo/SBo/packages`. You may compile as many additional packages as needed, and all the packages will be placed here. You may also move everything to a web server folder at this point:

```
# rsync --delete -as \
  /var/lib/slackrepo/SBo/packages/ \
  /var/www/htdocs/
# mv GPG-KEY /var/www/htdocs/15.0/x86_64/
```

Now, you just need to enable and launch the web server included with a Slackware full install:

```
# chmod +x /etc/rc.d/rc.httpd
# /etc/rc.d/rc.httpd start
```

You will now find your repository online at <http://192.168.3.40> (as shown in Figure 3).

## Client Setup

You now have a package repository, but how can you leverage it?

Packages stored in slackrepo repositories may be downloaded and installed manually, but Slackware supports a number of tools for automating the task. The previously mentioned `sbopkg` is a popular choice, but for this example, I am going to use `slackpkg`, because it is closer to the standard Slackware operational model.

While `slackpkg` is intended for use only with official Slackware repositories, the

`slackpkg+` plugin lets you work with separate repositories. You can install `slackpkg+` on a given client machine as follows:

```
# wget https://slakfinder.org/slackpkg+/
pkg/slackpkg+-1.8.0-noarch-7mt.txz
# installpkg slackpkg+-1.8.0-noarch-7mt.
txz
```

Next, you need to configure `slackpkg+`. The simplest way is to edit `/etc/slackpkg/slackpkgplus.conf` on each client machine and set the variables shown in Listing 4.

### Listing 3: slackrepo.conf Variables

```
01 USE_GENREPOS=1
02 REPOOWNER="Repomaster <repomaster@example.org>"
03 DL_URL=http://192.168.3.40/pkgrepo/%REPO%/%SLACKVER%/%ARCH%"
04 RSS_UUID=SOME_EXAMPLE_UUID
05 GPGBIN=/usr/bin/gpg2
06 FOR_SLAPTGET="1"
```

### Listing 4: Setting Variables in slackpkgplus.conf

```
01 REPOPLUS=( custom )
02 MIRRORPLUS[' custom'] =http://192.168.3.42/15.0/x86_64/
```

# IT Highlights at a Glance



Too busy to wade through press releases and chatty tech news sites? Let us deliver the most relevant news, technical articles, and tool tips – straight to your Inbox.


Linux Update • ADMIN Update • ADMIN HPC

Keep your finger on the pulse of the IT industry.

ADMIN and HPC: [bit.ly/HPC-ADMIN-Update](http://bit.ly/HPC-ADMIN-Update)

Linux Update: [bit.ly/Linux-Update](http://bit.ly/Linux-Update)

Finally, uncomment any of the official Slackware mirrors of your choice in `/etc/slackpkg/mirrors`. The repository may now be used to install any package you require:

```
# slackpkg gpg-update
# slackpkg update
# slackpkg install physfs ghc lua fpc 
hedgewars
```

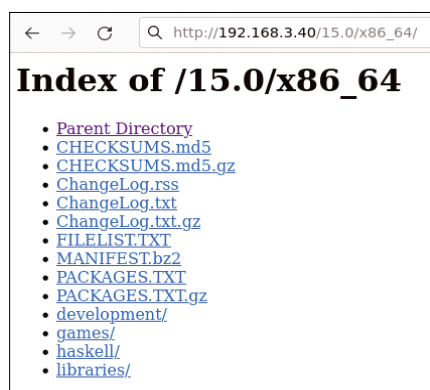
Keep in mind `slackpkg` does not do dependency tracking. However, when you perform your regular Slackware upgrades, any package from your custom repository that gets upgraded in the server will be picked up by `slackpkg`:

```
# slackpkg update
# slackpkg upgrade-all
```

## Custom Packages

At some point, you may need to build a custom package. If you want to package software not included in the SlackBuilds database, the best approach is to create your own SlackBuild script and submit it to SlackBuilds.org. This way, the community benefits, and you will have the build script available from a high-availability site.

If you need to build a package with a particular set of compilation options,



**Figure 3:** The repository built by `slackrepo` contains the Hedgewars packages with its dependencies. `Slackrepo` also supports delivering signed packages, which allow the clients to validate downloads before installation.

then your choices narrow. `Slackrepo`, by default, uses Git to synchronize a SlackBuilds tree stored in `/var/lib/slackrepo/SBo/slackbuilds` with the official SlackBuilds.org server. In the event that you need to use your own unofficial SlackBuilds or modifications of official SlackBuilds, the `slackrepo`'s original maintainer suggests that you set a Git branch based on the official branch you are taking as a foundation. Then, official updates to the SlackBuilds can be merged with your own branch periodically as needed [11]. Keep in mind that `slackrepo` allows you to set the Git repository and the branch you want to track in each repository's configuration file.

`Slackrepo` uses hintfiles to perform certain special actions required by some special packages. For example, the `tor` package requires user `tor` to be created for running the daemon. You can edit the hintfiles, which are stored in a dedicated folder under `/etc/slackrepo/SBo`, to your liking [12]. However, it is recommended not to edit the files in the `default_hintfiles` directory. Instead, add your overrides to the `hintfiles` directory, which will allow your edits to persist across upgrades.

## Conclusion

Creating a custom Slackware repository is easy if you can spare a build rig. If you only administer a couple of Slackware machines, a custom repository is not worth the effort. However, if you operate a big fleet of computers, having your own repository will save you a lot of time and headaches.

Keep in mind there are many binary repositories for Slackware packages, some of which have a trustworthy reputation. Eric Hameleers keeps a personal repository [13] that integrates well with `slackpkg+`, which he makes available to the public. Panagiotis Nikolaou operates the SlackOnly [14] repository, which is built using `slackrepo` against official SlackBuilds.

If you build packages from build scripts sourced from SlackBuilds.org, you cannot distribute them in a way that

suggests they are related to the SlackBuilds project. This means you need to remove the `SBo` tag from the package names if you ever want to share your packages with third parties. ■■■

## Info

- [1] The Slackware Linux Project: <http://www.slackware.com>
- [2] Discover: <https://apps.kde.org/discover/>
- [3] `slackpkg` from the Slackware Documentation Project: <https://docs.slackware.com/slackware:slackpkg>
- [4] SlackBuilds.org: <https://www.slackbuilds.org/>
- [5] `sbopkg`: <https://sbopkg.org/>
- [6] `sbotools`: <https://pink-mist.github.io/sbotools/>
- [7] `pkgsrc` from NetBSD: <https://www.netbsd.org/docs/pkgsrc/introduction.html#intro.platforms>
- [8] Original `slackrepo` homepage: <https://idle Moor.github.io/slackrepo/index.html>
- [9] Andrew Clemons' fork of `slackrepo`: <https://github.com/aclemoms/slackrepo>
- [10] Hedgewars: <https://www.hedgewars.org/>
- [11] David Spencer on using custom SlackBuilds: <https://www.linuxquestions.org/questions/slackware-14/slackrepo-and-custom-slackbuild-417592674/#post5625878>
- [12] Hintfile documentation: <https://idle Moor.github.io/slackrepo/hintfiles.html>
- [13] Eric Hameleers' repository: <http://www.slackware.com/~alien/slackbuilds/>
- [14] SlackOnly repository: <https://slackonly.com/>

## Author

Rubén Llorente is a mechanical engineer who ensures that the IT security measures for a small clinic are both legally compliant and safe. In addition, he is an OpenBSD enthusiast and a weapons collector.





# Looking for your place in open source?



Set up job alerts and get  
started today!

**OpenSource**  
JOB HUB



[opensourcejobhub.com/jobs](https://opensourcejobhub.com/jobs)



### Compiling coreboot firmware

# Start from Scratch

Coreboot lets you build your own custom firmware while learning more about Linux. *By Bruce Byfield*

The open source alternative to BIOS and UEFI, coreboot is often praised for its speed and security [1]. However, just as important is the accessibility of its firmware. According to the coreboot website, “The architecture of coreboot is designed to have an unbrickable update process. Updating firmware should be no more dangerous than installing your favorite app on your mobile phone.” As the major result of this goal, users can choose to customize their own firmware. Much like customizing the Linux kernel, the process is lengthy but mostly a matter of following instructions in the thorough documentation. Although the documentation is oriented towards Debian, it can easily be transposed to other distributions, the main difference being the occasional difference in the names of some of the necessary packages.

You may want to build your own firmware for several reasons. You might want to install coreboot on a machine that already has its own BIOS or UEFI. If your

machine already uses coreboot, you might want to tweak it to better suit your needs. For example, you may prefer to turn off virtualization support in order to get every bit of performance from your machine. Sometimes, too, an unsupported mainboard might be usable with a few experimental tweaks. Perhaps the main reason is that building coreboot firmware can satisfy your curiosity about a technology that is still unfamiliar to most Linux users.

Despite the project’s goals, some chance exists that you could brick your machine. Should that happen, another attempt at coreboot customization or the manufacturer’s firmware may restore functionality. All the same, before flashing your custom firmware, you should make sure you have a current backup and thoroughly research what you are doing before you begin.

### Preparing to Build

A work in progress, coreboot must be tailored to each mainboard. While several hundred mainboards are supported, you should check that your mainboard is supported before you build your own firmware. The project claims that its list of supported boards is current to within the last hour [2], and some hardware manufacturers maintain their own documentation. Some entries have an upstream link to an existing version of the firmware you can work with, following coreboot’s documentation [3].

You will also need the tools to build the firmware. On Debian, run the command:

```
apt-get install git build-essential \
gnat flex bison libncurses5-dev \
wget zlib1g-dev
```

This collection of packages includes the necessary compiler and other tools needed to build the firmware. You will also need Git to access coreboot’s files. You may also want to install Doxygen in

```
bb@system76-pc:~/coreboot$ git clone https://review.coreboot.org/coreboot
Cloning into 'coreboot'...
remote: Counting objects: 755206, done
remote: Finding sources: 100% (755206/755206)
remote: Total 755206 (delta 582644), reused 754926 (delta 582644)
Receiving objects: 100% (755206/755206), 183.11 MiB | 3.05 MiB/s, done.
Resolving deltas: 100% (582644/582644), done.
```

Figure 1: Coreboot’s source code is stored on Git.

Photo by pure julia on Unsplash

```
bb@system76-pc:~/coreboot/coreboot$ make crossgcc-i386
Welcome to the coreboot cross toolchain builder v2023-04-17_b68817d196

Building toolchain using 1 thread(s).

Target architecture is i386-elf

Found compatible Ada compiler, enabling Ada support by default.

Downloading and verifying tarballs ...
* gmp-6.2.1.tar.xz (downloading from https://ftpmirror.gnu.org/gmp/gmp-6.2.1.tar.xz)... 100%... hash verified (0578d48607ec0e272177d175fd1807c30b00fdf2)
* mpfr-4.2.0.tar.xz (downloading from https://ftpmirror.gnu.org/mpfr/mpfr-4.2.0.tar.xz)... 100%... hash verified (4f734ca3ebceac28e2f944b131a47133b19e2c5e)
* mpc-1.3.1.tar.gz (downloading from https://ftpmirror.gnu.org/mpc/mpc-1.3.1.tar.gz)... 100%... hash verified (bac1c1fa79f5602df1e29e4684e103ad55714e02)
* binutils-2.40.tar.xz (downloading from https://ftpmirror.gnu.org/binutils/binutils-2.40.tar.xz)... 100%... hash verified (fee4fbef9d632afc1988dd631d7f75d4394b7f8d)
* gcc-11.3.0.tar.xz (downloading from https://ftpmirror.gnu.org/gcc/gcc-11.3.0/gcc-11.3.0.tar.xz)... 5%
```

Figure 2: Coreboot smoothes the build process by providing its own toolchain.

order to read package documentation. If you have a compiler preference, GCC, G++, or *gcc-multilib* can replace *gnat*. If you use Clang or LLVM as a compiler, you should also add CMake. Similarly, if your hardware uses Advance Configuration and Power Interface (ACPI) for power management, add *iasl*. If you want a better graphical display when building, add *ncurses-dev*. For many users, the command above should be all you need. With other distributions, you may have to search for slightly different names.

## Build Steps

You can build coreboot on any machine. The process will not alter your machine until you flash the resulting firmware. To begin the build, clone the coreboot files and switch to the directory created for them:

```
git clone https://review.coreboot.org/coreboot
cd coreboot
git submodule update --init --checkout
```

The last command checks out a sub-repository for your use (Figure 1).

The final preparation stage is to build the toolchain. This step is necessary because different distributions include different toolchains. To prevent any problems, coreboot creates its own patched version of GCC, including all dependencies. Enter:

```
make crossgcc-i386
```

The process can take 10-15 minutes, depending on the machine (Figure 2).

## Building Custom Firmware

To build efficiently, you most likely need your hardware's specifications. If you no longer have the specifications or cannot find them online, you can still attempt to build, although success may take several tries. For practice, you might want to follow one of the online tutorials for the Qemu emulator, which requires fewer configuration choices [4].

From the directory with the source code, enter `make menuconfig` to display a text-based interface (Figure 3). Alternatively, if *ncurses* is installed, enter `make nconfig` for a more user-friendly menu. Detailed instructions are available in the coreboot documentation, although for some settings you may be able to accept the defaults. For a system with native graphics, follow the generic instructions in Figure 4, entering the information for your hardware. For non-native graphics, enter the information in Figure 5. Note that for some Intel and AMD boards, you may need to provide files from the vendor. These vendor files are most likely necessary for older hardware, because manufacturers are increasingly cooperating with the coreboot project.

The payload is the code that loads an operating system in the final stage of starting a machine with coreboot. You can use the default payload SeaBIOS or one of the other payloads [5]. Increasingly, hardware vendors like System76 and Star Labs provide their own pay-

loads, which may be necessary in order to use some of their utilities or apps. However, you can build successfully with other payloads as well.

When you have made all your choices, press `Esc` to exit the menu and save your choices. Then

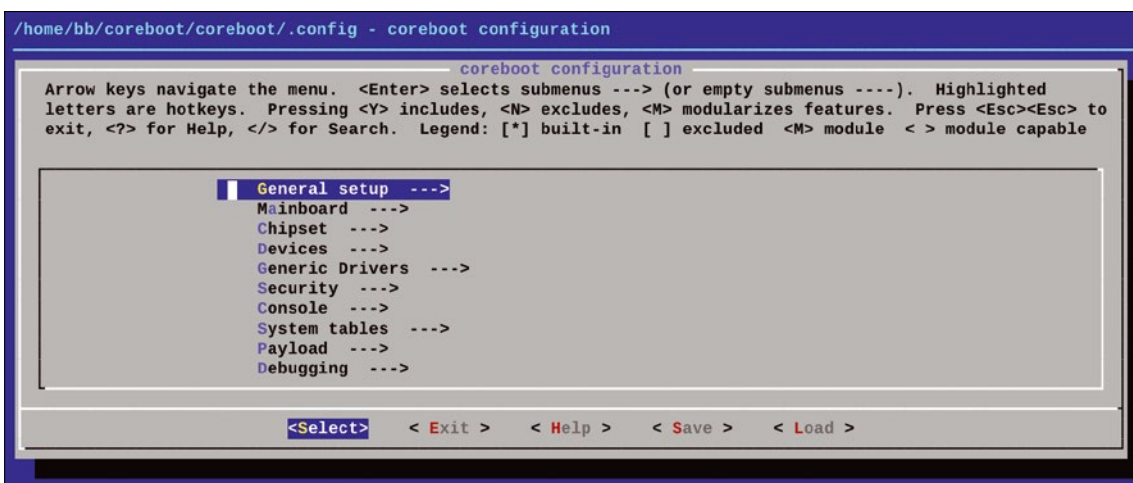


Figure 3: A text-based interface is used for configuring firmware. © <http://www.coreboot.com/>

```

General / Use CMOS for configuration values = enable (CMOS defaults are located in your boards directory
src/mainboard/OEM/MODEL/cmos.default)
Mainboard / Mainboard vendor = Name of manufacturer
Mainboard / Mainboard model = Model name
Mainboard / ROM chip size = size of flash chip
Chipset / Include CPU microcode in CBFS = Do not include microcode updates (NOTE: you probably want to
enable it on some systems)
Devices / Use native graphics initialization = enable (NOTE: not available on all systems)
Display / Keep VESA framebuffer = disable (disable for text-mode graphics, enable for coreboot vesa
framebuffer)
Generic Drivers / USB 2.0 EHCI debug dongle support = Enable
Generic Drivers / Enable early (pre-RAM) usbdebug = Enable
Generic Drivers / Type of dongle = Net20DC or compatible
Generic Drivers / Digitizer = Present
Console / USB dongle console output = enable
Payload / Add a payload = An ELF executable payload (change if you want a different payload)
Payload / Payload path and filename = grub.elf (assumes building GRUB manually. Change this if you want
a different payload)

```

**Figure 4: A generic guide to required firmware options for machines with on-board graphics.** © <http://www.coreboot.com/>

```

Devices / Run VGA Option ROMs = disable
Devices / Run Option ROMs on PCI devices = disable

```

**Figure 5: Machines without on-board graphics require two additional options.**

enter `make` from the directory that contains `coreboot` and any vendor or payload files (Figure 6). At the end of the build process, `make` indicates success or failure, listing errors that you can correct with some research before making another attempt. Only flash a successful result, never a failed one.

## Flashing, Modifying, and Removing

The above instructions are for building firmware from scratch. However, `coreboot` is developing rapidly as its adaptation grows, and a growing number of Linux hardware vendors are making custom firmware more accessible. For instance, System76 offers its own instructions, including scripts to assist in finding hardware specifications [6], and makes installing its own firmware updates much simpler than the steps shown here. Similarly, Ubuntu now ships with a firmware manager. While installing custom `coreboot` configurations are not yet as easy as updating an Android phone, that goal may not be far off.

For now, the easiest way to flash your custom firmware is to use `flashrom` [7], the same utility used to flash a

BIOS. So long as you have researched the required characteristics and built the firmware successfully, you should have few problems.

All the same, to be cautious, be sure to research the process before you attempt it, and backup your files. If you follow the instructions given above, you should be able to use `flashrom`'s internal programmer (i.e., the one used to install a file on the same computer). In the directory that contains your firmware, run:

```
flashrom -p internal -w coreboot.rom
```

You may need to boot the machine entering `iomem=relaxed` in the boot manager. Should this process not succeed, consult the `coreboot` documentation on other methods of flashing, which are usually more complicated [8].

Currently, the easiest way to edit `coreboot` firmware is to repeat the process shown here. However, Star Labs has

recently released a graphical tool called `coreboot-configurator` for Debian derivatives that allows editing of installed firmware and greatly eases the process [9]. Should you want to remove custom firmware for any reason, flashing of another version of `coreboot` firmware or a manufacturer's BIOS and UEFI should restore accessibility to your

machine. All you will have wasted will be your time. However, you will have learned more about Linux in general and an important emerging technology. ■■■

## Info

- [1] `coreboot`: <https://coreboot.org/>
- [2] Supported mainboards: <https://coreboot.org/status/board-status.html>
- [3] `coreboot` documentation: [https://www.coreboot.org/Build\\_HOWTO#debian](https://www.coreboot.org/Build_HOWTO#debian)
- [4] Qemu tutorial: [https://www.coreboot.org/QEMU\\_Build\\_Tutorial](https://www.coreboot.org/QEMU_Build_Tutorial)
- [5] Payloads: <https://doc.coreboot.org/payloads.html>
- [6] System76 documentation: <https://github.com/system76/firmware-open/blob/master/docs/flashing.md>
- [7] `flashrom`: <https://www.flashrom.org/Flashrom>
- [8] Flashing firmware: [https://doc.coreboot.org/tutorial/flashing\\_firmware/index.html](https://doc.coreboot.org/tutorial/flashing_firmware/index.html)
- [9] `coreboot-configurator`: <https://support.starlabs.systems/kb/guides/coreboot-configurator>

```

bb@system76-pc:~/coreboot/coreboot$ make
Updating git submodules.
HOSTCC util/sconfig/lex.yy.o
HOSTCC util/sconfig/sconfig.tab.o
HOSTCC util/sconfig/main.o
HOSTCC util/sconfig/sconfig (link)
SCONFIG mainboard/emulation/qemu-i440fx/devicetree.cb
#
# No change to /home/bb/coreboot/coreboot/.config
#
build/auto.conf:1611:notice: override: reassigning to symbol LAPIC_ACCESS_MODE
HOSTCC nvramtool/cli/nvramtool.o
HOSTCC nvramtool/cli/opts.o
HOSTCC nvramtool/cmos_lowlevel.o
HOSTCC nvramtool/cmos_ops.o
HOSTCC nvramtool/common.o
HOSTCC nvramtool/compute_ip_checksum.o
HOSTCC nvramtool/hexdump.o

```

**Figure 6: Compiling firmware: Only flash files that are successfully built.**

# Turn your ideas into reality!

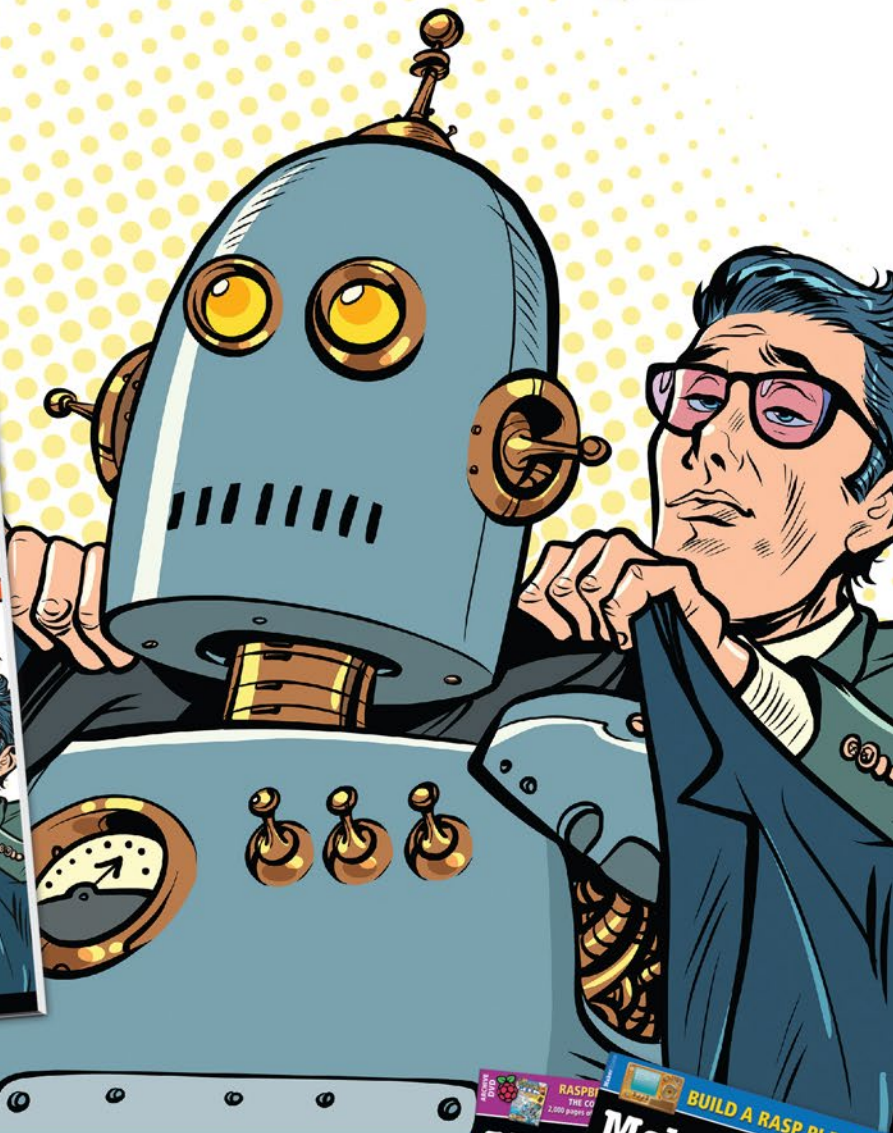
This is not your ordinary computer magazine! *MakerSpace* focuses on technology you can use to build your own stuff.

If you're interested in electronics but haven't had the time or the skills (yet), studying these maker projects might be the final kick to get you started.

This special issue will help you dive into:

- Raspberry Pi
- Arduino
- Retro Gaming
- and much more!

**MakerSpace**  
**#03**



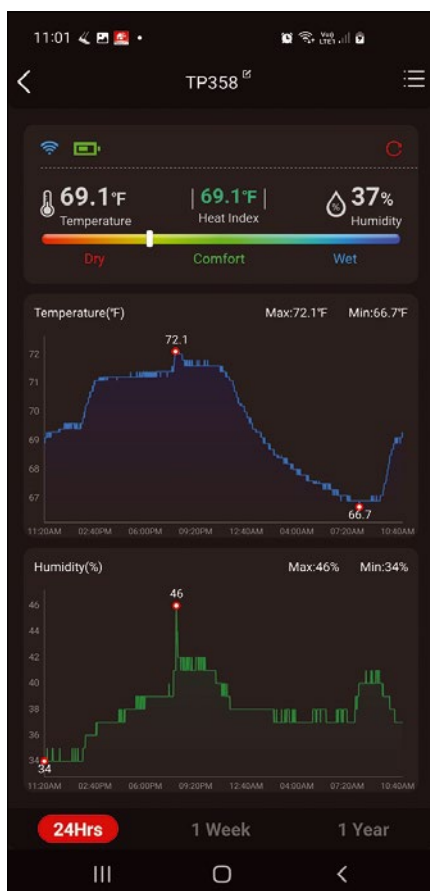
**ALSO LOOK FOR  
MAKERSPACE #01 & #02  
AND ORDER ONLINE:  
[sparkhaus-shop.com/specials](http://sparkhaus-shop.com/specials)**



Reverse engineering a BLE clock

# Perfect Time

What do you do when all your Bluetooth clocks show slightly different times? With some reverse engineering, you can write a Python program to synchronize your clocks. *By Koen Vervloesem*



**Figure 1:** The ThermoPro Sensor app synchronizes the time of the digital thermometer upon connection.

While ago, I bought a ThermoPro TP358, a Bluetooth Low Energy (BLE) digital thermometer with a display. The ThermoPro shows the temperature, humidity, and air comfort indicator, as well as the time and day of the week. Its big display is nice for immediate feedback, but the device also lets you read its values and view graphs in the ThermoPro Sensor app, available on Android and iOS (Figure 1). Moreover, every time you connect to the device with the app, it synchronizes the time.

While that is a nice feature, I have a couple of other types of Bluetooth sensors with a clock, and I didn't want to use multiple apps to view the sensor measurements and synchronize the clocks. For the sensor measurements, a solution already exists: Software such as Home Assistant [1] supported my devices out-of-the-box, letting me view their measurements in Home Assistant's dashboard. However, I couldn't find any solution that let me synchronize the time across all of my Bluetooth clocks without using the individual apps.

From past experience reverse engineering other Bluetooth devices, I knew that it should be possible to intercept the synchronization commands

between my Android phone and the clock. My plan was to figure out the meaning of the commands and then re-implement the same time synchronization command in a Python script that I could run once a day on a Raspberry Pi. This article describes how I did this for the ThermoPro TP358, but you can use the same procedure with any other BLE device.

## Investigating BLE Traffic Logs

There are various ways to intercept BLE traffic. If you're investigating BLE packets sent and received by an Android app, an easy way is to let your phone log its Bluetooth packets while using the app, then transfer the logs to your computer, and open the logs with Wireshark [2] to go through the recorded Bluetooth traffic.

Wireshark is a powerful open source network protocol analyzer. Network administrators use it to analyze network problems, and it also comes in handy to troubleshoot a WiFi or Ethernet network at home. Wireshark can also capture other protocols, including Bluetooth, live or from logs.

To transfer the Bluetooth logs from your phone to your computer, you

need the Android Debug Bridge (adb). Most distributions have both Wireshark and adb in their official repositories, so you can install them with your distribution's package manager. If not, you can find Wireshark on its website and adb as part of the Android SDK Platform Tools [3] package.

## Preparing Your Phone

First you need to enable USB debugging on your Android phone. This can differ on some models, but generally you need to go to *Settings*, tap *About phone*, then *Software information*, and finally tap *Build number* seven times. Then enter your PIN to unlock the *Developer options* menu, which will appear in your *Settings* menu.

Now, reopen the *Settings* menu, go to the new *Developer options* menu, and switch on *USB debugging* (Figure 2). Then tap on *Enable Bluetooth HCI snoop log*. After this, disable and re-enable Bluetooth to start logging Bluetooth traffic.

## Synchronizing Time

Now open the app and let it connect to your BLE device. After clicking on *Add Device*, the app finds all ThermoPro clocks in the vicinity. Tap the plus sign next to one of the clocks. The app then connects to the device, downloads sensor measurements, and synchronizes the time. Then close the app, and write down the current date and time. In my case, this was Sunday, March 5, 2023 at 11:01.

Because the app has communicated with the device, there should be some Bluetooth packets logged. Connect your phone to your computer via USB. Your phone will ask to allow the computer access. Confirm this, and then run the following command on your computer:

```
adb devices
```

This should show your phone as an attached device. If you haven't confirmed access yet on your phone, the device will be listed as "unauthorized."

Thanks to adb, you can generate a bug report file, which is a ZIP file including the `btsnoop_hci.log` file you need in `F5/data/log/bt`. Generate this bug report file and transfer it to your computer with:

```
adb bugreport
```

This can take a while, but the resulting file will have a name like `dump-state-2023-03-05-11-07-00.zip`, from which you will extract `F5/data/log/bt/btsnoop_hci.log`.

## Investigating the BLE Traffic

You can now unplug your phone from your computer. I also recommend disabling USB debugging and the Bluetooth HCI snoop log if you no longer need them. Disable and re-enable Bluetooth to stop logging.

Now start Wireshark and open the *File | Open* menu. Select the `btsnoop_hci.log` file you extracted. Wireshark now shows you the logs from your phone's Bluetooth Host Controller interface. From here, you can start your investigation.

Wireshark shows each BLE packet in a row with columns for the packet number, the time (seconds since the start of the log), the source, protocol, and some information. When the source is controller or host, these are low-level packets you're not interested in. Scroll through the list until you encounter the name of your phone or the BLE clock in the *Source* column.

The phone or BLE clock packets show that your phone starts connecting and asking for the different types of information that your clock offers as BLE services and characteristics. There are a lot of packets flowing in both directions, so it's a bit like searching for a needle in a haystack. However, you are looking for something very specific: a date and time written by the phone to a characteristic of the clock. You can filter the packets on those writes.

## Filtering the Write Packets

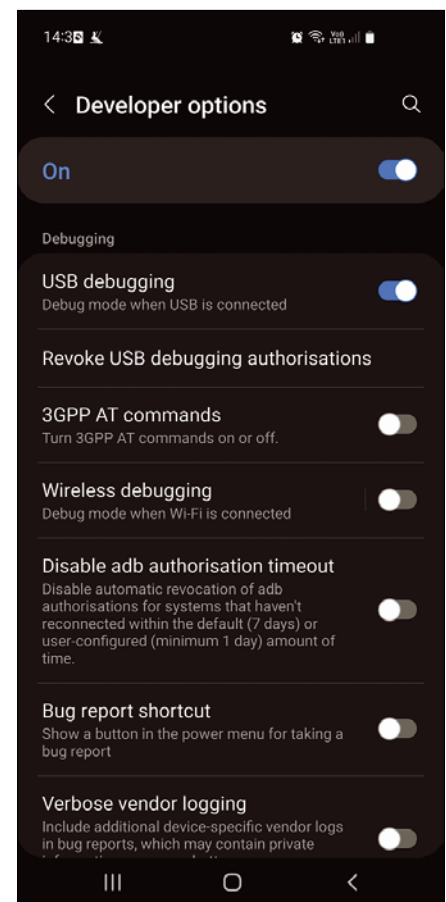
Under the *Info* column, you will see some packets with *Sent Write Request* and others with *Sent Write Command*. A Write Request expects a Write Response, while a Write Command doesn't. However, you don't know beforehand which ones the app uses to write the time, so you'll have to investigate both types.

Click on a line of one of these types, unfold the *Bluetooth Attribute Protocol* drop-down in the pane with details below, and unfold *Opcode*. Now right-click on *Method: Write Request (0x12)*, and then choose *Apply as Filter | Selected*.

The number of shown packets now decreases dramatically (Figure 3), which makes it much easier to determine the command structure. Based on the date and time you wrote down earlier (Sunday, March 5, 2023 at 11:01), you want to look for numbers like 0 or 7 (for Sunday, as the day of the week), 3 (March), 5, 2023, or 23, 11, and 1.

The first Write Request just enables notifications, which is probably needed to get the log of sensor measurements from the device. But the second packet, a Write Command, immediately looks like a winner with a value of `a51703050b010707015a`. The sequence `03050b01` immediately stands out here – it looks like March 5 at 11:01. The hexadecimal value 17 is also not too difficult to understand:  $0x17 = 1 \times 16 + 7 = 23$ ; that's the year.

The two 07 values make decoding a bit difficult. One could be the day of the week, Sunday, but which one? You should create another Bluetooth log while using the app to figure this out. You'll see that every packet always



**Figure 2:** Enable USB debugging and the Bluetooth HCI snoop log in your phone's developer options.

starts with the value *a5* and ends with *5a*, so this looks like a header and footer. The *01* before the *5a* also doesn't change, so this could be part of the footer, or maybe encode something else. Moreover the second *07* stays *07*, so that looks like it encodes the day of the week, while the first *07* changes to something else. Maybe those are the seconds, which aren't shown on the clock's display.

### Downloading the Mobile App

You have decoded enough now from the Bluetooth logs to be able to send the right commands to the clock to set its time, but there are still some questions. I'll decompile the Android app ThermoPro Sensor and try to figure out those last unknown bytes.

You first have to download the APK file for the Android app. If you're still connected through USB and have enabled the debug connection, you can get the package using `adb`. First list the paths of the available packages:

```
adb shell pm list packages -f
```

Then search for the app's name in the resulting list. Unfortunately, ThermoPro isn't listed in the output. A search on the Google Play website and a look at the URL of the app's web page shows that the package has the name `com.ihunuo`.

`ykr_hn_2005a_t1w66`. So the following line in the package list is the app's package:

```
package:/data/app/~~9-mLh7bIZiKuB28ZkE7
I3EW==/com.ihunuo.ykr_hn_2005a_t1w66-07
iLaTUvHdWrAhZ51gxUzUQ==/base.apk=com.ihunuo.ykr_hn_2005a_t1w66
```

Now copy the APK file from the phone to your computer:

```
adb pull /data/app/~~9-mLh7bIZiKuB28ZkE7
EI3EW==/com.ihunuo.ykr_hn_2005a_t1w66-07
0iLaTUvHdWrAhZ51gxUzUQ==/base.apk
```

and use the path after `package:` and before the last `=` character. The result is now saved on your computer in `base.apk`.

If getting the APK file this way doesn't work, you can also download it from a third-party APK downloader site, such as APKPure [4].

### Decompiling the Mobile App

This APK file contains the Dalvik bytecode that Android executes, but you can decompile it into Java source code using the powerful `jadx` [5] decompiler, which comes in a command-line version as well as in a graphical interface. You will need the 64-bit version of Java 8 or later to run `jadx`.

Download the latest release of `jadx` from GitHub and unpack the ZIP file. Then start the graphical interface by

running `./bin/jadx-gui` from the command line inside the unpacked directory.

Select the `base.apk` file you downloaded. The program now decompiles the app and shows a tree structure of its packages and Java files at the left.

### Follow the Breadcrumbs

You are now ready to search for the time setting command. Look again at the Bluetooth log in Wireshark. You already figured out the meaning of the written value in the packet that sets the time, but where's it written? Click on the *Handle* drop-down in the details pane. It shows a service UUID, `000102030405060708090a0b0c0d1910`, and a characteristic UUID, `000102030405060708090a0b0c0d2b11`. These two values are the first breadcrumbs to follow.

In `jadx`, open the *Navigation | Text search* menu. Be patient: `jadx` now starts decompiling the whole app. When it's ready, paste one of the UUIDs in the search field. This won't result in a match for both UUIDs, but if you search with just `0001`, you see that the UUIDs are definitely in the code as a string with delimiters.

When reverse engineering BLE connections, the characteristics are always more interesting than the services, because those are the actual containers for data. Services are just lists of characteristics. If you enter the characteristic

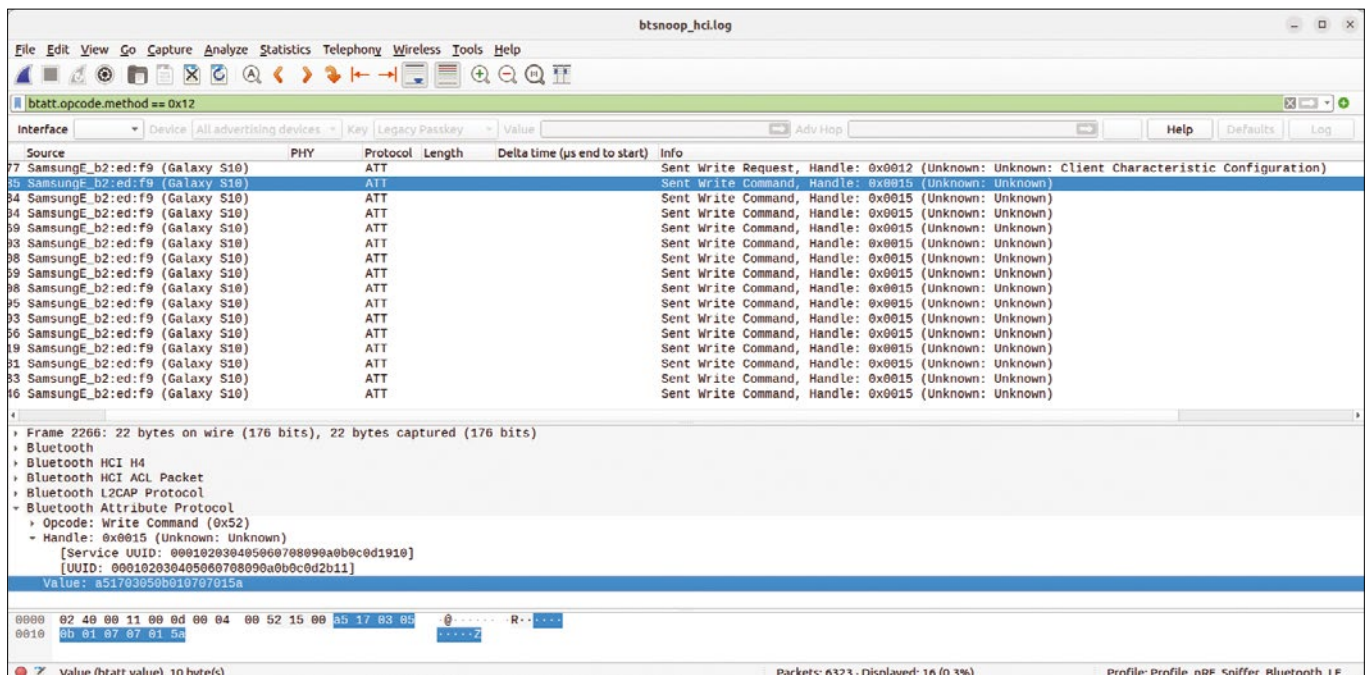


Figure 3: Determine the meaning of the commands your phone sends to the BLE clock.



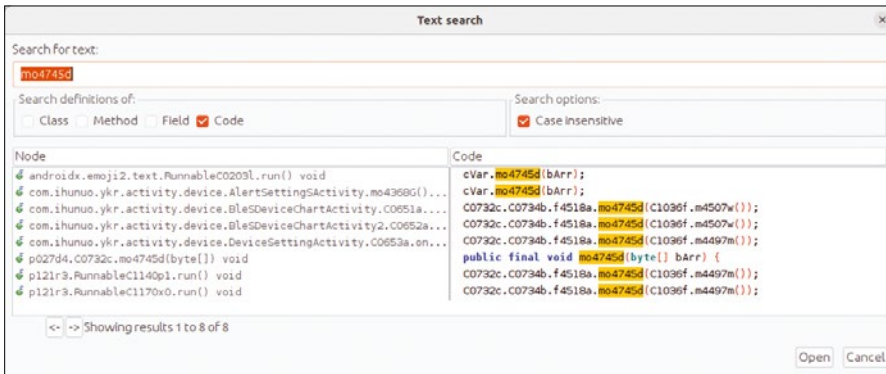


Figure 4: Following breadcrumbs in the decompiled code.

UUID as `00010203-0405-0607-0809-0a0b0c0d2b11` in the search field, you find one occurrence:

```
public final UUID f4476c = UUID.  
fromString("00010203-0405-0607-0809-  
0a0b0c0d2b11");
```

This means that code that uses this UUID refers to it by its name `f4476c`. Open the text search menu again and search for `f4476c`. There's only one other occurrence:

```
BluetoothGattCharacteristic  
characteristic =  
service.getCharacteristic(  
aVar.f4476c);
```

A few lines later this characteristic is used as follows:

```
characteristic.setValue(bArr);
```

Here, the value in `bArr` is written to the characteristic. This `bArr` is an argument to the method where this piece of code comes from, and this method is called `mo4745d`. So, that's the next breadcrumb. Search for this name in the text search menu to see where this method is called. This gives eight results (Figure 4), so look at each result and try to figure out what they're doing.

## Finding the Right Code

After following all the breadcrumbs, you find the code in Listing 1, which immediately shows the structure that you had figured out: a header byte, the year, month, day, hour, and then minute, which confirms that after this comes the seconds. Then comes the day of the week, and the `l` that you couldn't identify turns out to be a

value for the 24-hour format. My clock was showing the time on its display in 24-hour format, but apparently this bit changes the format to a 12-hour clock using AM/PM.

You also see that this code supports another time setting command structure, without the day of the week or 24-hour format. This presumably is for another type of clock without those capabilities.

## Finding Clocks

You now have a pretty good idea of the command format to send to the Bluetooth clock in order to set its time. I'll put this all together and create a Python script to do this without the app. You can do this with Bleak [6], a cross-platform BLE client library for Python. Install it with:

```
pip3 install bleak
```

First, you scan for all BLE devices in the vicinity and show their names and addresses. The script `find_devices.py` (Listing 2)

runs the main function asynchronously (because Bleak uses `asyncio`). This function creates a `BleakScanner` object, with `device_found` as a callback function that's called when a BLE advertisement is found. Then the script starts the scanner, waits five seconds, and stops the scanner. The `device_found` function just adds the device to a set of all found devices. After the scan is complete, the script shows the names and addresses of the devices in the set.

If you run this Python script with python `find_devices.py`, after five seconds it shows the names and addresses of all the BLE devices in your vicinity. For example, my clock is found with the name `TP358 (52C6)`, where `52C6` is the last four hexadecimal digits of the clock's address, so you now know you can search

## Listing 1: Decompiled Code for Setting the Time

```
01 public void mo4809a(boolean z) {  
02     byte[] bArr;  
03     if (mo4825v()) {  
04         if (z) {  
05             bArr = new byte[10];  
06             Time time = new Time();  
07             time.setToNow();  
08             bArr[0] = -91;  
09             bArr[1] = (byte) (time.year % 2000);  
10             bArr[2] = (byte) (time.month + 1);  
11             bArr[3] = (byte) time.monthDay;  
12             bArr[4] = (byte) time.hour;  
13             PrintStream printStream = System.out;  
14             printStream.println(DateFormat.is24HourFormat(this)  
15                 + "24 Hour" + time.hashCode());  
16             bArr[5] = (byte) time.minute;  
17             bArr[6] = (byte) time.second;  
18             int i = time.weekDay;  
19             bArr[7] = i == 0 ? 7 : (byte) i;  
20             bArr[8] = DateFormat.is24HourFormat(this);  
21             bArr[9] = 90;  
22         } else {  
23             Time time2 = new Time();  
24             time2.setToNow();  
25             bArr = new byte[]{-91, (byte) (time2.year % 2000),  
26                 (byte) (time2.month + 1), (byte) time2.monthDay,  
27                 (byte) time2.hour, (byte) time2.minute, (byte)  
28                 time2.second, 90};  
29         }  
30     }  
31     if (mo4826w() != null) {  
32         mo4745d(bArr);  
33     }  
34 }
```

Listing 2: find\_devices.py

```

01 import asyncio
02
03 from bleak import BleakScanner
04 from bleak.backends.device import BLEDevice
05 from bleak.backends.scanner import AdvertisementData
06
07
08 def device_found(device: BLEDevice, advertisement_data:
    AdvertisementData):
09     devices.add(device)
10
11
12 async def main():
13     scanner = BleakScanner(device_found)
14
15     print("Scanning for devices...")
16     await scanner.start()
17     await asyncio.sleep(5.0)
18     await scanner.stop()
19
20     for device in devices:
21         print(f"{device.name} | {device.address}")
22
23
24 if __name__ == "__main__":
25     devices = set()
26     asyncio.run(main())

```

for this type of Bluetooth clock by filtering on the name starting with TP358. You only have to change the callback function to get the script `find_clocks.py`:

```

def device_found(device: BLEDevice,
advertisement_data: AdvertisementData):
    if device.name.startswith("TP358"):
        devices.add(device)

```

## Sending the Time

Now that you know your clock's Bluetooth address, you can write another script, `synchronize_clock.py` (Listing 3), which receives this address as a command-line argument and then sets the time of the clock.

With your previous knowledge, this code is fairly straightforward. The script defines the characteristic to send the command and then defines a function, `get_bytes_from_time`, to convert a Unix timestamp to the bytes that need to be sent to the clock to set its time. The main function just connects to the address set on the command line, computes the correct bytes from the current time, and writes them to the right characteristic.

If you now run this command with your clock's Bluetooth address as an argument, for example,

```

python synchronize_clock.py E7:2E:00:B1:38:96

```

it should set your clock's time.

Listing 3: synchronize\_clock.py

```

01 import asyncio
02 from datetime import datetime
03 import sys
04 from time import time
05
06 from bleak import BleakClient
07
08
09 COMMAND_CHAR = "00010203-0405-0607-0809-0a0b0c0d2b11"
10
11
12 def get_bytes_from_time(timestamp: float) -> bytes:
13     """Generate the bytes to set the time on a ThermoPro
    TP358.
14
15     Args:
16         timestamp (float): The time encoded as a Unix
            timestamp.
17
18     Returns:
19         bytes: The bytes needed to set the time of the device
            to `timestamp`.
20     """
21     date_time = datetime.fromtimestamp(timestamp)
22     return bytes(
23         [
24             0xA5,
25             date_time.year % 2000,
26             date_time.month,
27             date_time.day,
28             date_time.hour,
29             date_time.minute,
30             date_time.second,
31             date_time.weekday() + 1, # Monday-Sunday -> 0-6
32             1, # Use 24-hour format
33             0x5A,
34         ]
35     )
36
37
38 async def main(address: str):
39     print(f"Connecting to {address}...")
40
41     async with BleakClient(address) as client:
42         command = get_bytes_from_time(time())
43         await client.write_gatt_char(COMMAND_CHAR, command)
44
45     print("Time synchronized")
46
47
48 if __name__ == "__main__":
49     try:
50         asyncio.run(main(sys.argv[1]))
51     except IndexError:
52         print("Please provide the Bluetooth address of the
            device")

```

**Listing 4: Discovering Bluetooth Clocks**

```
01 $ bluetooth-clocks discover
02 Scanning for supported clocks...
03 Found a ThermoPro TP358: address BC:C7:DA:6A:52:C6, name TP358 (52C6)
04 Found a Xiaomi LYWSD02: address E7:2E:00:B1:38:96, name LYWSD02
05 Found a ThermoPro TP393: address 10:76:36:14:2A:3D, name TP393 (2A3D)
06 Found a Qingping BT Clock Lite: address 58:2D:34:54:2D:2C, name Qingping BT
    Clock Lite
07 Found a Current Time Service: address EB:76:55:B9:56:18, name F15
```

**Listing 5: Setting a Bluetooth Clock's Time**

```
01 $ bluetooth-clocks set -a E7:2E:00:B1:38:96
02 Scanning for device E7:2E:00:B1:38:96...
03 Writing time to device...
04 Synchronized time
```

my Bluetooth clocks, and I published the result, *bluetooth-clocks*, on the Python Package Index (PyPI) [7].

After installing the package with

```
pip3 install bluetooth-clocks
```

you can discover all your supported clocks as shown in Listing 4. Then you can set the clock's time with a given Bluetooth address as shown in Listing 5. If you want to regularly synchronize the time on the device, you can run Listing 5 as a service, for example, with a systemd timer or in a cron job. See Figure 5 for the result.

**Further Work**

This simple script is just the beginning. You can make it much more robust by catching some exceptions, such as connection problems. You also can add other arguments, such as the choice of the 24- or 12-hour format. This is just for one type of clock. You can add support for other types of Bluetooth clocks. I did this for all of

If you have a Bluetooth clock at home that isn't supported yet by *bluetooth-clocks*, please try the reverse engineering approach demonstrated here and contribute your knowledge to the project.

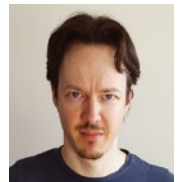
Finally, if you want to learn more about BLE and how to use this wireless protocol in your own programs, read my book on the topic [8]. ■■■

**Info**

- [1] Home Assistant: <https://www.home-assistant.io>
- [2] Wireshark: <https://www.wireshark.org>
- [3] Android SDK Platform Tools: <https://developer.android.com/studio/releases/platform-tools>
- [4] APKPure: <https://apkpure.com>
- [5] jadx: <https://github.com/skylot/jadx>
- [6] Bleak: <https://bleak.readthedocs.io>
- [7] bluetooth-clocks: <https://github.com/koenvervoesem/bluetooth-clocks>
- [8] Vervloesem, Koen. *Develop your own Bluetooth Low Energy Applications*. Elektor, <https://www.elektor.com/develop-your-own-bluetooth-low-energy-applications>

**Author**

**Koen Vervloesem** has been writing about Linux and open source, computer security, privacy, programming, artificial intelligence, and the Internet of Things for more than 20 years. You can find more on his website at [koen.vervoesem.eu](https://koen.vervoesem.eu).



**Figure 5:** You can now keep all your clocks synchronized.

Control your backup NAS from the desktop

# Magic Cargo

To be able to power up and shut down his NAS and check the current status without getting out of his chair, Mike Schilli programs a graphical interface that sends a Magic Packet in this month's column. *By Mike Schilli*



As a backup solution, I use a Synology NAS with some hefty hard drives. But because of the strict regulations driven by paranoia and noise-protection goals in the hallowed halls of Perlmeister Studios, the device only runs when actually needed (i.e., when a backup is running). To switch it

## Author

Mike Schilli works as a software engineer in the San Francisco Bay Area, California. Each month in his column, which has been running since 1997, he researches practical applications of various programming languages. If you email him at [mschilli@perlmeister.com](mailto:mschilli@perlmeister.com) he will gladly answer any questions.



on when needed, I prefer to push a mouse around a GUI and click occasionally instead of getting out of my chair.

The Syno desktop application presented here (Figure 1) powers up the NAS via the local network at the push of a button and graphically displays the milestones during the boot process with progress bars. As soon as the system has finished booting and is ready for access, it notifies the user.

After the work is done, a mouse click on the GUI's *Down* button is all it takes, and the NAS receives the command to shut down via the network. While the shutdown is running, the GUI checks whether the NAS is still operational or if it is no longer responding to pings and has finally gone to sleep (Figure 2).

## Just Switch It On

How does this magic work? Even when switched off, the NAS actively waits for a Wake-on-LAN (WoL) signal on the local network. Despite its control lights being off, the network card is running in a low power mode. If a matching broadcast packet arrives via the LAN, it tells the device's power supply or motherboard to switch on. The Magic Packet sent in this case contains the target system's MAC address, so that a controlling transmitter can notify different network nodes independently.

Figure 3 shows a practical example of the Magic Packet's format [1]. The first 6 bytes of the packet header each contain a fixed value, 0xFF. This is followed

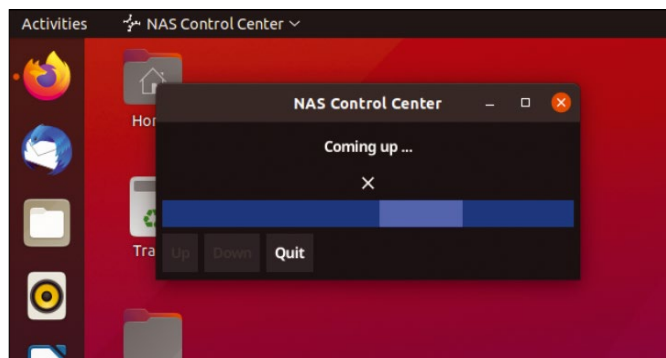


Figure 1: Control via desktop app: Mike's Synology NAS booting.

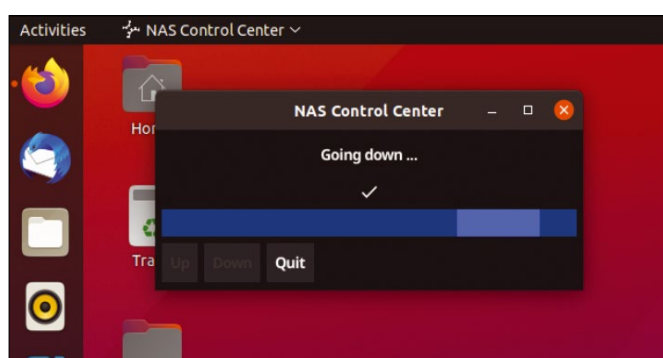


Figure 2: At the push of a button, the network storage device then shuts down again.

Lead Image © alphaspirt, 123RF.com

```

$ ./magic | hexdump -C
00000000 ff ff ff ff ff ff 00 11 32 6c ab cd 00 11 32 6c |.....2l....2l|
00000010 ab cd 00 11 32 6c ab cd 00 11 32 6c ab cd 00 11 |....2l....2l....|
00000020 32 6c ab cd 00 11 32 6c ab cd 00 11 32 6c ab cd |2l....2l....2l..|
00000030 00 11 32 6c ab cd 00 11 32 6c ab cd 00 11 32 6c |..2l....2l....2l|
00000040 ab cd 00 11 32 6c ab cd 00 11 32 6c ab cd 00 11 |....2l....2l....|
00000050 32 6c ab cd 00 11 32 6c ab cd 00 11 32 6c ab cd |2l....2l....2l..|
00000060 00 11 32 6c ab cd |..2l..|
00000066

```

Figure 3: A Magic Packet's hexdump for the MAC address 00:11:32:6c:ab:cd.

by the payload, which consists of 16 repetitions of the 6-byte MAC address of the target device (00:11:32:6c:ab:cd in this example). Each network card has its own setting, which specifies the

device's manufacturer, model, and individual identifier.

Listing 1 cobbles together the Magic Packet in Go. Line 7 sets the NAS's MAC address as a string, while line 8 specifies

### Listing 1: wol.go

```

01 package main
02 import (
03     "bytes"
04     "encoding/binary"
05     "net"
06 )
07 const synMAC = "00:11:32:6c:ab:cd"
08 const MacLen = 6
09 type MagicPacket struct {
10     header [6]byte
11     payload [16][MacLen]byte
12 }
13 func sendMagicPacket() {
14     var packet MagicPacket
15     hwAddr, err := net.ParseMAC(synMAC)
16     if err != nil {
17         panic(err)
18     }
19     for idx := range packet.header {
20         packet.header[idx] = 0xFF
21     }
22     for idx := range packet.payload {
23         for i := 0; i < MacLen; i++ {
24             packet.payload[idx][i] = hwAddr[i]
25         }
26     }
27     buf := new(bytes.Buffer)
28     if err := binary.Write(buf, binary.BigEndian, packet); err != nil {
29         panic(err)
30     }
31     conn, err := net.Dial("udp", "255.255.255.255:9")
32     if err != nil {
33         panic(err)
34     }
35     defer conn.Close()
36     _, err = conn.Write(buf.Bytes())
37     if err != nil {
38         panic(err)
39     }
40 }

```

that its length really is 6 bytes. In rare cases, there could be devices with longer MAC addresses, but their Magic Packets are more difficult to build. For the purpose of illustration, I'll keep things short and sweet.

The MagicPacket structure starting in line 9 abstracts the packet's two different areas: the header and payload. The sendMagicPacket() function starting in line 13 then wraps the packet and, when done, sends it to the broadcast address 255.255.255.255 on the LAN. This means that all devices on the local network get to see the packet and can react accordingly. If a device listening via WoL receives a packet in which the packaged MAC address matches its own, it initiates the boot process.

The ParseMAC() function from the standard net Go library treasure trove translates the MAC string from line 7 into a binary hardware address, which the library's network functions need in order to send off the packet. The packet header with 6 0xFF bytes is assembled by the for loop starting in line 19. The subsequent double loop starting in line 22 then writes the MAC address in binary format 16 times in succession to the packet's payload area.

To convert the Go structure of the MagicPacket type to a binary stream of bytes for packet recipients listening on the network, the standard binary.Write() function in line 28 trawls the structure of the packet variable. To do this, it writes the bytes of the structure in network format (big-endian, most significant byte first) to the buf buffer. Line 36 then uses Write() to send the buffer content via the UDP socket opened in line 31 by net.Dial() to the LAN's broadcast address.

Be careful: binary.Write() can only serialize a structure without error if all of its fields are of a fixed length. The function does not support Go's dynamically expandable array slices. You will see some really ugly runtime errors if you proceed without heeding this warning.

## Panic

The states the application can be in at runtime are those of a simple finite machine. After starting the program, the NAS is usually asleep (state *DOWN*). The user issues the wake-up command

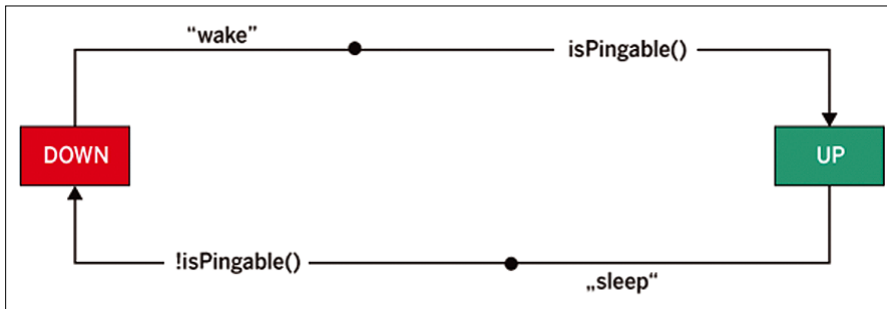


Figure 4: The states and transitions of the simple finite machine.

by pressing the *Up* button. Following the bootstrap, the application keeps checking if the NAS can be pinged yet. If it shows no response, it is probably still asleep (i.e., resting in the *DOWN* state). But as soon as the ping command reports success, the NAS is ready for operation and the finite machine jumps to the *UP* state.

Figure 4 shows the state machine diagram, while Listing 2 contains the implementation using the Go *fsm* library from GitHub. The `NewFSM` function creates a new finite state machine starting in line 8 that processes two events: `wake` (line 11), which transitions from the *DOWN* state to the *UP* state, and `sleep` (line 12), which switches the machine from *UP* to *DOWN*. Listing 2 defines the conditions for these transitions in the `enter_UP` and `enter_DOWN` callbacks; the machine jumps to each of these before actually starting a transition.

Each of these two callbacks erects a hurdle that the program flow needs to

clear before entering the new state. An infinite for loop keeps running `isPingable()` to check whether the desired state has already been reached, in which the NAS is either running or asleep, depending on the callback. If not, the callbacks wait a second and then try again. This is repeated until the desired state is reached.

Then the callbacks send a message with the new state of the machine on the `stateReporter` channel provided by the caller earlier. The `run()` function's final act is to return a reference for the ready-to-go state machine to the caller. With this handy tool at its disposal, the caller will be

sending new commands to the machine using methods such as `Event()` to initiate the associated state transitions.

Interestingly, the third-party library used for the *fsm* library from GitHub uses strings instead of typed variables for its states. This is not recommended Go style, because it means that a simple typo in a state in the code is enough to trigger a frantic search for runtime errors. This way, the type checker in the Go compiler has no way to detect the bug at compile time. The library clearly still has room for improvement, but you can't look a gift horse in the mouth.

## On Screen!

Listing 3 has all the code to draw the small GUI shown in the screenshots. It is based on the Fyne framework, which the code pulls in from GitHub at compile time.

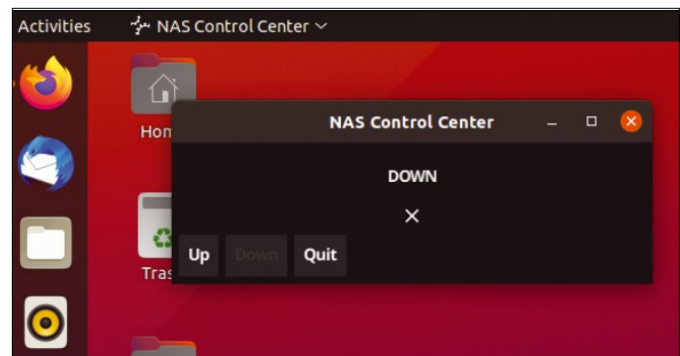


Figure 5: Initially, the *Down* button is inactive.

## Listing 2: fsm.go

```

01 package main
02 import (
03     "context"
04     "time"
05     "github.com/looplab/fsm"
06 )
07 func run(stateReporter chan string, startState string)
08     *fsm.FSM {
09     boot := fsm.NewFSM(
10         startState,
11         fsm.Events{
12             {Name: "wake", Src: []string{"DOWN"}, Dst: "UP"},
13             {Name: "sleep", Src: []string{"UP"}, Dst: "DOWN"},
14         },
15         fsm.Callbacks{
16             "enter_UP": func(ctx context.Context, e *fsm.Event) {
17                 for {
18                     if isPingable() {
19                         stateReporter <- "UP"
20                     }
21                     time.Sleep(1 * time.Second)
22                 }
23             },
24             "enter_DOWN": func(_ context.Context, e *fsm.Event) {
25                 for {
26                     if !isPingable() {
27                         stateReporter <- "DOWN"
28                         return
29                     }
30                     time.Sleep(1 * time.Second)
31                 }
32             },
33         },
34     )
35     return boot
36 }
  
```

The app features an application window with a label widget that displays the NAS status (*UP* or *DOWN*). In addition, there is an icon (check mark if the NAS is operational; X if not) and three buttons for user control. A progress bar also appears during the transition

phases. After starting the program, initially only the *Up* button is active, while the *Down* button is grayed out (Figure 5). If you press a button to initiate an action, the app grays all buttons to prevent further impatient clicking triggering confusing actions.

The application causes some parts of the GUI to change dynamically with the program flow by making certain widgets disappear in some situations (the `Hide()` function is used for this) and reappear later on using `Show()`. The NAS status icon – either a check mark or an X – actually

### Listing 3: syno.go

```

01 package main
02 import (
03     "context"
04     "fyne.io/fyne/v2"
05     "fyne.io/fyne/v2/app"
06     "fyne.io/fyne/v2/canvas"
07     "fyne.io/fyne/v2/container"
08     "fyne.io/fyne/v2/theme"
09     "fyne.io/fyne/v2/widget"
10     "os"
11 )
12 func main() {
13     state := "DOWN"
14     headText := "NAS Control Center"
15     a := app.New()
16     w := a.NewWindow(headText)
17     status := widget.NewLabelWithStyle(state, fyne.
18         TextAlignCenter, fyne.TextStyle{Bold: true})
19     progress := widget.NewProgressBarInfinite()
20     progress.Stop()
21     progress.Hide()
22     okIcon := widget.NewIcon(theme.ConfirmIcon())
23     okIcon.Hide()
24     downIcon := widget.NewIcon(theme.CancelIcon())
25     stateReporter := make(chan string)
26     runner := run(stateReporter, state)
27     var upButton *widget.Button
28     var downButton *widget.Button
29     upButton = widget.NewButton("Up", func() {
30         upButton.Disable()
31         downButton.Disable()
32         status.Text = "Coming up ..."
33         status.Refresh()
34         sendMagicPacket()
35         progress.Show()
36         go func() {
37             runner.Event(context.Background(), "wake")
38         }()
39     })
40     downButton = widget.NewButton("Down", func() {
41         upButton.Disable()
42         downButton.Disable()
43         status.Text = "Going down ..."
44         status.Refresh()
45         progress.Show()
46         shutdownNAS()
47         go func() {
48             runner.Event(context.Background(), "sleep")
49         }()
50     })
51     downButton.Disable()
52     go func() {
53         for {
54             select {
55                 case newState := <-stateReporter:
56                     progress.Hide()
57                     switch newState {
58                         case "DOWN":
59                             okIcon.Hide()
60                             downIcon.Show()
61                             upButton.Enable()
62                         case "UP":
63                             okIcon.Show()
64                             downIcon.Hide()
65                             downButton.Enable()
66                     }
67                     status.Text = newState
68                     status.Refresh()
69                 }
70             }()
71             img := canvas.NewImageFromResource(nil)
72             img.SetMinSize(
73                 fyne.NewSize(400, 0))
74             grid := container.NewVBox(
75                 img,
76                 status,
77                 okIcon,
78                 downIcon,
79                 progress,
80                 container.NewHBox(
81                     upButton,
82                     downButton,
83                     widget.NewButton("Quit", func() {
84                         os.Exit(0)
85                     })),
86                 )
87             w.SetContent(grid)
88             w.ShowAndRun()
89         }
90     }

```

**Listing 4: util.go**

```

01 package main
02 import (
03     "os/exec"
04 )
05 const synIP = "192.168.3.33"
06 func shutdownNAS() {
07     cmd := exec.Command("ssh", "synuser@"+synIP, "sudo", "/sbin/poweroff")
08     if err := cmd.Run(); err != nil {
09         panic(err)
10     }
11 }
12 func isPingable() bool {
13     cmd := exec.Command("ping", "-c", "1", "-t", "3", synIP)
14     if err := cmd.Run(); err != nil {
15         return false
16     }
17     return true
18 }

```

consists of two separate widgets `okIcon` and `downIcon`, but the app only displays one of them at any given time and keeps the other one tucked away invisibly.

The infinite progress bar is also always present in the application window. However, it is only visible and moving if an action is currently running (e.g., in the callback of the `Up` button starting in line 28). After line 33 calls `sendMagicPacket()` to send the packet that starts the NAS via the network, line 34 calls `progress.Show()` to display the progress bar. If the state of the machine changes, line 55 uses `Hide()` to make the progress bar visually disappear from the app again.

**Wake Up on Command**

If the user clicks the `Up` button, the callback notifies the state machine in line 36, which uses its transition rules to determine the next steps of the application. Line 36 sends the `wake` event to the machine with the `Event()` function. The machine then blocks until the NAS wakes up, before finally sending the `UP` message on the `stateReporter` channel. Because I don't want the GUI to freeze while the machine is blocking, the callback wraps the call to the machine's `Event()` function in a Go routine that continues to run concurrently while the callback completes.

Events from channel `stateReporter` are intercepted by a Go routine running concurrently starting in line 51. It uses a

`select` statement to listen on the channel. As soon as the state machine announces a new state, the code jumps to one of the two case blocks handling the up or down states. These in turn prompt the GUI to refresh the display to reflect the new state.

The new container generated starting in line 74 lines up all the widgets, both visible and invisible. `NewHBox()` in line 80 defines the sub-container at the bottom of the main GUI container, and the three buttons for controlling the app are arranged horizontally next to each other at the bottom of the application window. Meanwhile, the main container uses

**Listing 5: Building the Application**

```

$ go mod init syno
$ go mod tidy
$ go build

```

`NewVBox()` to stack the rest of the widgets on top of each other; they include the text and icon for the state, the progress bar, and the sub-container.

Line 88 dumps everything into the application window. Finally, line 89 jumps to the endless GUI loop by calling `ShowAndRun()`. From there, the GUI intercepts mouse input from the user and displays the GUI changes initiated by the program code running smoothly and concurrently.

**Not Below 400 Pixels**

The Fyne framework not only runs on desktop interfaces, but it was designed to work on mobile devices as well. This is why, among other things that look awkward at first, it does not provide an option for setting an application window to a defined minimum size after the program start.

That's a good strategy in the grand scheme of cross-platform framework development. However it's quite jarring if an application like Syno comes up as a mini window measuring 50x50 pixels that you can hardly see on the desktop. Fortunately, you can use a trick to define a minimum size. The app stuffs the main container with an empty image measuring 400x0 pixels, which invisibly



**Figure 6:** The Go program also looks smashing on a MacBook.



becomes part of the VBox with the widgets. This forces the renderer to open an application window with a width of at least 400 pixels wide from the outset.

### Switching Off with sudo

The last part of the application in Listing 4 defines the utility function `isPingable()`, which checks whether the NAS is operational. This could of course be done with a component from the `net` standard library from the Go repository, but for simplicity's sake the function simply calls the Ping program in the shell. Depending on that command's return code, the function returns a true or false value to the caller.

To shut down the NAS when the user presses the *Down* button, the `shutdownNAS()` function contacts the NAS at its IP address starting in line 6 and logs into a predefined SSH account. It then issues the `sudo poweroff` command in the opened shell. This causes the monster disk array to shut down and disconnect

itself from the power supply. Meanwhile, the state machine notices this in the course of its continuous checks and the GUI jumps to the *DOWN* visualization.

To do this, the controlling host needs to be able to use SSH to log in to the NAS without entering a password. This is typically achieved by storing the public key of a key pair maintained by the controlling host in the `~/.ssh/authorized_keys` file on the NAS. Also, the user on the NAS needs `sudo` privileges to execute the `poweroff` command. This is handled by the line

```
synouser ALL=(ALL) NOPASSWD: ✓
/sbin/poweroff
```

in `/etc/sudoers` file on the NAS.

### Not Only for Linux

All that remains is to build the application using the commands from Listing 5. The first two `mod` commands fetch libraries used in the code from GitHub. The

`build` command links everything together into a `syno` binary that can be copied and executed anywhere with a similar operating system environment. The IP and MAC addresses used in the code, and the SSH user on the NAS, need to be adapted to match the local conditions.

One benefit of the Fyne framework is its platform independence. The program can also be built on other operating systems and desktop environments. Figure 6 shows the application running on a MacBook, but it could be used on Windows, too. Fyne even has tools for Android to help you generate the application in line with the operating system's specifications [2]. This framework is a genuine cross-platform jack-of-all-trades! ■■■

### Info

- [1] WoL: <https://en.wikipedia.org/wiki/Wake-on-LAN>
- [2] Fyne mobile packaging: <https://developer.fyne.io/started/mobile>

# What?!

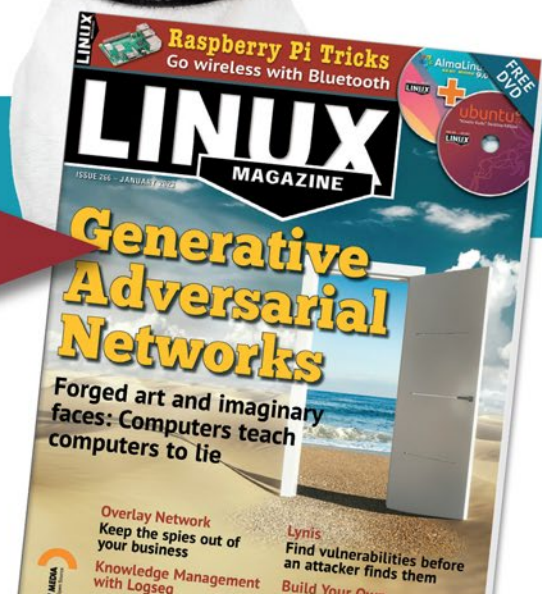
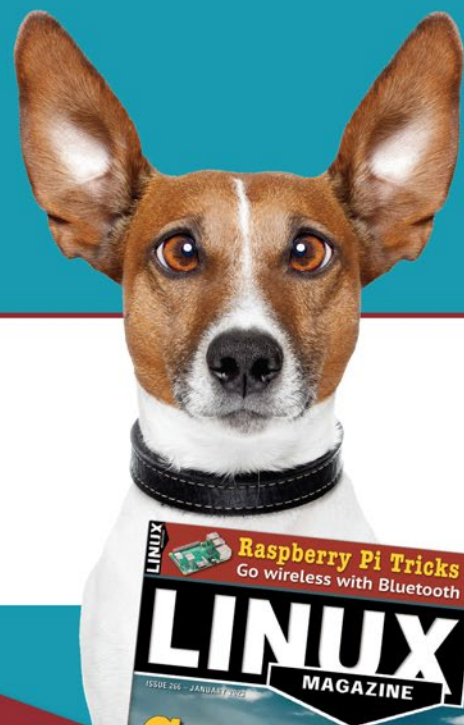
I can get my issues SOONER?



Available anywhere, anytime!

Sign up for a digital subscription and enjoy the latest articles on trending topics, reviews, cool projects and more...

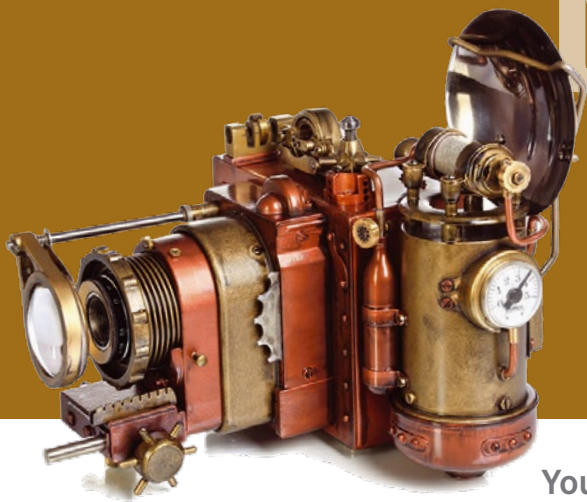
Subscribe to the PDF edition:  
[bit.ly/Linux-Mag-digital](http://bit.ly/Linux-Mag-digital)



# MakerSpace

Make a camera for  
lenticular photography

## Wiggle Time



You can take lenticular images with a homemade camera to re-create the “wiggle” pictures of your childhood.

By Günter Pomaska

**L**enticular images store multiple exposures in the same area. Animation is achieved by tilting the image. Another application creates a spatial appearance without special tools (autostereoscopy). The digital version of this often shows up on social media as a “wigglegram.”

### Lenticular Cameras

On the consumer market, lenticular cameras are sold under the name ActionSampler. More than 40 years ago, the four-lens Nishika (Nimslo) appeared,

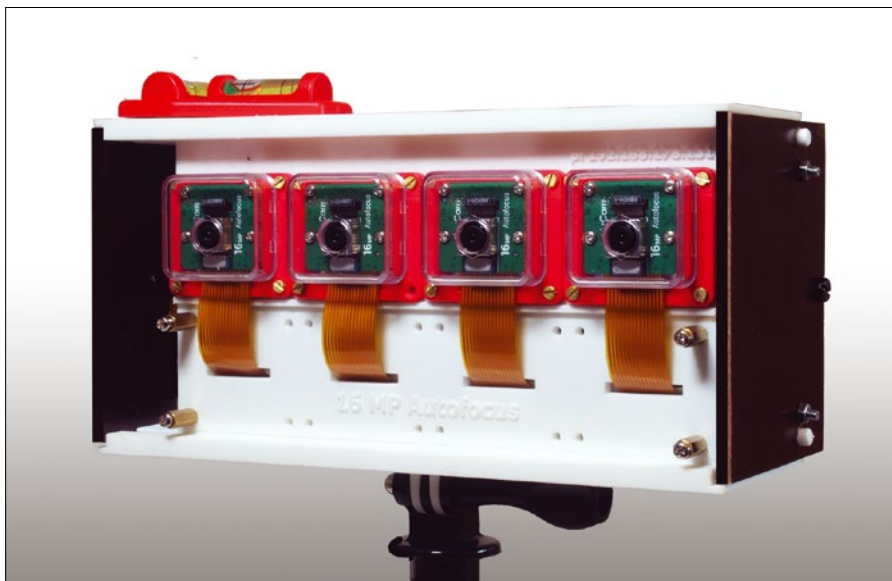
followed by Fuji’s eight-lens Rensha Cardia in 1991. Unlike the Nishika’s synchronous shutter action, the Fuji exposed the 35mm film sequentially. Even today, the analog scenes are still very popular on Instagram and the like.

One way of creating a multilens digital recording system is to use a Raspberry Pi and a Camarray HAT [1] (hardware attached on top) by ArduCam [2]. The camera I make in this article uses four Sony IMX519 sensors arranged at a distance of 4cm apart (Figure 1). After the first exposure, you can move the device by half the camera distance, which produces eight shots of a subject at equal distances with a total of 32 megapixels (MP).

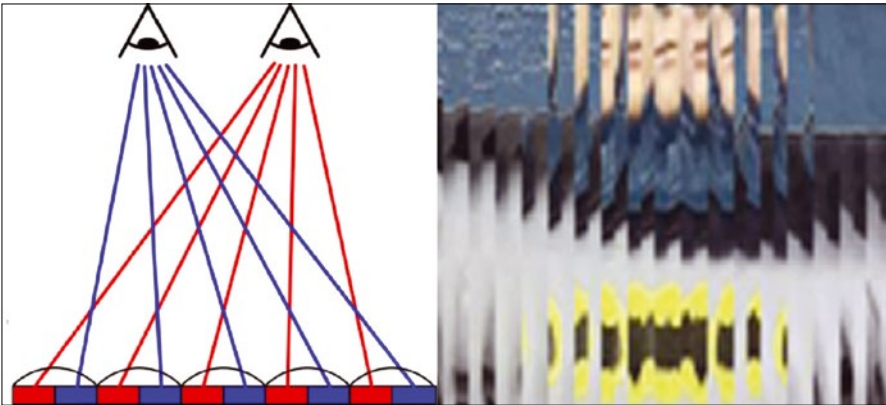
### Lenticular Technology

The predecessors of today’s lenticular screens are corrugated and lamellar screens that take two and three displayable images, respectively. Unlike the planar image strips of their predecessors, the lens screens commonly used today are transparent films of semi-cylindrical strips that show multiple images simultaneously [3]. Depending on the viewer’s angle of view, the left eye sees something different than the right eye, and the viewer perceives the view as three-dimensional (Figure 2).

The lenses differ in terms of thickness and curvature radius; resolution is stated in lines per inch (lpi). Changing the



**Figure 1:** With four cameras, you can take a total of eight images at the same distance by moving the camera a short distance.



**Figure 2:** Depending on the viewing angle, the left and right eyes see different images, and together perceive a three-dimensional image.

image, animation, and zooming and morphing effects can be achieved with image strips arranged horizontally. To separate spatial images you need vertical image strips; the input images are encoded strip by strip in line with the lens spacing and are printed on a self-adhesive foil or as mirror images on the reverse side of the foil.

You can achieve a spatial vision effect by nesting the individual images inside each other, which leads to image separation for the viewer. However, you do not need to restrict yourself to two images; instead, you can compile a series of images. To do this for static scenes, you move the camera step-by-step. Alternatively, you can use camera technology with multiple lenses, which is also how to capture dynamic scenes. StereoPhoto Maker [4] is freeware for preparing image series. If you want to look more closely into wigglegrams, it is a good idea to take a look at the Tri-axes 3DMasterKit [5] software.

### Four-Lens DIY Camera

As the control unit, I will add the Ardu-Cam Camarray HAT to a Raspberry Pi 4B. The Pivariety manufacturer makes extended solutions for Raspberry Pi standard cameras that act as Video for Linux version 2 (V4L2) devices. The HAT operates four Sony IMX519 sensors over a Camera Serial Interface (CSI), which you address on the I2C bus. The sensors have an image memory of 16MP, but depending on the addressing, one or more sensors share the image memory. The sensors can be operated in autofocus mode or with manual adjustment. The field of view is 80 degrees horizontally, and sharpness starts at about 8cm.

While you are on the move, a 5V power bank supplies the unit with energy. Three-dimensional printed components let you design a case. The camera boards sit side by side in the supplied brackets. Of course, you can't adjust a setup like this with single-pixel accuracy, but you don't need that because

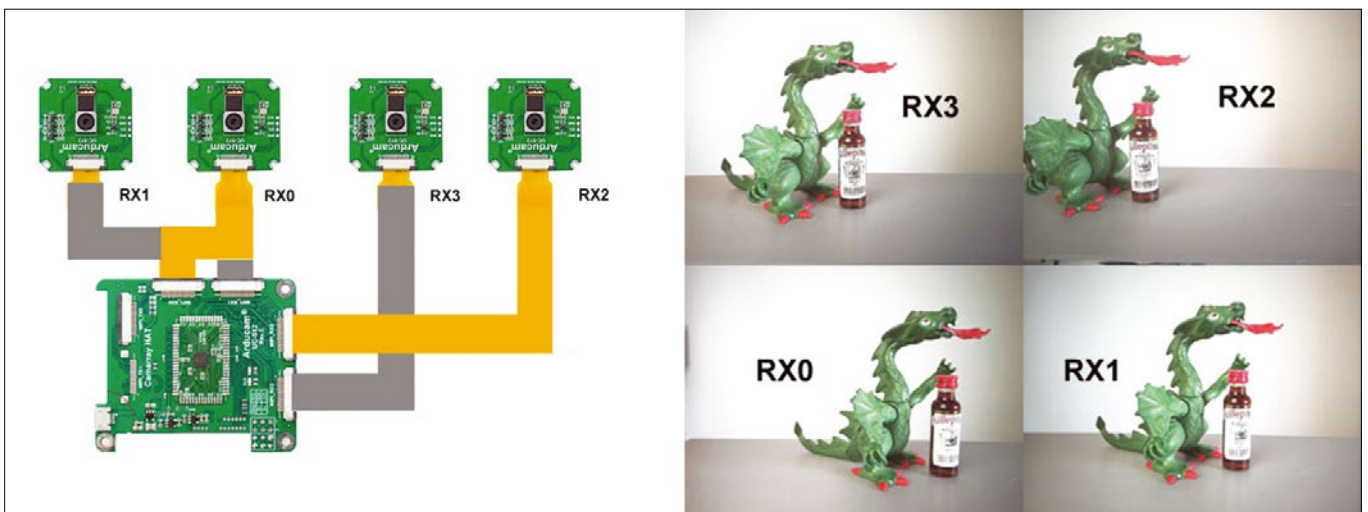
you can use the application software instead. The housing is designed so you can move the entire lens board by half the camera distance. In this way, the data for a lenticular image can be assembled from eight exposures, each 2cm apart over a base of 14cm.

The multicamera adapter is connected to the sensors by four interfaces that use ribbon cables. You then need to connect it to the computer on the CSI interface. Three spacer screws secure the mechanical connection to the Raspberry Pi; the 5V power supply comes through the GPIO pins. How you arrange the cameras is entirely up to you. The boards each come in a small case, and you install them 40mm apart. If you do not use the housings, the minimum distance is reduced to 24mm. The software addresses the sensors as a single frame. By setting the corresponding I2C parameters, you can configure one, two, or four sensors. The cameras always have to share the available resolution.

The default is

```
i2cset -y 10 0x24 0x00
```

(i.e., Quadro mode). Accordingly, the resolution for each image is restricted to a maximum of 2328x1746 pixels, and synchronization is in pairs at frame level. If you specify the `i2cset` parameter as above, the result is a resolution of two times 2328x3496 pixels in dual mode, which is extrapolated to two times 4656x3496 pixels later in the application. You may already be familiar with image compression from stereoscopy.



**Figure 3:** The close-ups on the right reflect the camera arrangement. The cabling diagram is shown on the left.

The images in Figure 3 on the right were taken from close up and therefore clearly reflect the camera layout and cabling (from left to right: RX2, RX3, RX1, and RX0). Despite the convenient autofocus mode, it is important not to forget the manual focus options. Especially at close range, manual focus results in some interesting photographic options (Figure 4).

## Installing the Camarray HAT

You can install the required applications and the driver for the quad kit with the shell script

```
install_pivariety_pkgs.sh
```

(Listing 1). More information is available in the ArduCam documentation.

After the `libcamera-hello` command, the camera will respond for a short while. The

```
libcamera-still --list-cameras
```

command (Listing 1, last command) checks which cameras are connected. As mentioned before, the software identifies the four sensors as a single device.

## Libcamera

The release of the Raspberry Pi OS “Bullseye” operating system in November 2021 fundamentally changed the handling of the camera module. Brand new *libcamera* commands have since replaced the tried and trusted command-line tools `raspi-still` and `raspi-vid`. You can still use `raspistill` in legacy mode, but makers with more ambitious goals need to get comfortable with the *libcamera* library.

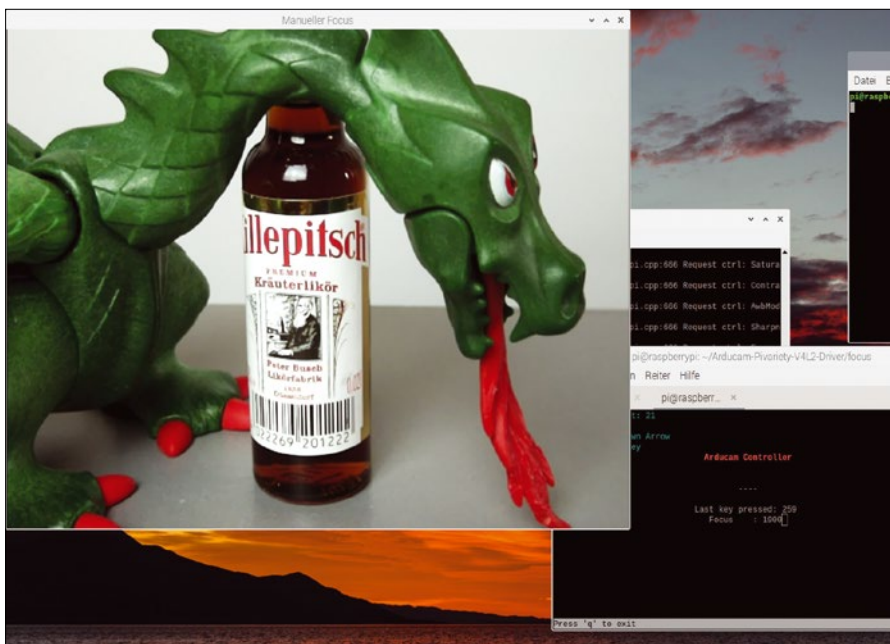
The transition of the camera control to the Linux kernel’s Libcamera driver ensures a standards-compliant solution

without proprietary code. New commands such as `libcamera-still` or `libcamera-vid` are available, and you can build your own apps on the Libcamera code. Extensive documentation can be found on the Raspberry Pi Foundation [6] website.

If you have already worked with `raspistill` or `raspi-vid`, it should not be difficult to come to grips quickly with Libcamera. The sample code

```
$ i2cset -y 10 0x24 0x24 0x00
$ libcamera-still -t 30000 --ev -5 --gain 8 --roi 0,0,1,1 --autofocus --info-text "Killepitsch" -o testQuadro.jpg
```

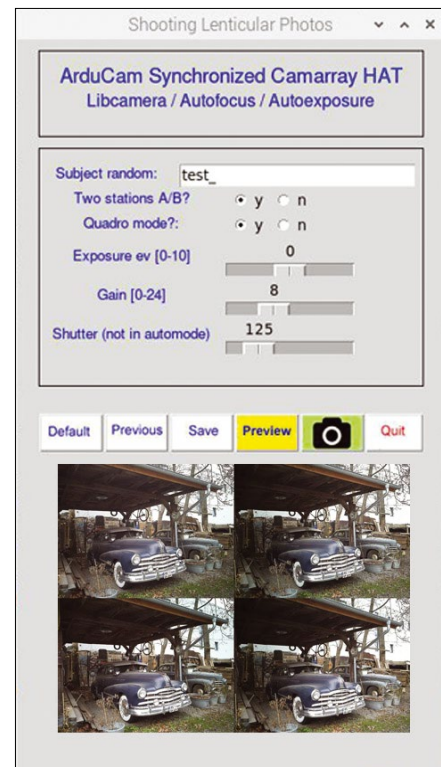
captures the entire image (region of interest, `--roi`) in autofocus mode after a preview time of 30 seconds (time out, `-t`) with an exposure compensation of -5 (exposure value, `--ev`). The `--gain 8` parameter corresponds to an ISO value of 800, and the `--info-text` flag lets you manipulate the header in the application; the output file is assigned the name `testQuadro.jpg` (output, `-o`).



**Figure 4:** By focusing manually, you can significantly increase your freedom as a photographer.

## Listing 1: Installing the Camarray HAT

```
$ wget -O install_pivariety_pkgs.sh https://github.com/ArduCAM/ArduCam-Pivariety-V4L2-Driver/releases/download/download/install_script/install_pivariety_pkgs.sh
$ chmod +x install_pivariety_pkgs.sh
$ sudo apt-update
$ ./install_pivariety_pkgs.sh -p libcamera_dev
$ ./install_pivariety_pkgs.sh -p libcamera_apps
$ ./install_pivariety_pkgs.sh -p imx519_kernel_driver_low_speed
[...]
$ libcamera-still --list-cameras
0 : imx519 [4656x3496] (/base/soc/i2c0mux/i2c@1/imx519@1a)
Modes: 'SRGB10_CSI2P' : 1280x720 1920x1080 2328x1748 3840x2160 4656x3496
```



**Figure 5:** You can set the basic values for the shot in a GUI.

## Shooting Lenticular Photos

The DIY camera is designed to be point-and-shoot, but the implementation is a little more modest because of the available technology. In my test environment, the system is connected to the local WiFi network behind a mobile router. After switching on the camera, the operating system boots and logs on to the WiFi network. The Virtual Network Computing (VNC) server starts up at boot time.

The same applies to the graphical user interface (GUI; Figure 5), with simple setup functions such as image name, shutter speed, exposure value, preview image, and shutter trigger. The GUI offers more functions, but I will not be using them for the time being. The

## Graphical User Interface

In the download section for this article you will find the `lentiCam.py` GUI [7], which I programmed in Python. The `dcim/` directory also contains some recent images as examples of the components of a lenticular image.

software, written in Python, uses Guizero with object-oriented controls. It keeps its settings in a dictionary and uses system commands to call the camera functions (see also the “Graphical User Interface” box).

The images for positions A and B and the individual image tiles end up in the `dcim/randomCode/` directory with randomized image labels. The GUI displays the generated random code, which can be changed alphanumerically if required. You decide in advance whether you want to create two, four, or eight exposures. The camera settings can be saved so that you can reuse them for later shots. To align the camera, click the *Preview* button; status messages are displayed in the header.

Finding a shooting scenario is now the problem. The camera is oriented horizontally. By rule of thumb, the distance to the object is 30 times the distance from the right to the left camera (close-up distance 1/30). Intermediate images split this distance evenly, as you can see in Figure 6, working with four sensors at a distance of 12cm with

two intermediate images. This results in a close-up distance of about 3.6m. If you move the lens board by 2cm, you end up with four shots at 8cm apart, and ideally approach the subject to within 2.4m.

These approximations are only rough, based on experience, and by no means binding. The close-up value could well be closer to 1/20 than 1/30. As soon as you move your camera between two positions, you are forced to limit your work to static objects. Manual exposure by shutter speed and gain settings is generally recommended.

After capturing an image, you then need to break the frame down into individual tiles for further processing with the help of the FFmpeg suite or ImageMagick tools, which you can install with:

```
sudo apt-get install imagemagick
sudo apt-get install ffmpeg
```

To ensure that the images are ordered correctly, add a numerical suffix in the file name.



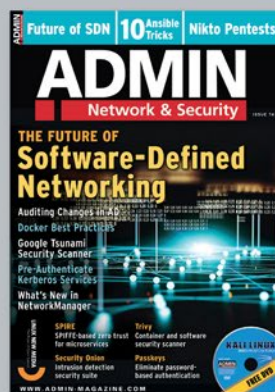
sparkhaus-shop.com

Discover the past and invest in a new year of IT solutions at Linux New Media's online store.

Want to subscribe?

Searching for that back issue you really wish you'd picked up at the newsstand?

DIGITAL & PRINT SUBSCRIPTIONS



SPECIAL EDITIONS



➤ sparkhaus-shop.com



## Making One Out of Many

Once you have done the field work, you can continue processing the image series on the computer. Download the images with an FTP client; then, StereoPhoto Maker (for example) will give you all the functions you need for downstream processing. Triaxes [8] is also a good choice for lenticular images. With just two exposures, you can create an anaglyph image for red and cyan glasses or a simple wiggle image. More uniform motion and lenticular images will always require a series of images.

To begin, get a series of eight images for a spatial image. The images must be aligned uniformly; even slight skew and small vertical differences will make the images unusable. In StereoPhoto Maker, select *File | Multiple Images | Auto rotation adjustment* and select the images to be adjusted. In the second step, you need a common reference point in each image. The function for this can be found in *File | Multiple Images | X-Y adjustment and cropping*.

Now you can print the image by selecting *Edit | Create Lenticular Image*. Set the *Lenticular Lens Pitch* and the printer resolution to match the lenticular film. Finally, print the image with *File | Print preview*. Lenticular film of 15x10cm is available with different lens spacings with vertical and horizontal alignment. You need to align the self-adhesive films

over the printout and then carefully press them on. A laminator is useful for larger formats.

If you want to process the image series as a wigglegram, use the ImageMagick convert function. The following command adds all JPEGs with the `image` prefix in the current directory to the animated GIF:

```
$ convert -delay 10 -loop 0 \
image*.jpg <Wiggle>.gif
```

The playback is in an infinite loop at 10fps. The procedure depends on the size and number of images. Instead of a GIF, you can use the MP4 format. A wiggle cannot be printed, of course, so check out the examples online [9] if you need a visual reference.

The Triaxes 3DMasterKit, which is a commercial product, is a good choice for lenticular images. The license will not cost you much, and the investment is definitely worthwhile. After you upload the frames, you can reorder them and orient them alternately before cropping the images and computing the lenticular image. The kit also has other useful features, such as animations and layered 3D.

## Conclusions

Lenticular images as analog 3D representations, and animations and wiggles for

the Internet, give photographers a creative tool. Even with a conventional camera, you can achieve presentable results with a little practice.

The Camarray HAT by ArduCam lets you use a multisensor system in single, dual, or quadro mode and construct a DIY camera that suits your ideas. This hardware opens up a wide field of experimentation for amateur photographers, ranging from high-quality stereo images to low-resolution shaky images.

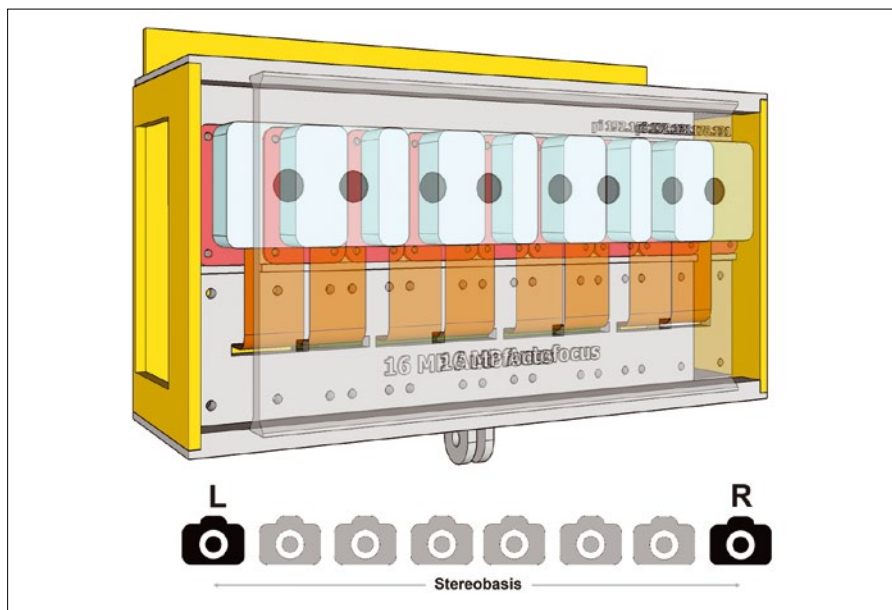
All you need for the build is a Raspberry Pi, a multicamera system, and a power pack. On the local WiFi network, a smartphone or tablet gives you a graphical user interface, and Python, Libcamera, and Guizero form the software underpinnings. StereoPhoto Maker and Triaxes take care of downstream processing. ■■■

## Info

- [1] Installing the Camarray HAT: <https://forum.arducam.com/t/imx519-quad-hat-mode-switching-and-faq/2399?u=wong>
- [2] Documentation for the Camarray HAT: <https://www.arducam.com/docs/cameras-for-raspberry-pi/raspberrypi-libcamera-guide/>
- [3] Source for Lenticular film: [https://www.glaserde.de/shop/Lentikular\\_Folien\\_DIN\\_A6/index.html](https://www.glaserde.de/shop/Lentikular_Folien_DIN_A6/index.html)
- [4] StereoPhoto Maker: <http://stereo.jp.org/ger/stphmkr/index.html>
- [5] 3DMasterKit: <https://triaxes.com/3dmasterkit/>
- [6] Libcamera documentation: [https://www.raspberrypi.com/documentation/computers/camera\\_software.html](https://www.raspberrypi.com/documentation/computers/camera_software.html)
- [7] Code for this article: <https://linuxnewmedia.thegood.cloud/s/5Rzx9tQW2FJ6N3Z>
- [8] Triaxes: <https://triaxes.com/legend/>
- [9] 3D-Foto und Video: <https://www.3d.imagefact.de> (in German)

## Author

**Dr. Guenter Pomaska** worked for many years in the fields of photogrammetry, application software development and support. He was teaching computer graphics and related topics in higher education institutes. After retirement he directed his activities to 3D imaging.



**Figure 6:** Intermediate frames divide the base (distance between the right- and left-most cameras).

# GET TO KNOW ADMIN



*ADMIN Network & Security* magazine is your source for technical solutions to real-world problems.

*ADMIN* is packed with detailed discussions aimed at the professional reader on contemporary topics including security, cloud computing, DevOps, HPC, containers, networking, and more.

**Subscribe to *ADMIN*** and get 6 issues delivered every year



Want to get ADMIN in your inbox?

**Subscribe free to ADMIN Update**

and get news and technical articles you won't see in the magazine.

[bit.ly/HPC-ADMIN-Update](https://bit.ly/HPC-ADMIN-Update)



**ADMIN**  
Network & Security



@adminmagazine



@adminmag



ADMIN magazine



@adminmagazine



# MakerSpace

Circuit simulation with hardware  
description languages

## Test Run

Designing field-programmable gate arrays is only half the job: The hardest part is the simulation, but Linux is the best place to tackle certain challenges. *By Marco Fioretti*

**I**n the previous issue of *Linux Magazine*, I introduced the digital integrated circuits called field-programmable gate arrays (FPGAs) [1], explained why they are important, and described the bare minimum needed to set up and try a state-of-the-art FPGA design environment on Linux. This article, which is divided into three parts, concludes that introduction.

First I'll describe the main types of elementary digital circuits that can be built, combining the logic gates covered in the first tutorial and the basic features of hardware description languages (HDLs) that allow their implementation and simulation.

The central part combines some HDL code that's freely available online [2] [3] into one simple HDL module and the code necessary to simulate it. The final part shows a run of that simulation and concludes by explaining why Linux is the perfect platform for working with FPGAs.

### Representing Digital Circuits with HDL

FPGAs (and every other digital integrated circuit) comprise millions of logic gates, with boolean input and output signals – that is, having one of two states, conventionally represented as 1 or 0.

Sequential components are combinations of logic gates that sample all their

input ports, and (with constant, predefined delays) update their outputs on the positive or negative fronts of one or more signals called clocks (i.e., square waves of a fixed frequency that constitute common timing references for the chip).

The simplest sequential block is a memory element, also called a flip-flop [4], that can load and preserve binary values: Its output changes only on the fronts of a control signal, almost always a clock, copying and then preserving whatever value the input pin has in those instants.

You can think of combinational logic circuits as the exact opposite of sequential circuits, because they use no clock signal and have no memory. Their output never depends on a previous value but changes immediately every time one of the inputs changes. In practice, there will always be some delay, which will depend on the complexity of the wiring.

Synchronous circuits are what you get by attaching the outputs of a combinational circuit to the inputs of a memory element. The inputs of the combinational circuit are generic signals combined in a feedback loop with the outputs of that same memory element. The simplest synchronous circuit possible is probably the toggle [4] (Figure 1). The triangle represents a combinational inverter, whose output is always the opposite of its input: When Q is 1, D is 0, and

vice versa. Q, however, is only updated when the clock (clk) signal has a positive transition and thus has the behavior shown in Figure 2.

Of course, synchronous circuits can be much more complicated than a toggle. Theoretically, a synchronous circuit can have any number of memory elements and input signals in addition to the clock. Consequently, the output of a synchronous circuit will be multibit binary buses, not single wires, and the combinational part could be a very complex boolean function of all the inputs and memory elements together.

The presence of a clock signal that makes the outputs change only at predefined instances makes synchronous circuits very reliable, but with a significant constraint: They behave as designed only if the inputs of all their memory elements are always stable when a clock transition comes to sample them. Because those signals come out of the combinational part of the circuit, that part cannot have so many levels of logic gates that signal propagation through it would take more than one clock cycle.

Aside from memory banks, the most common type of synchronous circuit is the finite-state machine (FSM), which is an “abstract machine that can be in exactly one of a finite number of states at any given time” [5]. In practice, the state of an FSM at any clock cycle can depend on the whole sequence of transitions it



experienced in the previous  $N$  clock cycles, where  $N$  depends on how many states and input signals the FSM has and how complex its combinational part is. Because that “state” could represent pretty much anything, from a plain counter to the phases of a real-time cryptographic algorithm, FSMs are the most common component of many HDL designs.

## Making It in HDL

To facilitate design and reuse, digital circuits are partitioned in hardware modules that can contain submodules; however, as far as the rest of the circuit is concerned, it is basically a black box.

Digital designers create models of their module with HDL languages such as VHDL or Verilog. I focus on Verilog in this article, but almost everything you learn here applies to VHDL as well.

HDLs add two classes of features to those present in software programming languages. One class comprises all the operators and constructs needed to describe the physical characteristics of the logic gates, such as the capability to change only on clock transitions. The other class includes everything needed to simulate and analyze the real-time behavior of a circuit model.

HDLs let designers describe blocks that will always operate and be simulated in parallel with all the other blocks of the design. If you follow the rules, the behavior of every block will always be the same regardless of the order in which all the other blocks are instantiated. The other main features are the data types and syntax constructs that support modeling the circuit types.

## Reg vs. Wires

The most important of HDL language data types are wires and regs (registers). Both wires and regs can store multiple bits.

Wires correspond to logical “nets” (i.e., connections between circuit elements). Although wires can represent actual direct connections between two or more modules, in those cases,

you cannot assign values to a wire (except by forcing it, for debug purposes), because it’s only driven by modules. You may, however, define a local wire whose value is an instantaneous Boolean function of some other signal. In all cases, when the moment comes to create the circuit, wires always translate to combinational logic.

Unlike wires, regs are a Verilog data type that can store values. The values must be explicitly calculated in the HDL code. Depending on how you write that code, regs can become either combinational or sequential logic.

HDL code that explicitly declares every reg variable that must exist in the target silicon and explicitly calculates its values is called Register Transfer Level (RTL) code. Well-written RTL code might take

## Listing 1: Sample Verilog Code

```
01 module sample_counter;
02   input CLK;                // Clock signal
03   input RST;               // Reset signal, active low
04   output reg[23:0] COUNTER; // 24-bit counter;
05   output wire SOME_OTHER_SIGNAL; // single bit output
06   wire EIGHT_PULSE;
07
08   always @(posedge CLK)
09   begin
10     if (RST == 1'b0)
11       COUNTER <= 24'd0;    // set all the bits of COUNTER to 0
12     else
13       COUNTER <= COUNTER + 24'd1; // 24 bit adder
14   end
15
16   assign EIGHT_PULSE = (COUNTER[2:0] == 3'd0) ? 1'b1: 1'b0;
17
18   controller ctl (EIGHT_PULSE, SOME_OTHER_SIGNAL);
19
20 endmodule
```

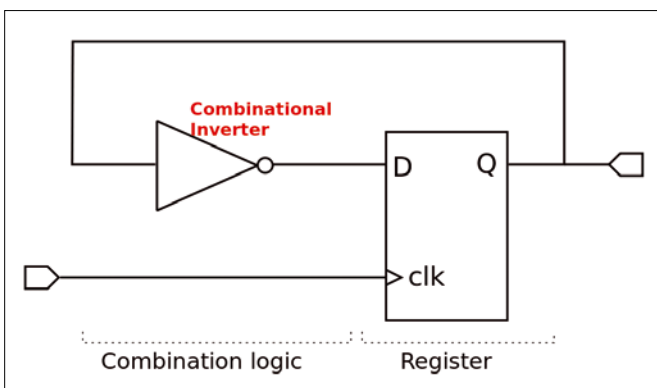
physical circuit that behaves exactly like its HDL model without having to declare and connect every single logic gate manually to all the others.

The deliberately dumb code of Listing 1 shows both kinds of data types and the Verilog syntax to handle them at the RTL level.

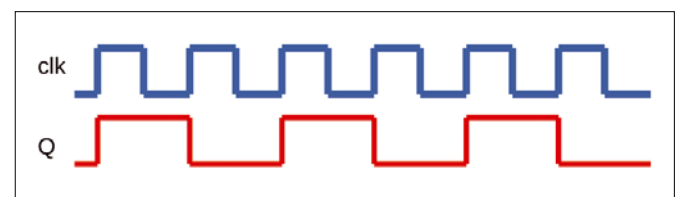
In Verilog, modules are enclosed by the `module` and `endmodule` keywords. Verilog modules always have a unique name and can instantiate other modules. Listing 1 describes a simple module called `sample_counter` that contains an instance called `ctl` of another module called `controller`.

The first five lines after the `module` declaration (lines 2-6) list all the input and output ports of the module and their nature: two single-bit inputs, `CLK` and `RST`; two outputs, `COUNTER` and `SOME_OTHER_SIGNAL`; and a local single-bit wire, `EIGHT_PULSE`. `COUNTER` is a 24-bit-wide reg, and `SOME_OTHER_SIGNAL` is a single-bit wire. Comments have the same syntax as in C. I recommend you

more time to write than higher level code, but you will have a much higher probability of generating a



**Figure 1:** The simplest combination of memory and combinational logic: a toggle that inverts its value at every clock cycle.



**Figure 2:** The feedback loop of Figure 1 generates an output that is synchronous with the clock, at half the frequency.

use comments extensively to document what each signal is and how the module works.

Between the declaration of the variables and the instantiation of the `ctl` submodule, `sample_counter` contains two blocks, or processes, that work concurrently. The `always` block starting on line 8 declares that the value of `COUNTER` must be updated at every positive edge of the `CLK` signal with a non-blocking assignment. As long as `RST` is low, `COUNTER` will be zero; otherwise, its value will be incremented by 1. It is important to note the syntax of the non-blocking

### Blocking

Blocking assignments are just like “normal” software assignments and always translate to combinational logic. The name describes the fact that they “block” the simulator, which must execute them serially, one at a time, in the order they appear. The obvious consequence is that, all inputs being equal, the complete output of a series of blocking assignment will depend on their order. In other words, blocking assignments are one of the most common ways for inattentive novice HDL designers to hurt themselves, producing code that mysteriously does not work as they wanted.

Non-blocking assignments are almost the opposite. For example, assume that a simulation has a time resolution of 1ns; that is, it must calculate all the values of all the wires and registers it contains every nanosecond. Non-blocking means that, at the beginning of each nanosecond, the simulator:

- samples the values in that instant of all the variables that appear on the right side of all the non-blocking statements it contains, at every level of the hierarchy, and
- then uses those sampled values (some of which might be feedback loops!) to calculate the next values of all the variables on the left side of all the non-blocking statements.

The ability of HDL simulators *not* to block themselves when calculating any next value before they have collected all the inputs for all the assignments made with the `<=` operator is what emulates the real behavior of real hardware registers. Working in that way, the result of each statement does not depend on the order in which all the non-blocking statements are written.

assignment operator (`<=`) and that you should always declare the width of numeric constants. If line 11 said `5'd0`, for example, the five least significant bits of `COUNTER` would be set to zeroes, but all the other bits might end up with undefined values.

See the “Blocking” box for more on what blocking and non-blocking means in HDL. For now, just take note that because it uses the non-blocking assignment and works on clock edges, the `always` block will be synthesized as a synchronous 24-bit counter.

Line 13 shows the beauty and power of HDL languages: You don’t need to know, or write down in detail, how 24-bit binary addition actually works. Just use `+`, and the Verilog compiler, synthesizer, and simulator take care of all the implementation and simulation details.

The same considerations apply to the last piece of the `sample_counter`: The `assign` keyword that controls `EIGHT_PULSE` is a continuous blocking assignment, because it uses `=` instead of `<=`, which runs every time any of the three least significant bits of `COUNTER` change (that is what `COUNTER[2:0]` means). When all those bits are zero, that is when `COUNTER` is 0 or 8, 16, ...; then, `EIGHT_PULSE` becomes immediately high; otherwise, it is low. Because it uses the `assign` keyword and the blocking assignment, this part of the module will be synthesized as combinational logic.

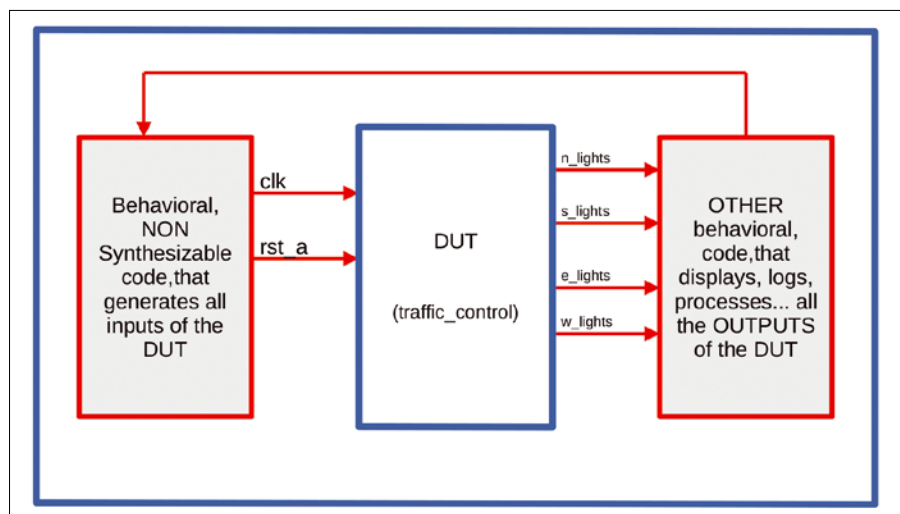
### HDL Test Benches

HDL languages have many keywords and features that could never be

mapped to silicon but are absolutely essential to create and run test benches. A test bench is a virtual HDL environment used to test all the capabilities of a specific module or combination of modules. Figure 3 shows the test bench of a circuit. The outer blue box is the module called `traffic_control_tb` (described in detail later).

In general, a test bench is an HDL module that instantiates the device under test (DUT) together with non-synthesizable HDL code written specifically for simulation and verification and often called “stimuli.” Some of that code (left block in Figure 3) generates all the inputs of the DUT. Other parts (the right block of Figure 3) display and log all the outputs of the DUT and might even change the DUT inputs dynamically in a feedback loop represented by the red line in the upper part of Figure 3. The code in Listing 2, for example, would write *Simulation started!* to the logfile right at the beginning of the simulation, display the message *Waiting for edge* after 100 time units, then do nothing until the signal `FINISHED` goes from low to high, when it would write *FINISHED* to the same logfile. A test bench can contain as many `initial` blocks as you want, and they will all work independently.

As far as stimuli are concerned, the two most important Verilog features are functions and tasks. Functions process some variables instantaneously whenever the variable changes, returning one, and always one, value. Tasks



**Figure 3:** Every simulation of digital hardware consists of a test bench that feeds the device under test (DUT) with the same inputs as the real circuit and then automatically displays and analyzes its outputs.

**Listing 2: HDL Stimuli**

```
initial
begin
  $display("Simulation started!");
  #100 $display("Waiting for edge");
  @(posedge FINISHED) $display("FINISHED!");
  ...
```

can return any number of results (even none) and call other tasks or functions; most importantly, a task can have any duration you want. A task can contain any number of time-consuming statements like always @(posedge CLK) or delays

expressed with the hash symbol, as in Listing 2.

At the end of the day, writing good test benches is at least as important as writing good modules, to the point that some HDL designers eventually specialize in test benches to validate the designs of the rest of the team.

**Listing 3: Traffic Signal Controller**

```
001 module traffic_control(n_lights,s_lights,e_lights,w_      049
lights,clk,rst_a);                                       050
002                                                                 051
003   output reg [2:0] n_lights,s_lights,e_lights,w_lights;  052
004   input   clk;                                           053
005   input   rst_a;                                         054
006                                                                 055
007   reg [2:0] state;                                       056
008                                                                 057
009   parameter [2:0] north  = 3'b000;                       058
010   parameter [2:0] north_y = 3'b001;                     059
011   parameter [2:0] south  = 3'b010;                       060
012   parameter [2:0] south_y = 3'b011;                     061
013   parameter [2:0] east   = 3'b100;                       062
014   parameter [2:0] east_y = 3'b101;                       063
015   parameter [2:0] west   = 3'b110;                       064
016   parameter [2:0] west_y = 3'b111;                       065
017                                                                 066
018   reg [2:0] count;                                       067
019                                                                 068
020   always @(posedge clk, rst_a)                          069
021     begin                                               070
022       if (rst_a == 1'b0) // Reset signal is active LOW  071
023         begin                                           072
024           state <= north;                                073
025           count <= 3'b000;                              074
026         end                                             075
027       else                                              076
028         begin                                           077
029           case (state)                                   078
030             north :                                     079
031               begin                                     080
032                 if (count == 3'b111)                   081
033                   begin                                 082
034                     count <= 3'b000;                   083
035                     state <= north_y;                  084
036                   end                                   085
037                 else                                   086
038                   begin                                 087
039                     count <= count+3'b001;             088
040                     state <= north;                    089
041                   end                                   090
042                 end                                     091
043             north_y :                                   092
044               begin                                     093
045                 if (count == 3'b011)                   094
046                   begin                                 095
047                     count <= 3'b000;                   096
048                   end                                   097
049                 end                                     098
050             south :                                     099
051               begin                                     100
052                 if (count == 3'b111)                   101
053                   begin                                 102
054                     count <= 3'b0;                     103
055                     state <= south_y;                  104
056                   end                                   105
057                 else                                   106
058                   begin                                 107
059                     count <= count+3'b001;             108
060                     state <= south;                    109
061                   end                                   110
062                 end                                     111
063             south_y :                                   112
064               begin                                     113
065                 if (count == 3'b011)                   114
066                   begin                                 115
067                     count <= 3'b0;                     116
068                     state <= east;                     117
069                   end                                   118
070                 else                                   119
071                   begin                                 120
072                     count <= count+3'b001;             121
073                     state <= south_y;                  122
074                   end                                   123
075                 end                                     124
076             east :                                     125
077               begin                                     126
078                 if (count == 3'b111)                   127
079                   begin                                 128
080                     count <= 3'b0;                     129
081                     state <= east_y;                   130
082                   end                                   131
083                 else                                   132
084                   begin                                 133
085                     count <= count+3'b001;             134
086                     state <= south_y;                  135
087                   end                                   136
088                 end                                     137
089             end                                         138
090           end                                           139
091         end                                             140
092       end                                               141
093     end                                                 142
```

## Listing 3: Traffic Signal Controller (cont.)

```

098         end
099
100     east_y :
101         begin
102             if (count == 3'b011)
103                 begin
104                     count <= 3'b0;
105                     state <= west;
106                 end
107             else
108                 begin
109                     count <= count+3'b001;
110                     state <= east_y;
111                 end
112             end
113
114     west :
115         begin
116             if (count == 3'b111)
117                 begin
118                     state <= west_y;
119                     count <= 3'b0;
120                 end
121             else
122                 begin
123                     count <= count+3'b001;
124                     state <= west;
125                 end
126             end
127
128     west_y :
129         begin
130             if (count == 3'b011)
131                 begin
132                     state <= north;
133                     count <= 3'b0;
134                 end
135             else
136                 begin
137                     count <= count+3'b001;
138                     state <= west_y;
139                 end
140             end
141         endcase // case (state)
142     end // always @ (state)
143 end
144
145 always @(state)
146     begin
147         case (state)
148             north :
149                 begin
150                     n_lights = 3'b001;
151                     s_lights = 3'b100;
152                     e_lights = 3'b100;
153                     w_lights = 3'b100;
154                 end // case: north
155
156             north_y :
157                 begin
158                     n_lights = 3'b010;
159                     s_lights = 3'b100;
160                     e_lights = 3'b100;
161                     w_lights = 3'b100;
162                 end // case: north_y
163
164             south :
165                 begin
166                     n_lights = 3'b100;
167                     s_lights = 3'b001;
168                     e_lights = 3'b100;
169                     w_lights = 3'b100;
170                 end // case: south
171
172             south_y :
173                 begin
174                     n_lights = 3'b100;
175                     s_lights = 3'b010;
176                     e_lights = 3'b100;
177                     w_lights = 3'b100;
178                 end // case: south_y
179
180             west :
181                 begin
182                     n_lights = 3'b100;
183                     s_lights = 3'b100;
184                     e_lights = 3'b100;
185                     w_lights = 3'b001;
186                 end // case: west
187
188             west_y :
189                 begin
190                     n_lights = 3'b100;
191                     s_lights = 3'b100;
192                     e_lights = 3'b100;
193                     w_lights = 3'b010;
194                 end // case: west_y
195
196             east :
197                 begin
198                     n_lights = 3'b100;
199                     s_lights = 3'b100;
200                     e_lights = 3'b001;
201                     w_lights = 3'b100;
202                 end // case: east
203
204             east_y :
205                 begin
206                     n_lights = 3'b100;
207                     s_lights = 3'b100;
208                     e_lights = 3'b010;
209                     w_lights = 3'b100;
210                 end // case: east_y
211         endcase // case (state)
212     end // always @ (state)
213 endmodule

```

## Traffic Light Controller

Listing 3 is a simple implementation of a single controller for four traffic lights, each placed at one side of the intersection between a road going north to south and another going east to west. For simplicity, unlike a normal intersection, the traffic signals are mutually exclusive: only one is green, or yellow, at any given time.

Listings 3 and 4 are my own combinations of code created for this article that you can download from online sources [2] [3]. Expert HDL designers would be able to express the same functionality with fewer lines of code, but I deliberately kept the verbose structure to make the code easier to understand.

Although the code in Listing 3 is verbose, it boils down to just three parts. The first part, from the beginning to line 19, just lists all the inputs, outputs, and internal variables. The only thing new in this section might be the `parameter` keyword, which gives human-readable names like `north` and `north_y` to otherwise hard-to-remember binary strings such as `011` or `110`.

Each of the parameters represents the state of all four light signals. The `north` and `north_y`, for example, are the states in which the signal on the north side of the intersection must be green or, respectively, yellow, while the other three must be all red. The other six parameters work in the same way, representing the moments when each of the other three signals is green or yellow, and all the others are red.

The `always` block (lines 21-144) is the FSM that makes the signals always work in an ordered way. When synthesized, it will produce a 3-bit state register just called `state`, and another 3-bit register for the counter called `count`. The sequence of states corresponds to the order in which the signals must become green. The counter is used to make every phase last exactly the right amount of time.

Whenever the reset `rst_a` is low, both state and counter revert to their default values of `north` and `0` and stay that way. If the reset is high, the `case` statement (line 29) takes control to calculate the next state of the FSM. Because all the transitions work in the same way, an in-depth look at lines 31 to 56 is enough to understand how the whole FSM works.

When the state is `north`, the counter is incremented at every clock cycle (line 40) for eight cycles, until it is equal to 7

(`3'b111` in binary). When that happens, the counter is reset and the next state is decreed to be `north_y` (lines 35 and 36).

## Listing 4: Test Bench

```

01 `timescale 1ns/1ps
02 module traffic_control_tb;
03 parameter ENDTIME = 60000000;
04
05 wire [2:0] n_lights,s_lights,e_lights,w_lights;
06 reg clk,rst_n;
07
08 traffic_control DUT (n_lights,s_lights,e_lights,w_lights,clk,rst_n);
09
10 initial
11 begin
12   clk=1'b1;
13   forever #5 clk=~clk;
14 end
15
16 initial
17   begin
18     fork
19       reset_gen;
20       logwriter;
21     endsimulation;
22   join
23 end
24
25 /*****/
26 task reset_gen;
27   begin
28     rst_n = 0;
29     #23 rst_n = 1;
30   end
31 endtask
32
33 /*****/
34 task logwriter;
35   begin
36     $display("LOG: Verilog Simulation for Linux Magazine ----");
37     $display("-----");
38     $monitor("LOG_CHK = %d, reset = %b, N = %b, S = %b, E = %b, W = %b",
39             $time,rst_n, n_lights,s_lights,e_lights,w_lights);
40   end
41 endtask
42
43 /*****/
44
45 task endsimulation;
46   begin
47     #ENDTIME
48     $display("LOG_END");
49     $finish;
50   end
51 endtask
52
53 endmodule

```

As you will see, that transition will make the north signal turn its green light off and yellow light on.

When the state is north\_y (lines 44-56), the behavior is the same, with the only difference being that transition to state south (i.e., green light on the south side of the intersection and red on the other three) happens after just four cycles. As soon as count is equal to 3'b011 (line 46), the next state becomes south.

The rest of the code up to line 144 works in the same way, just cycling among the four signals in the order north, south, east, west. Each signal first stays green for eight clock cycles (states north, south, east, west) and then yellow for four (states north\_y, south\_y, east\_y, west\_y).

The last part of the traffic controller, the always @(state) block (lines 145-212), is the combinational logic that looks at the state of the FSM and drives the signals that turn on or off each individual light of each signal. You know it will be combinational because no clock edges are triggering the block in line 145, just variations of the state register, and because this block uses blocking assignments instead of non-blocking assignments like the FSM.

Even here, the code is much simpler than its length suggests, and looking at one branch of the case statement is enough to figure out the whole thing. All you need to know is that each 3-bit bus combines the light switches of one signal in this order: The rightmost bit is connected to the green light, the middle one

to the yellow light, and the leftmost to the red light. Therefore, according to lines 150 to 154, whenever the status is north, the green light of the north signal must be on (n\_lights = 3'b001) and the other three must be on red (value 3'b100). When the state is equal to east\_y instead (lines 208-210), the east signal must be on yellow (e\_lights = 3'b010) and all the others on red.

### Test Bench

The traffic\_control\_tb module in Listing 4 is a test bench for the traffic controller.

Because it starts with a backtick, the first line of the testbench is a directive for the simulation compiler; it means that the simulation time unit is 1ns and the resolution (i.e., the minimum time interval between two events to consider them distinct) is 1ps. After that directive, the first thing you should note is that the test bench module needs no inputs or outputs (line 2) and that the DUT, (the traffic controller of Listing 3) is instantiated in line 8. (See the “Regs vs. Wires” box for more on the differences between Listings 3 and 4.)

The initial block in line 10 just generates the clk signal, giving it an initial value and then inverting it every five time units (line 13). Given the compiler directive in line 1, each clock cycle lasts 10ns. Of course, with a clock this fast, the traffic controller of Listing 3 would be completely useless, because each green light would only last 80ns! In practice, I left that value as is to check

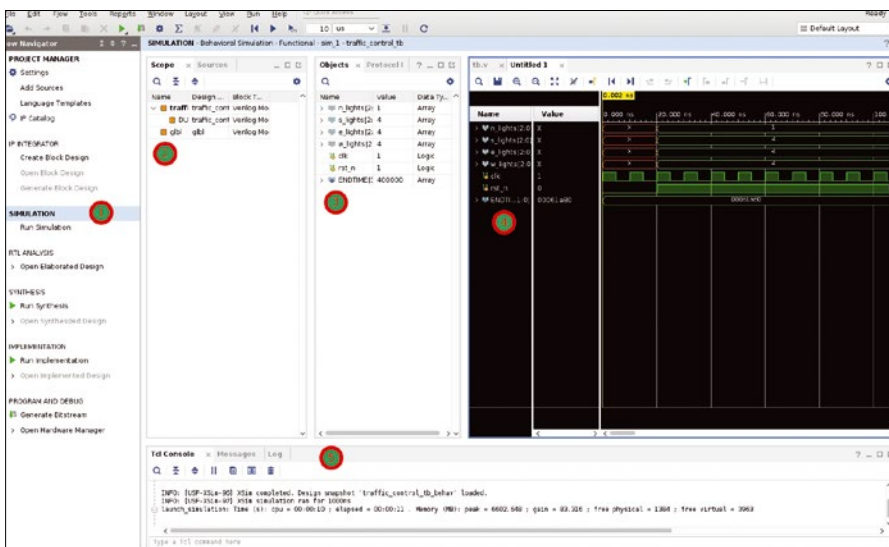
### Regs vs. Wires

Compare lines 5 and 8 of Listing 4 with line 3 of Listing 3: Why are n\_lights and the other three buses that are regs in the traffic\_control module declared as wires?

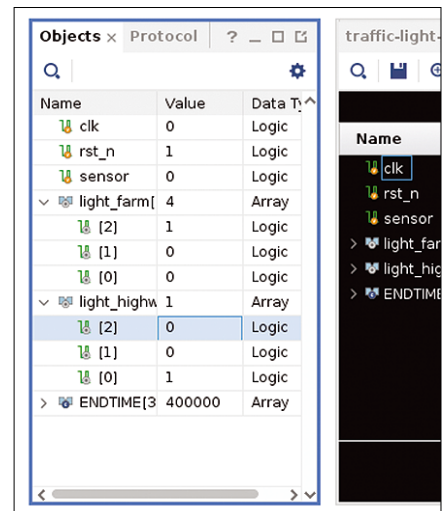
The answer is simple: The original author of the traffic\_control code declared those buses as regs to be able to control them in the always @(state) block of Listing 3. At the test bench level, however, those same buses cannot be controlled, because they pop up from, and are driven by, the traffic\_control module. That’s why they are declared as wires.

whether the outputs evolve in the right way; the frequency does not really matter.

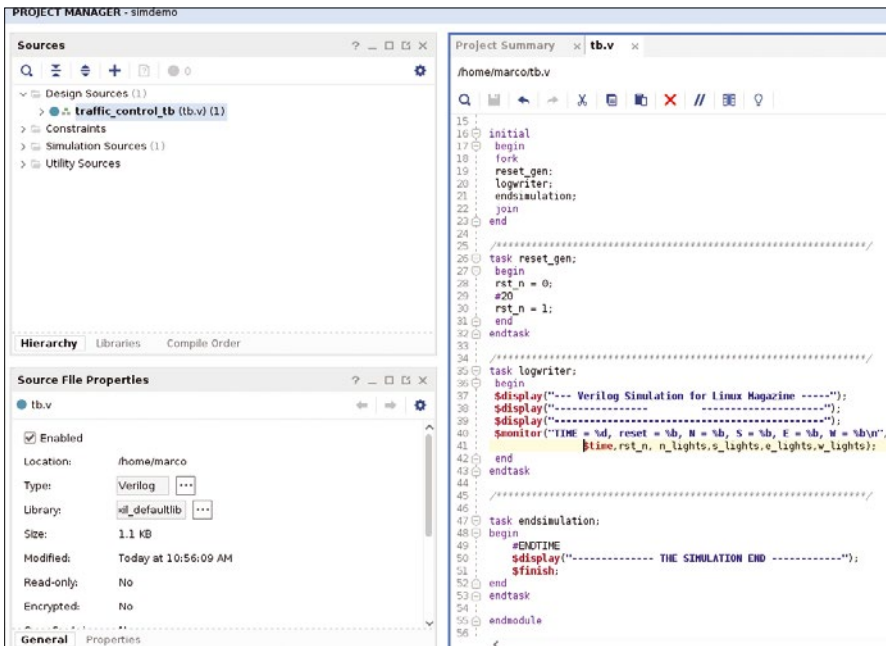
The other initial block starting in line 16 shows that you can launch many independent tasks, each running in parallel (but possibly interacting with the others), just by enclosing them between the keywords fork and join. The three tasks launched in that block are enough to show the flexibility of HDL test benches, because they have very different purposes: reset\_gen just generates the rst signal once and then stops; end\_simulation waits for ENDTIME nanoseconds, with ENDTIME defined in line 3, then prints a notification and terminates the simulation; the most important logger writes the values of all the signals of interest to the logfile for further analysis.



**Figure 4:** The graphical interface of the simulator in the Vivado design platform.



**Figure 5:** A close-up of the Objects panel with a list of all the signals at every level of the design hierarchy, their types, and their current values.



**Figure 6:** The built-in Vivado HDL editor is well integrated with all the other parts of the design and simulation flow.

## Testing the Traffic Controller

If you installed the Vivado environment as I explained in the previous article of this series [1], all you have to do is type `vivado` at a command prompt, load the source files shown here, and click *Run simulation* in the graphical interface. Figure 4 shows the Project Manager (1), a Scope panel with all your source files (2), and the Objects panel (3) with all the signals, also visible in Figure 5. Other panels host the waveform analyzer (4) and an interactive Tcl console (5) that displays the logfile and other information. The Waveforms panel hosts tabs in which you can edit your source files (Figure 6). The simulation appears in Figure 7.

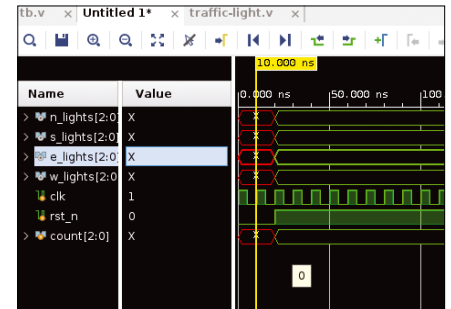
## Conclusion

Practicing FPGA and HDL design on Linux with state-of-the-art software is much easier than it might seem. Everything in this design flow is available as

good old plain text, from the source files to the stimuli and logs, and it's difficult to overestimate the true potential of this apparently trivial fact.

Look again at Listing 3: Yes, it's boring, but also really easy to generate automatically with a Python or Perl script. What's more, the same approach works wonders with simulations, before and after running them. You can have very complex stimuli with sophisticated tasks that interact in countless ways, written for you by much shorter scripts in higher level languages. Once the simulation is over, other scripts could parse logfiles to spot any problem automatically and point you to the exact instant of the simulation you should analyze personally to solve the problem.

In other words, not only is FPGA design easier than it seems, but an operating system like Linux with rock-solid text-processing tools like `sed`, `awk`, `Perl`, and `Python` make it even easier. ■■■



**Figure 7:** The same simulation of Figure 4 after fixing the Reset signal. Now both the counter and the output signals change with time.

## Info

- [1] "Designing field-programmable gate arrays" by Marco Fioretti, *Linux Magazine*, issue 271, June 2023, pg. 58, <https://www.linux-magazine.com/Issues/2023/271/Introduction-to-FPGA-Design>
- [2] Traffic light system: <https://vlsicoding.blogspot.com/2013/11/verilog-code-for-traffic-light-control.html>
- [3] Another traffic light controller: [www.fpga4student.com/2016/11/verilog-code-for-traffic-light-system.html](http://www.fpga4student.com/2016/11/verilog-code-for-traffic-light-system.html)
- [4] RTL/flip-flop: [https://en.wikipedia.org/wiki/Register-transfer\\_level](https://en.wikipedia.org/wiki/Register-transfer_level)
- [5] Finite state machines: [https://en.wikipedia.org/wiki/Finite-state\\_machine](https://en.wikipedia.org/wiki/Finite-state_machine)

## Author

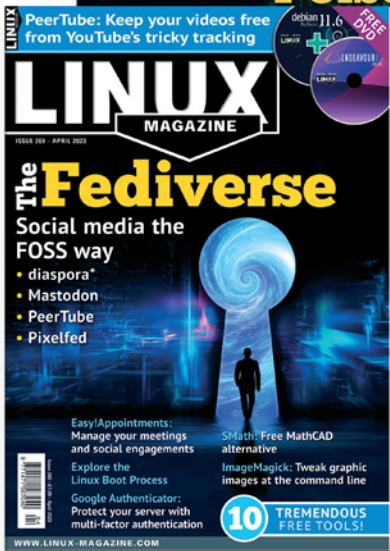
Marco Fioretti (<http://mfioretti.substack.com>) is a freelance author, trainer, and researcher based in Rome, Italy, who has been working with free/open source software since 1995 and on open digital standards since 2005. Marco also is a board member of the Free Knowledge Institute (<http://freeknowledge.eu>).



# Linux Magazine Subscription

Print and digital options  
12 issues per year

▶ SUBSCRIBE  
[sparkhaus-shop.com](http://sparkhaus-shop.com)



Expand your Linux skills:

- In-depth articles on trending topics, including Bitcoin, ransomware, cloud computing, and more!
- How-tos and tutorials on useful tools that will save you time and protect your data
- Troubleshooting and optimization tips
- Insightful news on crucial developments in the world of open source
- Cool projects for Raspberry Pi, Arduino, and other maker-board systems

Go farther and do more with Linux, subscribe today and never miss another issue!

Follow us



@linux\_pro



Linux Magazine



@linuxpromagazine



@linuxmagazine

## Need more Linux?

Subscribe free to Linux Update

Our free Linux Update newsletter delivers insightful articles and tech tips to your inbox every week.

[bit.ly/Linux-Update](https://bit.ly/Linux-Update)





**For many users, the best thing about Linux is the tools** – thousands of tools, and most of them available for no cost. You can be like a kid in a toy store, or, even better, like a kid in a toy maker’s castle, where you can choose any toys you want and you don’t have to worry about a parent saying “only choose one” or “choose the least expensive one.” The tools of Linux let you build exactly the environment you want, customized for your preferences and habits. In this month’s Linux Voice, we introduce you to three inspired utilities you might find useful and you won’t find in the default configuration: the Espanso text expander, the nnn text-based file manager, and the terminal browser Carbonyl.



Image © Alexandr Moroz, 123RF.com

# LINUX VOICE

**Doghouse – Databases 74**

*Jon “maddog” Hall*  
There are many FOSS databases available inexpensively today, and they might serve new projects well.

**Espanso 75**

*Ferdinand Thommes*  
Espanso is a cross-platform text expander that can do far more than simply replace text modules.

**nnn 80**

*Nate Drake*  
If you’re a Linux lover, you’ll know the command line is the slickest and most efficient way to interact with the system. Free yourself from point-and-click with developer jarun’s nnn command-line file manager.

**FOSSPicks 84**

*Graham Morrison*  
This month Graham looks at Seamly2D, Arianna, X-Pipe, Nosey Parker, IEM Plug-in Suite, Open Hexagon, and more!

**Tutorial – Carbonyl Graphical Web Browser 90**

*Marco Fioretti*  
This Chromium port can run inside any console, with minimal resources, and is a great tool for making old computers really useful – and learning programming along the way.

**ELVIE**



CC-BY-SA WWW.PEPPERTOP.COM



Jon “maddog” Hall is an author, educator, computer scientist, and free software pioneer who has been a passionate advocate for Linux since 1994 when he first met Linus Torvalds and facilitated the port of Linux to a 64-bit system. He serves as president of Linux International®.

# MADDOG'S DOGHOUSE

There are many FOSS databases available inexpensively today, and they might serve new projects well. BY JON “MADDOG” HALL

## Considering FOSS Databases

Recently a friend started taking a postgraduate course in software engineering, and part of that course was the topic of databases. My friend had learned a little about them in his undergraduate degree in computer engineering, but not that much. Perhaps he would have learned more if he had studied “Computer Science.”

When I started at my first full-time job after university, databases as we know them were just getting started. System R at IBM and Ingres at the University of California, Berkeley, were research projects, trying to determine the best way to store and access data. Often funded by research grants from the government, tapes of the code were available at a nominal charge.

If you did not use a database to store your data, you had to deal with various issues such as backing up your data to get a consistent view of your total data, different byte and word size, different endianness types (little endian vs. big endian), loss of data due to power failures and system crashes due to lack of journaling, pulling large amounts of data over the very slow networking of the day, doing transaction processing (which is often used in business), and a variety of other issues. Eventually databases could also store and run complicated functions inside the engine itself, locating the processing right next to the data.

There was also a formalism of data when you used a database. Companies created data dictionaries and had data administrators who thought about the data itself, and not just the programs that were creating data or using data. Programmers and data administrators concentrated more effort on filtering bad data from getting into your database, which makes programming better overall.

Languages for accessing the data were created, and the one that most people know today is Structured Query Language (SQL), which has been extended over time to have new abilities.

Unfortunately for databases, the prevailing model of selling them in 1986 was as very expensive closed-source products. Commercial databases might cost as much as \$100,000 per server, and that was when \$100,000 was “a lot of money.” Consequently, in 1986 only about four percent of Unix systems had a database engine on the system, and that market was split between the market leaders (Oracle, Informix, Sybase, Ingres, and more).

“Commercial” operating systems such as Digital Equipment Corporation’s (DEC’s) VMS or IBM’s MVS systems were

considered to be the premier platforms for databases, and the database engines would write their files into the filesystems.

Not on Unix. On Unix the database companies insisted on writing directly to partitions of the disk, bypassing the filesystems altogether and the filesystem buffers, making it difficult to have data logs and backups done. Plus the performance of the databases was poorer on Unix systems than on a proprietary “commercial” OS of the same hardware platform.

As a product manager of Unix systems at DEC, I worked with an Ingres salesman to embed a relational database engine into every Ultrix, our trademarked Unix-like system we sold at a price so low that the customer didn’t even notice they were paying for it. Instead of only one percent of our Ultrix systems having a database engine for use by programs, now 100 percent of the systems could use them.

After we had accomplished this contract, I asked our Ingres salesman why Ingres worked so well on VMS (where his developers did their programming) and not so great on over 100 different Unix systems they supported.

He was a little embarrassed at first, but with prompting he gave me a white paper entitled, essentially, “The Eleven Reasons Why Unix Systems S\*#k as a Database Platform,” written by some of his engineers.

According to the paper, Unix needed multi-threaded I/O, good threading in general (many early Unix systems had single-threaded libraries incapable of multi-threading), synchronous filesystems, mmap (the ability to map files into virtual memory), ability to lock virtual memory into RAM, and a few other components, all of which were available through software called POSIX real-time extensions which Ultrix already had, but were not installed through default. The Ingres engineers were using compatibility libraries to replace the functionality they used on VMS, but did not know they could have that functionality native on Ultrix.

I informed the Ingres engineers that all they had to do was ask the customers to install that software package (which they’d already paid for) and they could have Ingres run as efficiently on Ultrix as it did on VMS. The next release of Ingres took advantage of the POSIX real-time extensions.

Today we have many FOSS databases to use, most of them low or no cost for the license. It could be worthwhile to use them in new projects. ■■■

# Espanso: Text expander and more Abbreviated

Espanso is a cross-platform text expander that can do far more than simply replace text modules. **BY FERDINAND THOMMES**

A text expander replaces predefined abbreviations with stored text modules. One example of this is *wkr*, which expands to *With kind regards* after entering the abbreviation. The AutoKey [1] tool I used until a year ago was no longer fit for purpose because it only works on Linux and only on X11 – which leaves Wayland out in the cold.

As an alternative to AutoKey, Espanso [2] has been available since 2019, and it goes beyond simply replacing abbreviations in texts. The text expander, written in Rust, runs on Linux, macOS, and Windows. You can install the application on Linux for Debian, Ubuntu, and other derivatives using a DEB package or as a DIY binary package. There are also AppImages and packages for Snap. Only the Arch User Repository (AUR) wants you to build Espanso directly. If your distribution is not directly supported, you can use the AppImage or build a package yourself. The developers are looking to extend this support over time.

Start by running the `echo $XDG_SESSION_TYPE` command to check whether the system uses X11 or Wayland as the session type. Depending on the result, you may need to select different packages or source code in each case, as described in the installation instructions [3]. Listing 1 shows the setup of the DEB package under X11. To do this, first download the package, then run `apt` to install and make sure that the installation worked (lines 1 to 3). Finally, introduce the Espanso service to `systemd` (line 4).

Then run the `espanso start` command to call the tool. After the welcome window closes, there is no trace sign of Espanso on the Linux desktop, whereas an icon appears in the control bar on macOS and Windows. An overview of the commands you can use to control Espanso is provided by `espanso -h` (Figure 1).

In its function as a text expander, Espanso waits for you to enter a keyword in the form `:keyword` and substitutes it for the stored text. For example, if you type `:espanso`, the correct response is `Hi there!`. Espanso replaces abbreviations everywhere you can write them, including in the shell.

You configure the application in several files below `~/.config/espanso`. The `default.yml` file in the `config/` folder contains global settings that you enable by removing the preceding hashtag. But be careful: YAML does not forgive mistakes with indentations.

## Configuration in YAML

You can use the `espanso edit` command to open the configuration file. If possible, the string to be replaced should not contain nonstandard characters. Replacing them works in some apps, but not in others. Because the configuration file is also written in YAML, you need to pay attention to the indentations again. Just follow the examples provided.

When you save your edits, Espanso shows you a pop-up window where you are prompted to confirm the action. If you made a mistake, a message will tell you at which position in which line this is (Figure 2).

### Listing 1: Installing Espanso

```
01 $ wget https://github.com/federico-terzi/espanso/releases/download/v2.1.7-beta/espanso-debian-x11-amd64.deb
02 $ sudo apt install ./espanso-debian-x11-amd64.deb
03 $ espanso status
04 $ espanso service register
05 service file already exists, this operation will overwrite it
06 creating service file in "~/.config/systemd/user/espanso.service"
07 enabling systemd service
08 service registered correctly
```

```
A Privacy-first, Cross-platform Text Expander

USAGE:
  espanso [FLAGS] [SUBCOMMAND]

FLAGS:
  -h, --help      Prints help information
  -v              Sets the level of verbosity
  -V, --version   Prints version information

OPTIONS:

SUBCOMMANDS:
  cmd          Send a command to the espanso daemon.
  edit        Shortcut to open the default text editor to edit config files
  env-path    Add or remove the 'espanso' command from the PATH
  help        Prints this message or the help of the given subcommand(s)
  install     Install a package
  log         Print the daemon logs.
  match       List and execute matches from the CLI
  migrate     Automatically migrate legacy config files to the new v2 format.
  package     package-management commands
  path        Prints all the espanso directory paths to easily locate configuration and matches.
  restart     Restart the espanso service
  service     A collection of commands to manage the Espanso service (for example, enabling auto-start on system
              boot).
  start       Start espanso as a service
  status      Check if the espanso daemon is running or not.
  stop        Stop espanso service
  uninstall   Remove a package
  workaround  A collection of workarounds to solve some common problems.
```

Figure 1: You can use the `-h` or `--help` parameters to investigate the available options.

The `match/` folder is where you specify what you want Espanso to replace and what with. The tool gives you some simple examples of replacements (Figure 3) in the `/match/base.yml` file. Replacements are defined by a pair of `trigger` and `replace` values. `trigger` denotes the string that triggers the replacement, while `replace` contains the replacement itself. A simple example looks something like this:

```
- trigger: ":kr"
  replace: "Kind regards,"
```

Another example, which often occurs at the end of letters and emails, gives you two lines of output.

```
- trigger: ":lp"
  replace: "Love,/nPeter"
```

In addition to `/n` for creating a new line, YAML uses two operators, `|` and `>`, as soon as more than two lines come into play. If you use `|` as the first operand after `replace`, Espanso outputs the text exactly as you write it. In line with this, the three lines from Listing 2 are also rendered as three lines. If you replace `|` with `>` here, the application writes the replacement text in a single line. You can learn more about these scalars in [YAML on Stack Overflow](#) [4].

### Static or Dynamic?

Espanso distinguishes between static and dynamic replacements. The examples described thus far belong to the static category; their output always remains the same. A typical example of a dynamic replacement, on the other hand, is outputting the current time of day. Espanso uses variables to let you do this.

The example from Listing 3 introduces the `time` variable in the replacement. I chose the name,

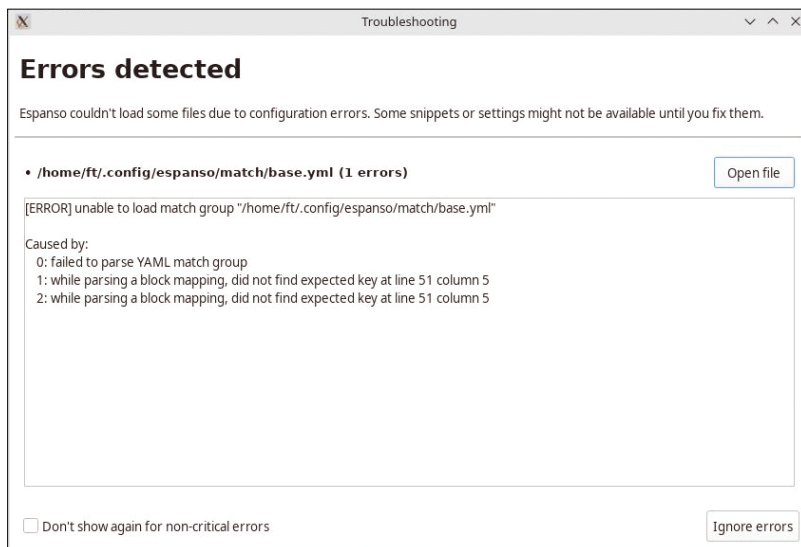


Figure 2: If an error occurs during an edit, you get a message after saving showing you the line and position of the error.

```
Listing 2: Operand |
- trigger: "three"
  replace: |
    This is replaced
    just like it is written
    here
```

```
Listing 3: Using Variables
- trigger: ":now"
  replace: "It's {{time}}"

vars:
  - name: time
    type: date

params:
  format: "%H:%M"
```

```

# espanso match file

# For a complete introduction, visit the official docs at: https://espanso.org/docs/

# You can use this file to define the base matches (aka snippets)
# that will be available in every application when using espanso.

# Matches are substitution rules: when you type the "trigger" string
# it gets replaced by the "replace" string.
matches:
  # Simple text replacement
  - trigger: ":espanso"
    replace: "Hi there!"

  # NOTE: espanso uses YAML to define matches, so pay attention to the indentation!

  # But matches can also be dynamic:

  # Print the current date
  - trigger: ":date"
    replace: "{{mydate}}"
    vars:
      - name: mydate
        type: date
        params:
          format: "%m/%d/%Y"

  # Print the output of a shell command
  - trigger: ":shell"
    replace: "{{output}}"
    vars:
      - name: output
        type: shell
        params:
          cmd: "echo 'Hello from your shell'"

  # And much more! For more information, visit the docs: https://espanso.org/docs/

```

**Figure 3:** The `base.yml` file contains some examples. You can extend them with your own replacements or create themed files yourself.

declared it to be a `date` type, and added matching parameters for the output. A similar dynamic example that already exists in `/match/base.yml` gives you the date output. The replacement can be quickly adapted to, say, a UK date format. To do this, change the `format: "%m/%d/%Y"` string to `format: "%d/%m/%Y"`. If you now enter `:date`, you will see the day and month in the typical UK order.

Espanso lets you define both short replacement texts and whole sentences or even longer text blocks. To avoid the standard configuration becoming too confusing, it is a good idea to create several YAML files such as `match/emails.yml` or `match/code.yml` as the number of text modules increases.

## Global Variables

One reason to create a new YAML file is to provide global variables that you can then use in multiple replacements. You define the global variables at the top of a YAML file. They are then available for all replacements defined in this file and its subfiles. Espanso also proves useful if you often mistype certain words, for example, if you often type *aditinal* instead of *additional*. The `true` parameter is used here, as you can see from the detailed documentation [5].

If you like to write code, Espanso will help you with that, too. I'm sure you will find out how to replace the HTML tag pair `<div></div>`, for example, on your own based on what we have looked at so far. But how do you move the cursor into the middle instead of to the end? The `$$` abbreviation is used for this purpose, as shown by the first two lines in Listing 4.

It is also useful to be able to specify multiple replacements with identical triggers using the `quote` parameter (Listing 4, lines 3 to 8). If you enter the `:quote` trigger, you can select which quote you want to use (Figure 4). For example, you can use it to implement multiple salutations in different

## Listing 4: Trigger Examples

```

01 - trigger: ":div"
02   replace: "<div>$$</div>"
03 - trigger: ":quote"
04   replace: "Three can keep a secret, if two of them are dead."
05 - trigger: ":quote"
06   replace: "To be or not to be, that is the question."
07 - trigger: ":quote"
08   replace: "If you want something done right, do it yourself."
09 - trigger: ":cat"
10   image_path: "/PATH/TO/IMAGE.png"

```

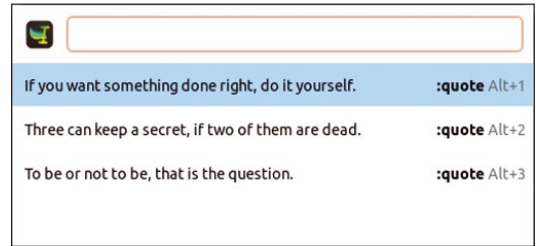
languages. If required, Espanso can also trigger images (line 9 and 10) and create various forms [6].

### Packages

You can reuse replacements created by other users thanks to Espanso's packages strategy. To let you do this, the software comes with its own package manager, which primarily works with the Espanso Hub (Figure 5) on the web, but also accepts other sources. The Espanso Hub website [7] sorts the packages that users have posted there by application or purpose. There is separate documentation [8] for this. In the simplest case, you just install a package by typing the `espanso install PACKAGE` command.

There are other user repositories [9] besides the Espanso Hub. I started by installing the `html-utils` package from the hub. Unlike DIY replacements, two colons trigger the replacement here, as in `::div`. The package configuration can be found in `~/.config/espanso/match/packages`.

I also added the third-party `espanso-config` [9] repository. Installing this is a little more complex. First, you need to clone the repository from GitHub to your own home directory using the command from line 1 in Listing 5. You will then find the `espanso-config/` directory, which you include in Espanso using a symbolic link (line 2). This is where



**Figure 4:** In Espanso, multiple substitutions can have the same trigger. The `:quote` trigger shows a window where you select what you want by pressing a keyboard shortcut.

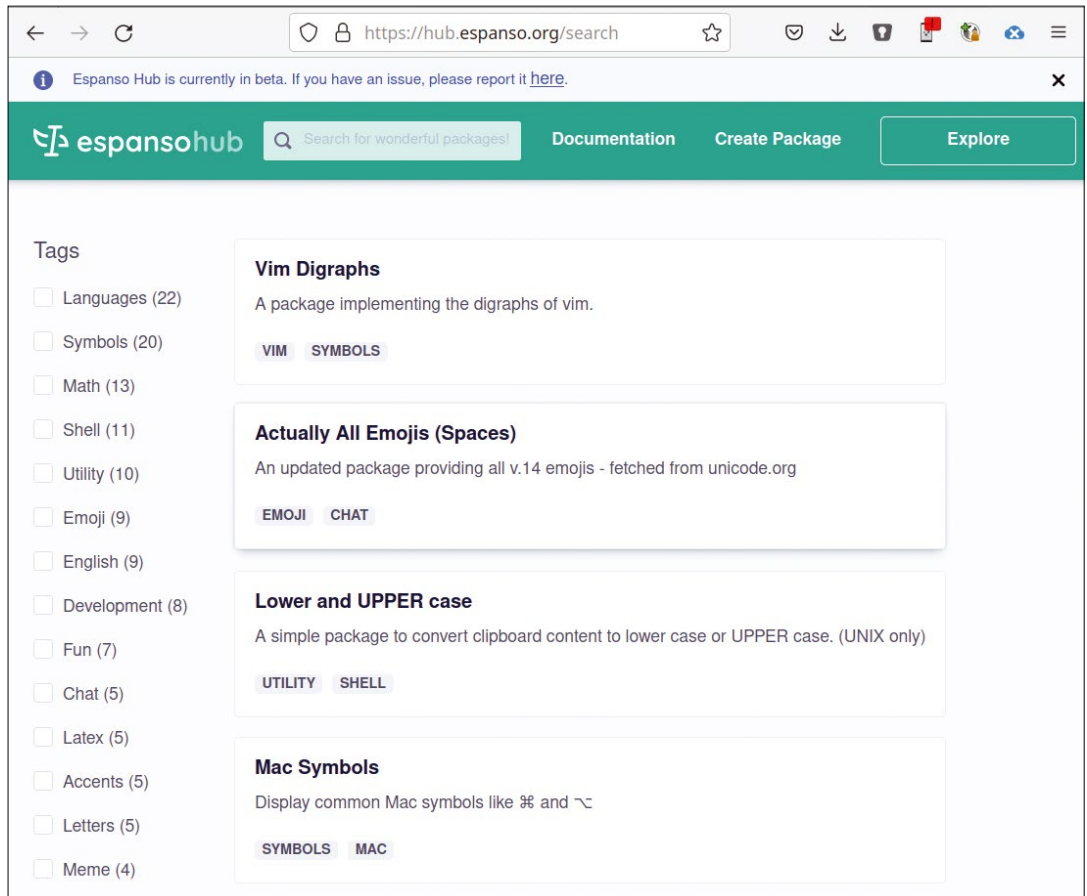
you will find the YAML files with some quite useful replacements.

### Conclusions

Calling Espanso a text expander doesn't do justice to what it provides. The program not only offers more than its competitors in terms of functionality, it also runs on multiple platforms and supports both X11 and Wayland on Linux. The tool comes

#### Listing 5: Including a Repository

```
$ git clone https://github.com/Lissy93/
  espanso-config.git
$ ln -s ~/espanso-config ~/.config/espanso/
  match/packages
```



**Figure 5:** The Espanso Hub web service lets you include packages with replacements on various subjects that have been posted by other users. To do this, Espanso comes with its own package manager.

with excellent documentation, and a configuration, once created, can be synchronized with other systems and platforms [10]. In about one year of use, I have not encountered any errors so far.

Considering the fact that Espanso was first released in September 2019 and that a small team of programmers still develop the tool in their spare time, the huge feature scope, which I only touched on here, is very impressive. And this is by no means the end of the road, as the roadmap shows [11]. A GUI is top of the wish list. It is intended to bring users on board who are wary of using the command line or don't get on with YAML. Following that, a version for Android is on the cards.

### Info

- [1] AutoKey: <https://github.com/autokey/autokey>
- [2] Espanso: <https://espanso.org/>
- [3] Installation: <https://espanso.org/docs/install/linux/#choosing-the-right-install-method>
- [4] Syntax: <https://stackoverflow.com/questions/3790454/how-do-i-break-a-string-in-yaml-over-multiple-lines>
- [5] Documentation: <https://espanso.org/docs/matches/basics/>
- [6] Forms: <https://espanso.org/docs/matches/forms/>
- [7] Espanso Hub: <https://hub.espanso.org/search>
- [8] Packages: <https://espanso.org/docs/packages/basics/>
- [9] Repositories: <https://github.com/Lissy93/espanso-config>
- [10] Synchronization: <https://espanso.org/docs/sync/>
- [11] Roadmap: <https://github.com/espanso/espanso/issues/255>

### The Author

**Ferdinand Thommes** lives and works as a Linux developer, freelance writer, and tour guide in Berlin.

## Shop the Shop

## sparkhaus-shop.com

### Missed an issue?

You're in luck.

Most back issues are still available. Order now before they're gone!

[sparkhaus-shop.com](https://sparkhaus-shop.com)



**GET IT NOW!**  
SAVE TIME ON DELIVERY WITH OUR ALTERNATIVE PDF EDITIONS



# Efficiently manage files with nnn CLI File Manager

If you're a Linux lover, you'll know the command line is the slickest and most efficient way to interact with the system. Free yourself from point-and-click with the nnn command-line file manager.

BY NATE DRAKE

**P**opular distros such as Ubuntu have given Linux to the masses with colorful icons and point-and-click interfaces. Still, experienced Linux users can save huge amounts of time and effort by staying within the command-line interface (CLI).

However, the Linux terminal itself doesn't make it easy to navigate quickly between files, plus you need to memorize a string of commands in order to rename, copy, and move files. This is where nnn comes in. If developer jarun's name sounds familiar, it's because he's also the author of ddgr – a CLI for the privacy-centric DuckDuckGo search engine, which we covered in issue 270 [1].

His utility nnn is inspired by an older minimalist file manager for the terminal, noice, from which it derives its rather recursive name (Nnn's Not Noice).

Like ddgr, nnn cuts out the GUI and allows you to perform operations efficiently. To get started, just fire up your regular terminal and run

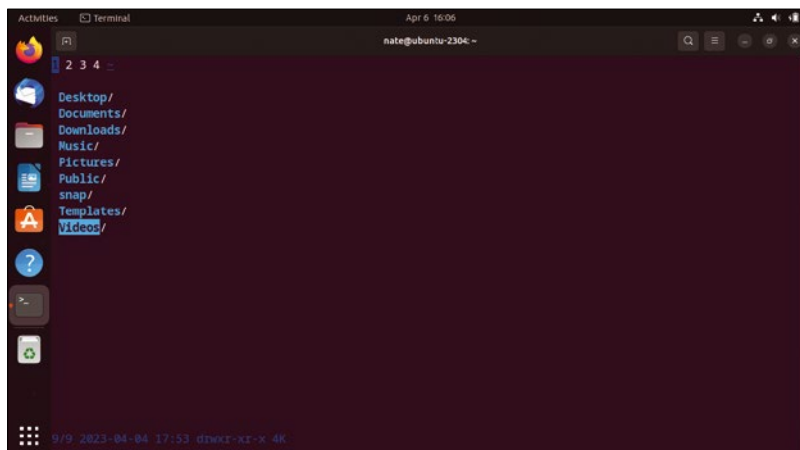
```
sudo apt-get install nnn
```

for Debian-based Linux distros or

```
sudo dnf install nnn
```

for RHEL-based distros.

**Figure 1:** By default, nnn will open to your current running directory (in this case the home folder), unless you specify the PATH variable.



You can now fire up nnn by running nnn from the command line (Figure 1).

## nnn Navigation

Once you fire up nnn, you can start navigating files and folders using the arrow keys. Press right to open a folder and left to return. Alternatively, you can use the same keys as for navigating Vim: *H* (left), *L* (right), *K* (up), and *J* (down).

When browsing files, simply hit enter in order to open a file within its default app. Out of the box, nnn only has limited support for opening files from within the terminal. You can, however, press *e* when a text file is highlighted to display contents. (Use *:wq* when you're done.)

## nnn Names

Once you're comfortable hopping from directory to directory, you may want to perform more advanced file operations.

Press *?* to view a complete list of all available commands. (You can also type *:wq* here to return to the main window.)

This list can seem overwhelming at first, so start simple. During testing, I downloaded a text file of Shakespeare's *Romeo and Juliet*, which Project Gutenberg has given the rather unhelpful filename *rg1513.txt*.

In order to rename it, I simply navigated to the folder in question, highlighted it, and pressed *Ctrl+R*. A prompt appeared at the bottom of the window to enter a new filename (e.g., *romeoandjuliet.txt*).

An interesting quirk of nnn is that you can also copy files and folders to a new destination from here. Just press *Ctrl+R* as if you wanted to rename the file/folder and then press enter (Figure 2). The prompt will now allow you to enter the destination to which you want to copy the file.

## nnn Nominations

Although copying individual files can be useful, most often you'll need to move multiple files or even the contents of entire folders.



This is one area where using a file manager such as Nautilus can be much more convenient than entering multiple terminal commands to select individual files.

Luckily, nnn makes the selection process intuitive. In order to select a file or folder, just right-click on the file/folder or press the space bar (Figure 3). You can do this to multiple files within a folder or just press *A* to select them all. A *+* icon will appear next to selected files.

If you can't find what you're looking for, use */* to open the prompt, which allows you to filter listings by keyword (Figure 4). You can also use *D* to toggle detailed file information such as creation date and size.

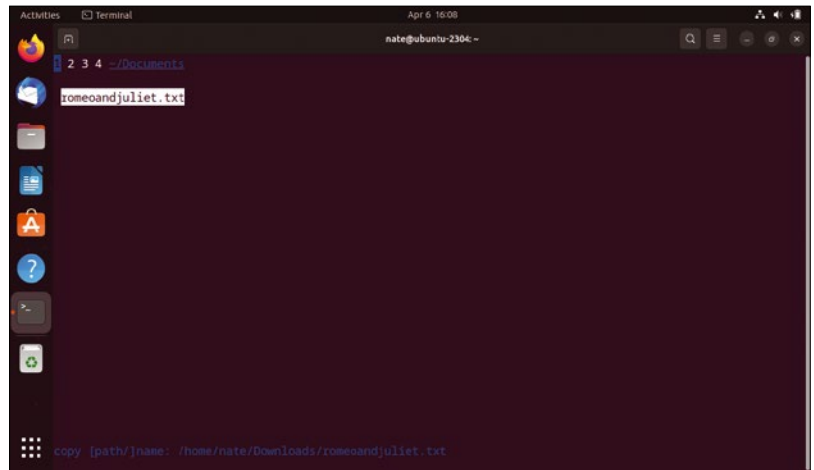
### Making Moves

Once you've made your selection, nnn can use *Ctrl+C* to copy and then *Ctrl+P* to paste to your destination of choice. Note that if you use *Ctrl+C* with a highlighted folder, nnn will, by default, simply paste that folder's contents to the destination folder. To copy the folder and its contents instead, select the folder using space or right-clicking.

If you want to actually move the selected files/folders to the destination in the traditional cut-and-paste way, use *Ctrl+C* and then *Ctrl+V*, instead. In testing, I cut and pasted a folder containing wallpapers from *Pictures* to *Desktop*. This worked fine, except the new folder denied access when we tried to open it, forcing us to use *chmod* to fix the issue.

### nnn Plugins

Although nnn is a very respectable file navigation utility in itself, you can enhance it further by using



**Figure 2:** To copy a file, press *Ctrl+R*, then enter without renaming. You can then type the new path.

plugins (Figure 5). The officially supported plugins are available from nnn's terminal page [2], though there are community options in the wild, too.

To get started with the official plugins, make sure you have *curl* installed, and then run:

```
sh -c "$(curl -Ls https://raw.githubusercontent.com/jarun/nnn/master/plugins/getplugins)"
```

Note that some of these plugins required additional dependencies to work. For instance, if you want to play audio files using the *mocq* plugin, you'll need to install the MOC console audio player. You can check plugin dependencies from the GitHub page.

Whichever plugin you decide to use, you'll need to bind a key to each of them using the environment variable *NNN\_PLUGIN*. For instance, to assign the letter *P* to play tracks using *mocq*, open the terminal and type:

```
export NNN_PLUGIN='p:mocq'
```

**Figure 3:** Use the space bar or right-click to select individual files. Alternatively, just press *A* to select all the files in a folder.

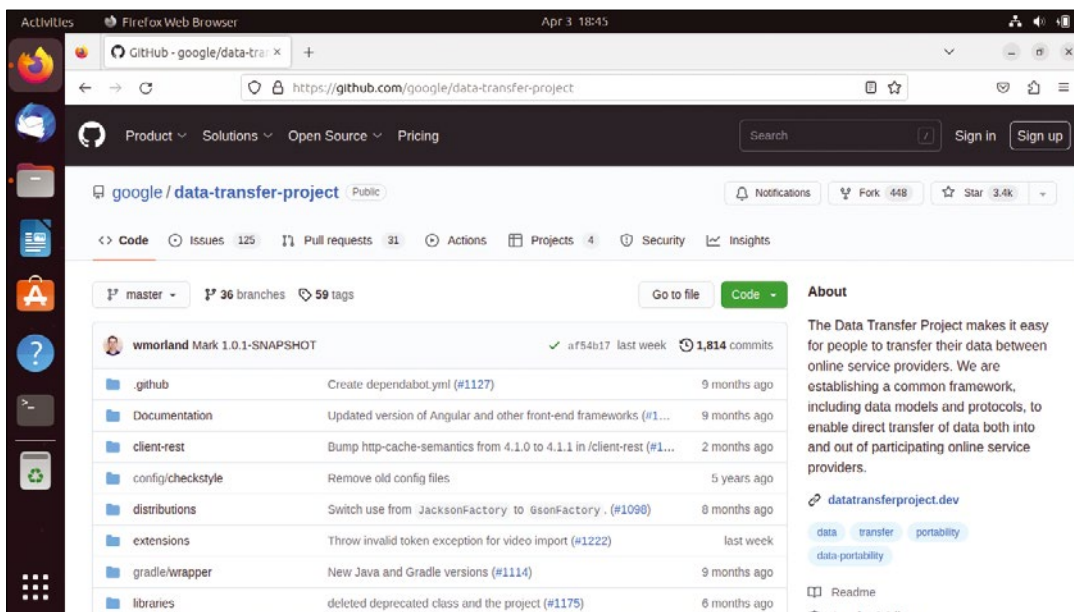
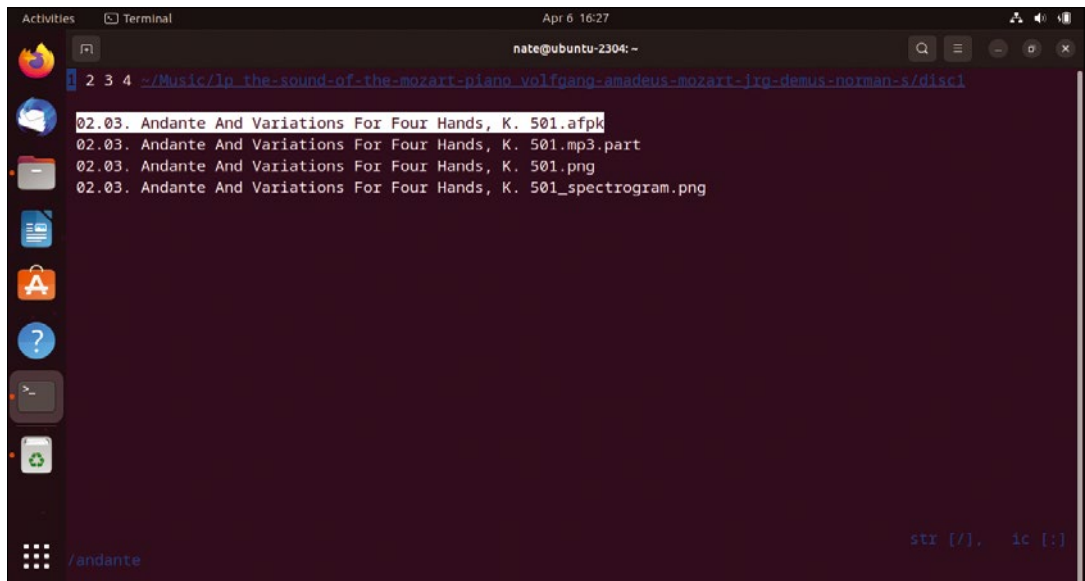


Figure 4: Use / to filter files in a particular folder by keyword. Press Esc to cancel and return to normal navigation.



You can now invoke the plugin by typing either ;p or Alt+P.

The nnn GitHub page claims you can daisy chain keybindings for multiple plugins using a single command, for example:

```
export NNN_PLUGIN='f:finder;o:fzopen;p:mocq;Z
d:diffs;t:nmount;v:imgview'
```

In theory, the command would save you the trouble of entering these keybindings individually, but in fact, I found that plugins would only run when I entered the keybindings one at a time.

You can use the command-line variable P plus your chosen key to start nnn with only that specific plugin enabled (e.g., nnn -P v).

### Fine TuNNning

Unlike many other Linux console utilities, nnn doesn't just have a conventional .conf ig file. It can also be configured via a number of environment variables – as you've seen with NNN\_PLUGIN, which assigns keys to specific nnn plugins.

The Arch Linux help pages [3] contain a detailed list of each of these variables, which you can use to enhance nnn in other ways such as assigning bookmarks, setting the color scheme, and managing archived files.

During testing, I used these variables to change nnn's default setting, which runs rm -rf when you delete any files in nnn via Ctrl+X. This deletes data permanently with no option to restore. To send items to the trash instead, open the terminal and run:

Figure 5: The very handy i mg v i ew plugin requires a utility such as s x i v to display images and set wallpaper.



```
export NNN_TRASH=2
```

You can also use `NNN_BMS` to bookmark a specific folder and then load it in `nnn` via a specific key. For instance, to bind the `V` key to the `Videos` folder, we opened the terminal and entered:

```
export NNN_BMS='v:/home/nate/Videos'
```

Next time you launch `nnn`, you can then press `B` to open bookmarks and then tap your assigned key to jump to that specific folder.

When you press `B`, it will show you any assigned bookmark keys (Figure 6).

### Exploring `nnn` Further

Although we've given you a brief overview, the number of customization options and plugins makes `nnn` virtually infinitely configurable.

If you're starting out, check out the Basic Use Cases section of the GitHub page [4] to learn more about built-in features such as how to configure `cd` on quit and manage images, video, and PDFs. I can strongly recommend working through the Live Previews page [5] to see more information on "hovered" files. The Ubuntu Manual Pages also contain an excellent cheat sheet

of commands [6]. If you run into any trouble, check if your error has come up before on the Troubleshooting page [7]. ■■■

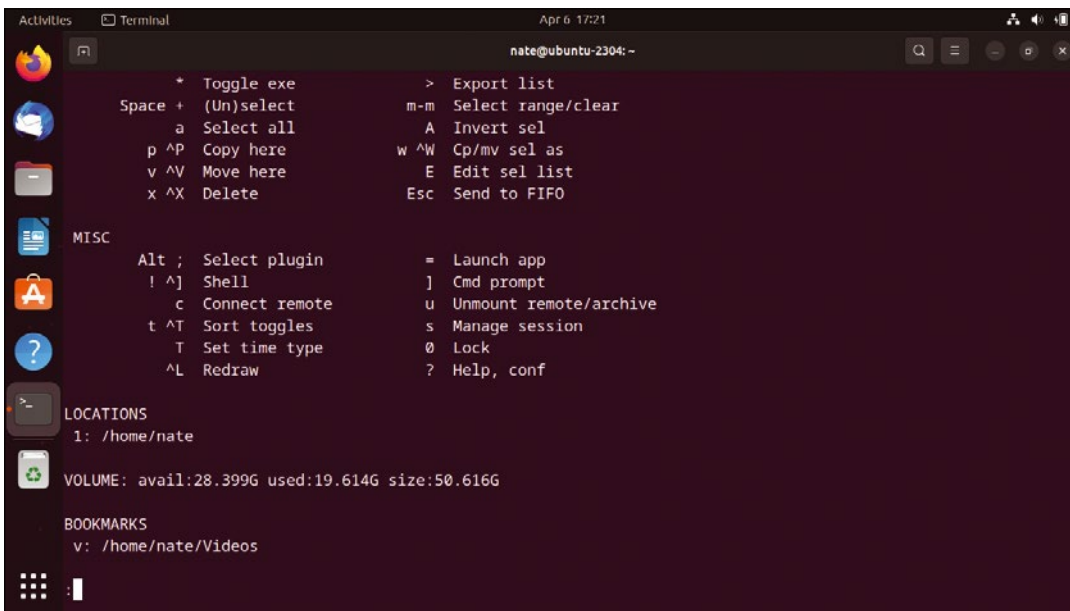
### Info

- [1] "Tutorial – `ddgr`" by Nate Drake, *Linux Magazine*, issue 270, May 2023, pp. 90-94
- [2] `jarun/nnn` on GitHub: <https://github.com/jarun/nnn/tree/master/plugins>
- [3] Arch Linux: <https://man.archlinux.org/man/nnn.1>
- [4] Basic Use Cases: <https://github.com/jarun/nnn/wiki/Basic-use-cases>
- [5] Live Previews: <https://github.com/jarun/nnn/wiki/Live-previews>
- [6] Ubuntu Manuals: <https://manpages.ubuntu.com/manpages/bionic/man1/nnn.1.html>
- [7] Troubleshooting: <https://github.com/jarun/nnn/wiki/Troubleshooting>

### The Author

**Nate Drake** is a tech journalist specializing in cybersecurity and retro tech.

**Figure 6:** You can also press `?` and then scroll to the bottom of the help screen to see all assigned bookmark keys and their destinations.



# FOSSPicks

Sparkling gems and new releases from the world of Free and Open Source Software



With the demise of Twitter API access to the beautiful open source client, Harpy, Graham has spent the month on fosstodon.org, hoping Harpy's developer brings their skills to the Fediverse. **BY GRAHAM MORRISON**

Sewing pattern designer

## Seamly2D

It might initially appear that the worlds of clothes and computers have very little in common. But it doesn't take much investigation to realize almost the opposite is true. Sewing, knitting, and crochet are all popular pastimes, regardless of your background, and in many ways the dextrous sequence of patterns they require is analogous to programming. This was even true over a century before the first computers, in the 1820s, when Jacquard's loom wove silk patterns at an industrial scale using

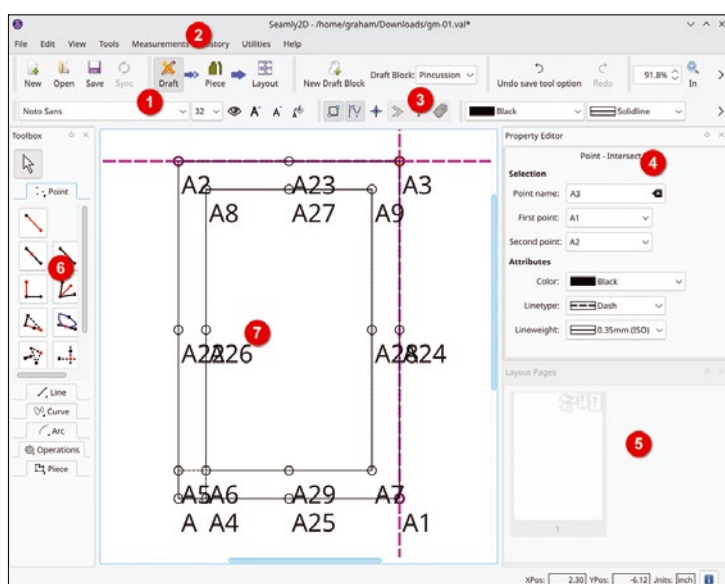
cards with human-programmed punched holes to encode and reproduce a pattern. Since then, computers and clothing have had a long history together, and it's now time for open source to start influencing designs too.

This is what Seamly2D does, making fashion and clothing design accessible to anyone with a computer. Seamly2D is to clothing design what FreeCAD is to mechanical engineers, and like FreeCAD, Seamly2D is a desktop application using the Qt GUI framework. It's also a project

that has recently won a place in GitHub's Accelerator program, which offers \$20,000 and 10 weeks of guidance for each project, plus workshops "with an end goal of building durable streams of funding for their work." Seamly2D describes itself as a "parametric CAD for engineering documents," which sounds complicated, but those engineering documents are clothing designs and they need to be parametric to fit the infinitely variable human form. This doesn't mean that using Seamly2D is easy. It has a learning curve closer to FreeCAD than Figma, but it doesn't require traditional programming skills. Designs are instead crafted from a familiar palette of tools and functions from three different modes of operation: Draft, Piece, and Layout.

The Draft mode is where designers will spend most of their time. It's this mode that has the most icons in the tools palette on the left, and the tools range from the ability to create points and lines at fixed and variable angles, with variable lengths, to curves and arcs with similar properties and simple operations for duplicating, moving, and mirroring groups of points already created. This is the software equivalent to drawing on plotter paper. There's also a property editor on the right for viewing and entering exact values for the points you can't approximate with a mouse, as well as the Inkscape-like features of line colors and weights. There's deep functionality here to help with seams and fitting, and getting good results will take time and patience. Fortunately, there's an active community who will help, a good manual, and various video tutorials to get you started.

From Draft mode a design moves into Piece mode. This efficiently arranges your design elements, or workpieces, in preparation for output much like printing fits images and text onto a piece of paper. Individual workpieces can be exported from here into many different graphical and design formats, including SVG and a huge variety of CAD formats. The step is through the Layout view. This is where the final export will happen for preparing workpieces to cut against material, or to guide the shapes themselves. As a result, the size of a page or sheet can be long, and the Layout view helps you divide sheets into strips, crop them, or unite pages together to make the best use of whatever output you're using. The end result should be the same as those plotter paper drawings, only editable, sharable, and drawn purely in open source software.



**1. Draft, Piece, Layout:** Each pattern is built from separate blocks, laid out into pieces, and exported in a layout. **2. Measurements:** All your designs create exact measurements that can be parametrically scaled. **3. Design:** In many ways, Seamly2D can feel like Inkscape for clothing design. **4. Properties:** Set exact values and attributes. **5. Layout:** See how your designs will look on the page. **6. Point editing:** Most designs are built from points and the lines and curves between them. **7. Main view:** Create your designs from separate blocks.

**Project Website**

<https://github.com/FashionFreedom/Seamly2D>

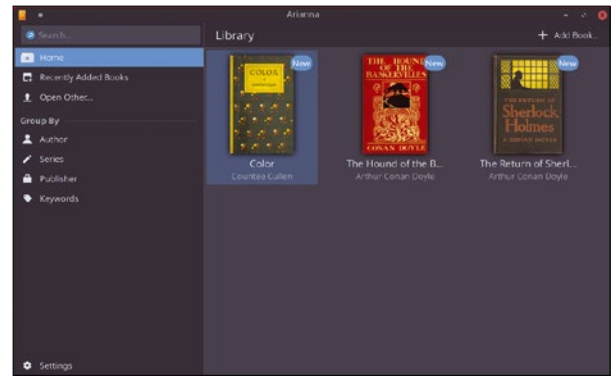
## Ebook reader

# Arianna

**W**e have plenty of image and PDF viewers but we don't have enough ebook readers. This may be because they have specific requirements that aren't common to PDF and image viewers, such as library management and adaptable reading views, and far fewer potential users. Calibre is the exception, because it's an ebook reader and librarian that solves these and many other problems, but it can also be overwhelming and unintuitive when all you want to do is read a book. Which is where this new Arianna application succeeds. It's a small, minimal ePub reader built for the KDE Plasma desktop from Qt and Kirigami, and it's perfect if you have a small collection of books you'd like to read when the

time allows. It won't remove DRM, doesn't require a theme to be installed, and won't send books to your e-ink device, which is the whole point.

The 1.0 release of Arianna comes at an early stage of development, which is why in many ways it can't compete with Calibre. But it can compete on aesthetics, especially for reading. Only ePub files can be imported, after which they're listed in the library view. This will show a thumbnail of the cover, and the library can be sorted by author, by series number, or by keywords attributed to the books in your collection. It's exactly what most of us need. Keywords could be used for "romance," "detective," or "mystery" stories, for example, and selecting a book will open the reader view. This is the best part of Arianna because the text is rendered beautifully, making it very easy to read for long periods even on a monitor. You have



Arianna is a great example of what the beautiful KDE Kirigami framework's capabilities.

control over the font and font size, as well as whether the colors are inverted. This is particularly useful if you use a dark theme, because the text would otherwise be light on a dark background, which isn't too good for light-sensitive or aging eyes. Another great feature is the simple progress bar, which is often missed from ebook readers. This helps you understand where you are in the book, and helps provide you context in terms of your progress, much like feeling pages and their shifting weight does with printed versions.

## Project Website

<https://download.kde.org/stable/arianna/>

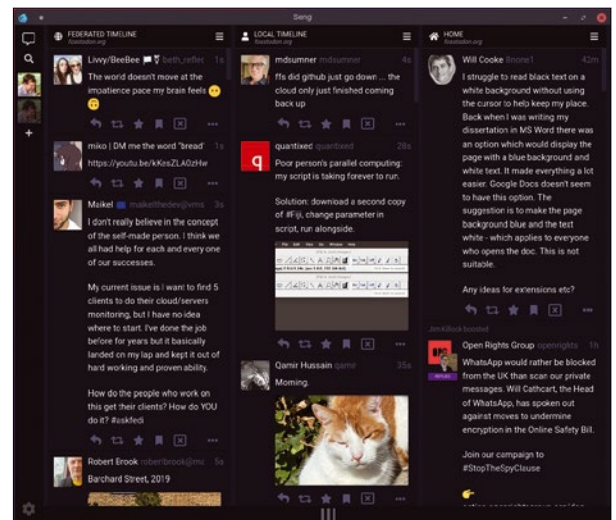
## Fediverse client

# Sengi

**N**ow that the dust has settled (a little) on the great Twitter takeover, with its third-party API closure and the clamor for all things Fediverse, it's a good time to consolidate and look at Mastodon alternatives. Several have been able to seize the momentum behind the recent attention to develop quickly, including the brilliant KDE Plasma client, Tokodon, and Sengi. Sengi is particularly unique because it's a Mastodon client that attempts to recreate the look and functionality of TweetDeck, which was itself an evolutionary window on the world of Twitter. It was TweetDeck's column view that made it so unique and powerful because each column could display the real-time results of a hashtag,

search result, user, list, or timeline. Users could configure their own real-time window on the world in much the same way a stockbroker might track their investments. This is what Sengi does for Mastodon.

TweetDeck was initially a third-party client for Twitter, eventually bought by Twitter, and more recently either quietly abandoned on the desktop or considered ripe for a subscription-only option via a browser. This makes Sengi an important project because it's attempting to bring TweetDeck's unique perspective on social media to the Fediverse, and it's already succeeding. You can add one or more accounts; add columns for your local timeline, your federated timeline, your lists, and searches; and it has live and manual updates. If you love data and would rather follow events, lists, and hashtags than specific users or instances, this is the best way to get the most out of Mastodon.



Sengi also works with Pleroma, a well-established Activity Streams alternative to Mastodon with some unique features of its own.

Being built on Electron, it can feel a little resource intensive, and you can't currently change fonts, adjust the text density, or tweak the theming, but these are all options that can hopefully come later. For now, Sengi is one of the best reasons to jump to Mastodon, and helps the platform to feel like a legitimate Twitter alternative.

## Project Website

<https://nicolasconstant.github.io/sengi/>

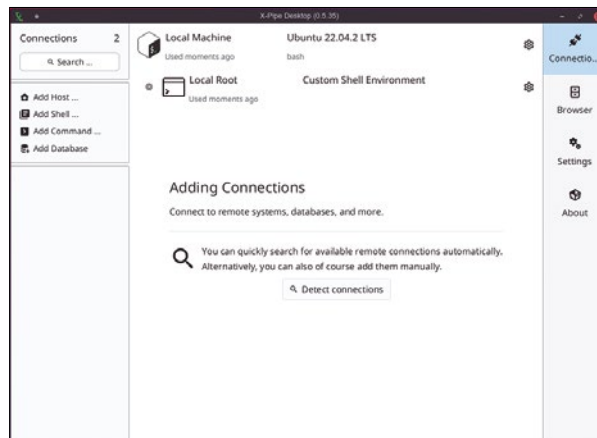
File transfer

# X-Pipe

With so many options for transferring files from one computer or location to another, it's surprising that it can still be a difficult and unpredictable process. Two computers need to support the same protocol or run the same client. They may need to be on the same network, have common access through a firewall, or permit a third-party proxy to negotiate a connection on their behalf. It's complicated. And that's before we've considered operating systems, or even which Linux distribution to run. These are the problems that X-Pipe attempts to solve through either a graphical file manager interface or its command line equivalent, and both have been specifically designed

to make transferring files and directories from one place to another as simple as possible.

X-Pipe takes a clever and pragmatic approach to providing all this seamless connectivity, and it accomplishes this by not really doing anything at all. Instead of trying to negotiate a connection and manage the transfer itself, it uses tools it natively finds on the client's respective system. For Linux users, this means X-Pipe will use SSH, but it will also work on macOS and Windows. The desktop application can automatically detect targets, including Docker and LXD instances, virtual machines, WSL on Windows, and even PostgreSQL databases. Adding any of these will take you to the connection panel, and entered details are securely saved. Hosts can now be selected in the main view, where they'll each appear as tabs, as will your local file-system, which is accessed in the



The only downside to X-Pipe, other than its use of Java, is its confusing licensing, which only releases the core components as open source.

same way. You can now explore the remote filesystem or data storage, copy and paste between tabs, or drag files into the download panel to save them locally. It works well and could be a great option if you need to help non-technical users access certain files across operating systems.

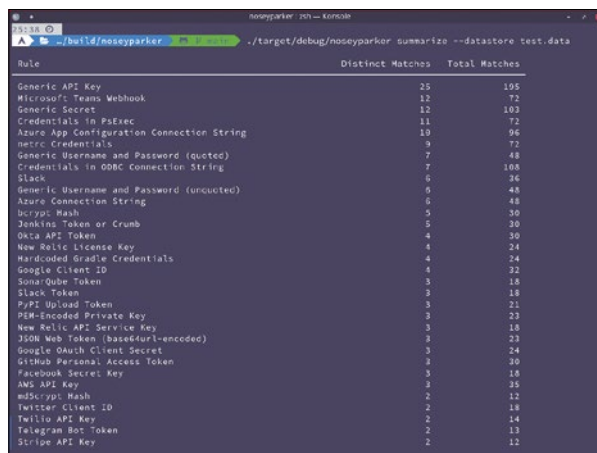
**Project Website**  
<https://github.com/xpipe-io/xpipe>

Secrets searcher

# Nosey Parker

It doesn't take much effort to search Google or GitHub for API keys and other secrets inadvertently shared with the world. This kind of information is often hidden within files or repositories and shared unwittingly along with whatever files or directories were intended. Those details can be very difficult to track unless you never share anything or keep those things you share separated from the rest of your system. Services such as GitHub can help, and GitHub in particular can scan your uploads automatically for shared keys and secrets, even checking them against known providers to check their validity, but it's far better if you can do this yourself locally. This is what Nosey Parker does.

Nosey Parker is a command-line tool that will look through files, directories, and even your Git history looking for anything it considers a secret. Its definitions for secrets are contained within 95 regular expressions, and these can be edited or expanded upon. The first step in performing an audit is to use the scan command with an argument for a local data store, to cache the results, and the destination to search. Nosey Parker can reportedly scan 100GB of kernel source code in less than two minutes on an old Mac. The process is fast, but even a scan of a modest project's Git history can take many seconds. But it takes a lot longer to digest the results, because in our experience, they're scary. Even choosing to scan the Git history



Fascinating and terrifying in equal parts, Nosey Parker will scan your data for the secrets it contains.

of a small Git-based project (we won't say which) revealed "25 unique API keys," including three for an AWS API, and numerous passwords – tests or otherwise. The results are summarized in the output, but you can also use the report argument to dive into a specific result. This will typically include the Git blob, which lines contain the secrets, and the secrets themselves. The results can be terrifying, but used prudently as part of a robust testing system, it could save you or someone else from serious embarrassment.

**Project Website**  
<https://github.com/praetorian-inc/noseyparker>

## Music exploration

# coltrane

**J**ohn Coltrane will need little introduction. He was one of the great jazz composers and performers of the last century and helped transform our expectations of what music can and should be. He did this through both exploration and exposition, experimenting and performing with unique scales and music theory. And it's these last elements of musical study that `coltrane` helps to make accessible in a unique and intuitive way from the Linux command line, regardless of whether or not you enjoy avant-garde jazz. It works as both an interactive interpreter and as a command that takes arguments to generate output of musical insights, and it has several modes of operation (pun intended).

At its simplest, `coltrane` will take three or more note letters and tell you which chord they are. Entering C, E, and G, for example, will output `CM` for C Minor. Chords can optionally be shown on a variety of ASCII-generated representations of guitar, bass, ukulele, and piano finger positions. These representations are very useful if you're learning to play one of the instruments, but they're also excellent for documenting song structures. This is helped by `coltrane`'s ability to also output scales built from the chords you input, as well as find common scales and chords in the scales themselves along with generated chord progressions. All of this can be output as simple text or as multiple ASCII visual



Learn about music theory even when you're on the command line, with the excellently named `coltrane`.

representations, and it looks fantastic if you want to impress anyone with your command-line hacking. There are chord progressions for jazz, blues, and pop, and you can also generate your own custom progressions. If you're at all interested in music, all of this is weirdly compelling, especially from the command line, and can make a useful distraction while you're waiting for some other background process to finish.

## Project Website

<https://github.com/pedrozath/coltrane>

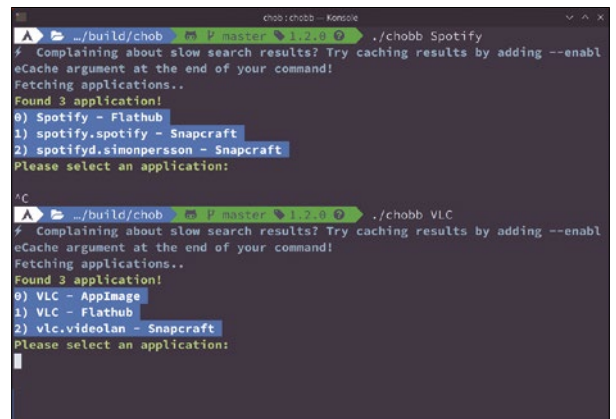
## Meta package management

# Chob

**A**s many of us are fond of saying, Linux is all about choice, and it can be difficult to argue that there's sometimes a little too much choice to wade through when you need one simple thing to get a job done. Package management has become a little like this, with a plethora of cross-distribution solutions offering imperfect coverage of applications, utilities, servers, and tools that might not otherwise be available in your distribution's package manager. You might only be able to install LXDE with Snap packages, for example, or the latest Steam from its Flatpak. And then your favorite game might only provide an AppImage, while your distribution remains two or

three major releases behind. Navigating this set of meta-dependencies can be confusing, and it's this confusion that this brilliant little project tries to solve.

`Chob` is best described as a meta package manager for third-party package management. It currently supports Flatpak for Flatpak packages, Snap Store for Snap packages and AppImages via a search through [appimage.github.io](https://appimage.github.io), which is itself a service that catalogs AppImage packages created by GitHub projects. With the exception of AppImage, you'll also need either Flatpak or Snap to be installed before you start, but `Chob` will handle the package installation itself. It takes a package name



`Chob` can search for and install Flatpak, Snap, or AppImage packages from a single command.

as the single argument, and will present results from all three resources, letting you choose the source you'd prefer. One option we'd love to see is a version comparison of the packages returned, so you can install whichever is the latest version, but we understand comparing version strings across different package formats is going to be difficult. However, if you need a simple tool to help with easy package installation, `Chob` is still definitely worth a look.

## Project Website

<https://github.com/MuhammedKpln/chob>

## Auto effects

# IEM Plug-in Suite

The IEM in the name of this audio effects collection is an acronym for the Institute for Electronic Music and Acoustics. The IEM is a renowned research institute based in Austria, with a focus on acoustics, surround sound, and signal processing, and it's perhaps best known for being closely associated with Pure Data, the functional programming environment for audio synthesis and composition. There's even an IEM-flavored version of Pure Data built specially for its students and faculty, and the institute produces and sponsors a lot of experimental content, some of which is software released as open source – which is exactly what happened with this suite of amazing audio effects that come directly from the institute's current research and student projects.

The suite consists of 21 different effects. They all have a focus on acoustics, and in particular, something called “ambisonics,” which is a specific surround-sound format. This is most evident in the greatest effect in the bundle, the GranularEncoder. It's a form of granular re-synthesizer,

with the granular element being tiny fragments of audio pulled from its stereo inputs. The location of the grain defaults to the beginning of a recording with a duration of 1/4 of a second, but these values are easily modifiable either through the GUI, or through OSC remote automation, which is also true of any of the other plugins in the suite. Grains can be faded in and out, and mixed with the original signal, but the main processing comes from a large circle to represent a top-down view of a three-dimensional sphere. This is the 360-degree ambisonic sound stage, which you use to change where the grains emanate from within a three-dimensional environment: left, right, front, back, up, down, and any position in between. All of this is encoded as surround-sound ambisonics data, which works perfectly well with your plugin host, but will need to be decoded for the positional encoding to make sense in headphones. The included BinauralDecoder plugin is very good for this, even simulating the audio effects caused by that dense clump of mass between your ears.



Apart from surround processing, the IEM effects suite includes compression and equalization that can be used on any music.

The other effects deal with room simulation and reverberation. The RoomEncoder, for example, is the most computationally intensive plugin here because it generates over 200 virtual wall reflections for the audio you position within its virtual room. It sounds incredible, and unnaturally realistic when you map your current room coordinates and listening position to the plugin. The positional energy for all this audio processing can be viewed with the EnergyVisualizer plugin, which is a great way to see how your creations might sound in an environment capable of true ambisonics recreation, and there are equalizers, compressors, and even delays that can process your 3D audio even further. It all sounds incredible, and it's equally amazing that such high-quality processing is available as open source for your favorite Linux audio software for free. The output doesn't just have to be positional 3D audio either, because these effects are also a great way to add subtle ambiance to a podcast, or to reposition one or more acoustic recordings within a simulated live environment, regardless of whether you consider yourself an audio wizard or not. Which is surely something the Institute for Electronic Music and Acoustics would be proud of.



The 3D room simulation, via the room encoder and directivity shaper, sounds amazing in binaural output with headphones.

**Project Website**  
<https://plugins.iem.at>



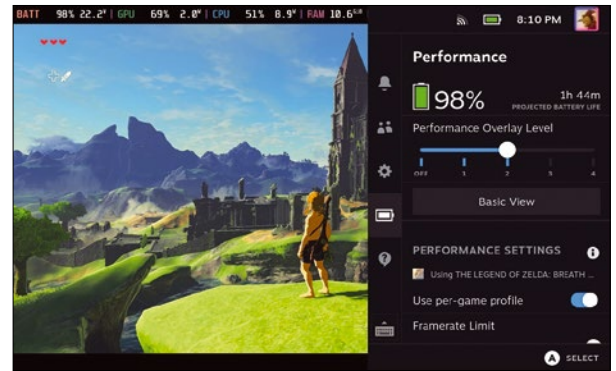
## Wii U Emulator

# Cemu 2

There are dozens or even hundreds of open source console emulators for Linux, from GNU Pong to the thousands of arcade games emulated by MAME. But it's been the relatively recent development of various Nintendo emulators that have really impressed. Dolphin is doing an incredible job with both GameCube and Wii emulation, and it's remarkable how well Yuzu can already run Switch games. It helps that Nintendo's hardware is never cutting edge, and often a revision of its previous generation's technology, but it's still surprising how well games run on even modest hardware like the Steam Deck. Until recently, one piece of the Nintendo emulation stack was

missing from Linux, and that was emulation of the massively underrated Wii U. The Wii U was the successor to the Wii, but it wasn't the success Nintendo hoped for. This was probably due to its large controller with an embedded screen, which made the whole system feel unnecessarily clunky when you still needed the console itself connected to a screen. But that didn't stop Nintendo making some amazing games for the unit, including definitive Zelda, Mario, Mario Kart, and Super Smash Bros. milestones.

If you own the originals, and are prepared to go through a somewhat convoluted extraction process, they can now all be played on Cemu, an emulator that until recently had been both Windows-only and closed source. The emulator has been around since 2015 and can already run many of the most popular games with excellent



The latest experimental versions of Cemu include Ubuntu and Appliance x64 Linux binaries.

performance, but the popularity of the Steam Deck has prompted its developers to create a native port, despite the Windows version working well with Proton. The native Linux release still has some catching up to do, but it already runs well and is compatible with all the mods and graphics files of the original. If you're fortunate to have access to the original hardware and a decent games collection, it's become the best way to play them either on the go, or in higher resolutions and frame rates when your native Linux hardware is up for the job.

**Project Website**

<https://cemu.info>

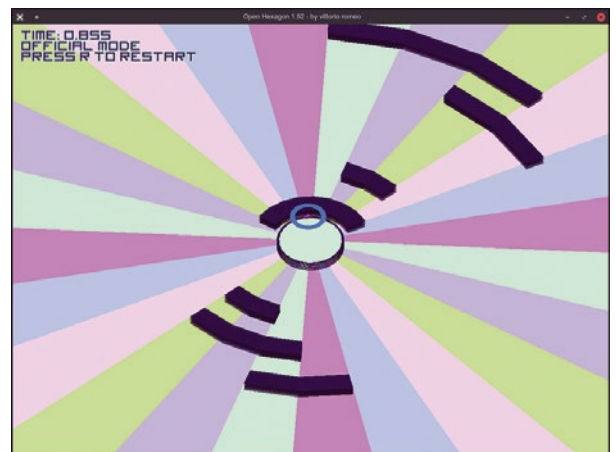
## Arcade survival

# Open Hexagon

Many of us idled hours and days away in our younger years playing video games. Life seemed to stretch forever, and a Sunday wasted collecting virtual acorns seemed like a good use of our time. One of the best solutions to adult commitments getting in the way of such abandon is to find a short, addictive, and intense arcade-style game to play when no one is looking, and one of the best games of this type is Super Hexagon. Super Hexagon was created by Terry Cavanagh from crude, simplistic two-dimensional blocks that converge into or expand from the center of the screen, accompanied by a thumping soundtrack. The blocks converge in such a way that there's

always a gap, and your job as the player is to spin a small triangle through these gaps for as long as you can. If you're any good, you'll last a few seconds. Any more than that, and you're either very young or you've had too much caffeine (or both). It's intense, addictive, and probably not a great idea if you suffer from epilepsy.

Open Hexagon is an open source recreation of the same game. Like the original, it's been available for over a decade but has also been endorsed by Terry Cavanagh. Unlike the original, however, Open Hexagon has just been updated to take advantage of the Steam Deck's growing popularity. This is a version that can be purchased on Steam, but you remain free



Open Hexagon is a sugar rush of colors and gameplay, undiluted by the years since its original release.

to build your own binaries from the project's GitHub repository, and it's definitely worth the effort. Open Hexagon is the original experience, and it still looks beautiful. It's a great match for the Steam Deck, but it's equally fun for a quick blast on your desktop and a game that feels just as timeless as it did when originally released.

**Project Website**

<https://github.com/vittorioromeo/SSVOpenHexagon>

# Graphical web browsing from the terminal

# Light Browsing

This Chromium port can run inside any console, with minimal resources, and is a great tool for making old computers really useful – and learning programming along the way.

BY MARCO FIORETTI

Carbonyl [1] is a fork of the 100-percent open source version of Chrome called Chromium [2], which has a unique mission: turning any Linux terminal into a modern graphical web browser. In this tutorial, I describe how Carbonyl works, how to use it on Linux, and above all, why I consider this project interesting for programmers – and much more interesting for ordinary web users.

## Features and Limitations

Let me acknowledge right away that, as cool as the concept is, Carbonyl has a few serious limitations – if you compare it to “ordinary” 100-percent graphical browsers, that is, and expect Carbonyl to replace them completely. The first obvious limit is that, while Carbonyl does handle images and video streaming, it does it at a much lower resolution than any browser running on top of a full-fledged, window-based interface.

Figure 1 is a good visual introduction to both the power and the limits of Carbonyl because it shows how the text browser w3m (left) and

Carbonyl (right) render a page of the *Linux Magazine* website.

Yes, the actual images are quite grainy and pixelated, no question about that. But overall, there’s no doubt that Carbonyl renders the web page much better, and much closer to what you would see in Firefox, Chrome, or Safari, than its text-only competitor w3m does in the left side of the figure. The fact that Carbonyl, while running in a Linux terminal, is much closer to “real” graphical browsers than to text-based ones is even more evident in Figure 2.

What you see in Figure 2 is the YouTube homepage loaded, left to right, inside w3m, Carbonyl, and good old Firefox (w3m and Carbonyl are running in two panes of the tmux terminal multiplexer [3]). YouTube is both recognizable and usable in the two rightmost windows, but no in the first one.

Running in the terminal, however, does not necessarily mean that all images will always be impossible to recognize, or that all video clips will be always impossible to follow. As proof of this statement, I offer Figure 3, which is a comedy skit on Italian moms that I was able to follow on YouTube with Carbonyl.

The left side of Figure 3 also shows the downside of what is by far the easiest “hack” you can use to make video inside Carbonyl viewable (at least for the moment): If you reduce the font size of the terminal that contains Carbonyl so much that text becomes really unreadable, video will be intelligible. Luckily, much better solutions are in the works (more on this below).

The second big limit of Carbonyl, which may be a showstopper for some users and a nonissue for others, is lack of support for Chrome/Chromium extensions. The third, which again may be an issue only if Carbonyl had to be your only browser, is the lack of bookmarks. However, because the developers’ explicit goal is to integrate the complete Chromium user interface, both bookmarks and extensions should eventually be available in Carbonyl.

If you ask me, however, Carbonyl already more than makes up for these hopefully temporary

**Figure 1:** This is how your favorite Linux website looks in purely text-based browsers (left) and in Carbonyl (right).



limits with an excellent, probably unparalleled compromise between “weight” and performance, especially if you count useful performance. This browser supports all the modern web standards that matter for the overwhelming majority of users of graphic browsers, from JavaScript to WebGL, WebGPU, and audio and video playback (at 60 frames per second) and does so without carrying along the big ballast that is graphic servers such as X.Org or Wayland.

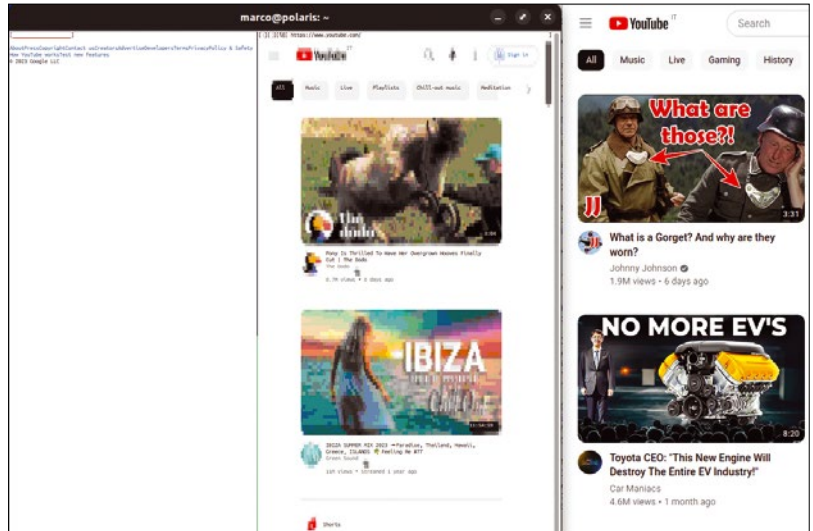
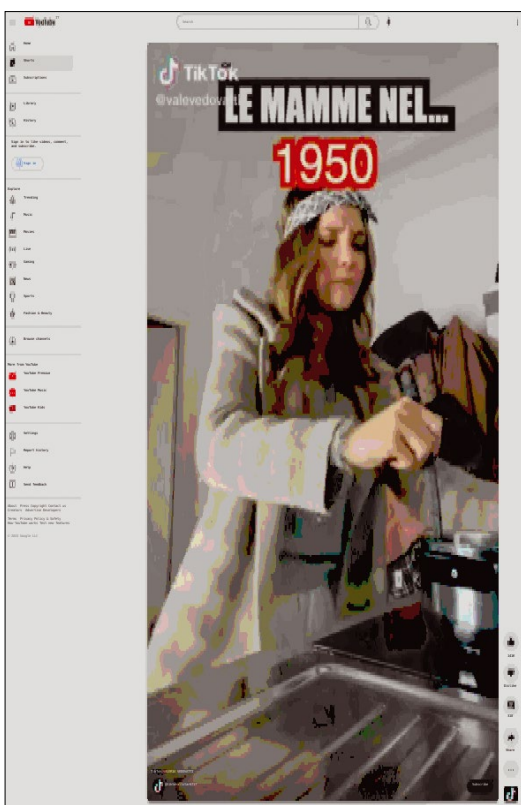
On my desktop, with 8GB of RAM, Carbonyl starts and displays complex web pages that look just like they would in Firefox, except for the quality of images, in a couple of seconds. Even better, it consumes almost no CPU cycles when it is idle. I could not test this personally, but its developers claim Carbonyl is usable even remotely over SSH, on platforms as lean as a Raspberry Pi.

### The Case for Programmers

The first reason why every programmer may want to play with Carbonyl is obviously the same as the great mountaineer George Mallory had for climbing Everest: “Because it’s there!” Or, as one of the first users of Carbonyl puts it [4], “if a full browser can be supported from the command line, what else is possible?”

A more pragmatic reason to participate in the development of this unusual browser may be career building. Carbonyl seems like an

**Figure 3:** Nobody would define this comedy skit high resolution, but it was clear enough to get all the jokes.



**Figure 2:** Carbonyl is close enough to a fully graphical web browser that you almost forget it’s running in a terminal!

excellent playground for learning lots of technologies and algorithms whose existence and relevance are unknown to many millions of people who depend on them every day (see the “Carbonyl Technologies” box).

### The Case for End-Users

I consider Carbonyl a great way, if not the best-possible solution, to deal with a certain serious issue that plagues many “ultra-light” Linux distributions made to bring old hardware back to life and keep it out of the landfills.

Sure, thanks to those projects, you can install and run Linux on computers where no other operating system, including most current Linux distributions, could ever boot, or even fit in the first place. But then what?

As cool and efficient as those tiny Linuxes are, any computer running them would be practically useless for most newbies and expert Linux users, for one simple reason: These days, most of the

### Carbonyl Technologies

Carbonyl is written in languages such as TypeScript, C++, and Rust, which would not look bad on any curriculum these days. The basis runtime used for rendering inside a terminal is the `html2svg` utility [5], which was originally developed to convert HTML pages, including graphics generated on the fly in Canvas elements, to vector or bitmap images.

Carbonyl also uses, or plans to use, separate software rendering for text and images in the same window, including really obscure stuff (for me, at least) such as “quadrant binarization,” which is a technique to render four visible pixels per terminal cell. At a higher level, playing with Carbonyl source code could be a good way to learn how to use and develop the Chromium profiles for Profile Guided Optimization (PGO) [6].

Last but not least, Carbonyl may teach many tricks about how to remove the cruft that plagues so many other FOSS packages. The first versions of Carbonyl had, just like ordinary Chromium, about 20 complex dependencies to deal with. The version current at the time of writing (Carbonyl 0.0.3) only has four. Not bad, is it?

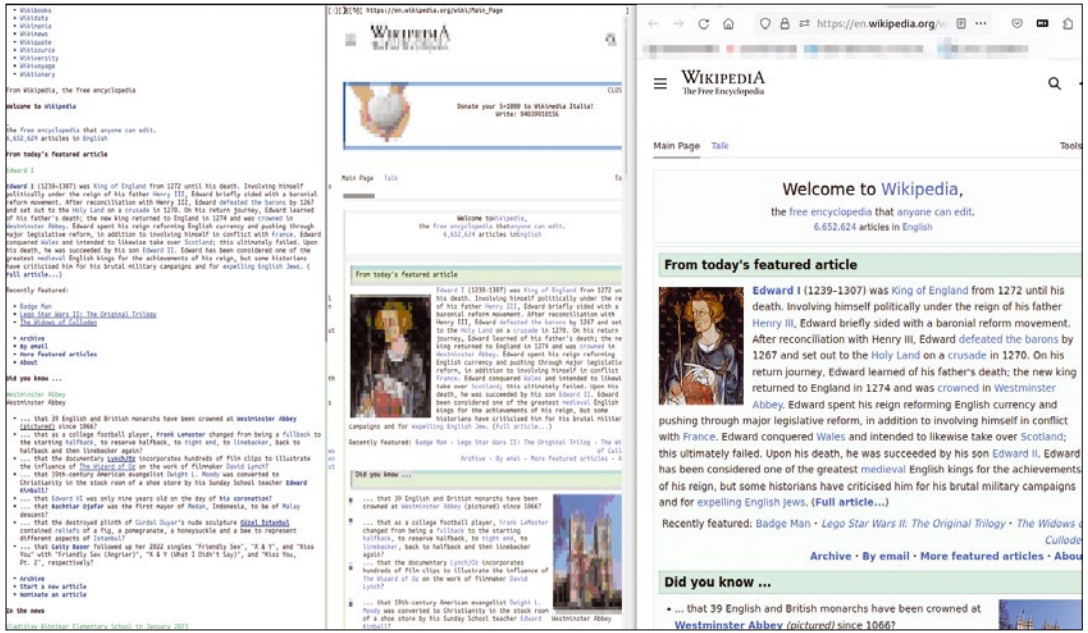


Figure 4: The same browsers from Figure 2, now loaded with the Wikipedia homepage.

things one needs to do with a computer can only happen online, inside a heavyweight graphical browser with lots of bells and whistles turned on.

In other words, certain distributions may be lighter than a feather, but without a browser that can really handle certain websites without bringing old hardware to its knees, very few people would find them useful. But this, as I hope I've already started to prove, is exactly the gap that Carbonyl may fill big time.

The reason is obvious: Normal GUI-based browsers cannot run without a windowing system underneath – something that may make low-end processors with little RAM choke, before you even try to start any GUI program. At the opposite

corner of the browser market, text-based applications such as Lynx or w3m are great, but let's be realistic: Even if their lack of support for media playback didn't matter, they just cannot cope with modern, interactive websites.

For example, I cannot use w3m, Lynx, or anything like that to make online payments through my bank's website. Very likely, this is the bank's fault, not the browsers', but that doesn't make any practical difference: Either I use a GUI browser, or I must visit the closest physical branch to pay my bills.

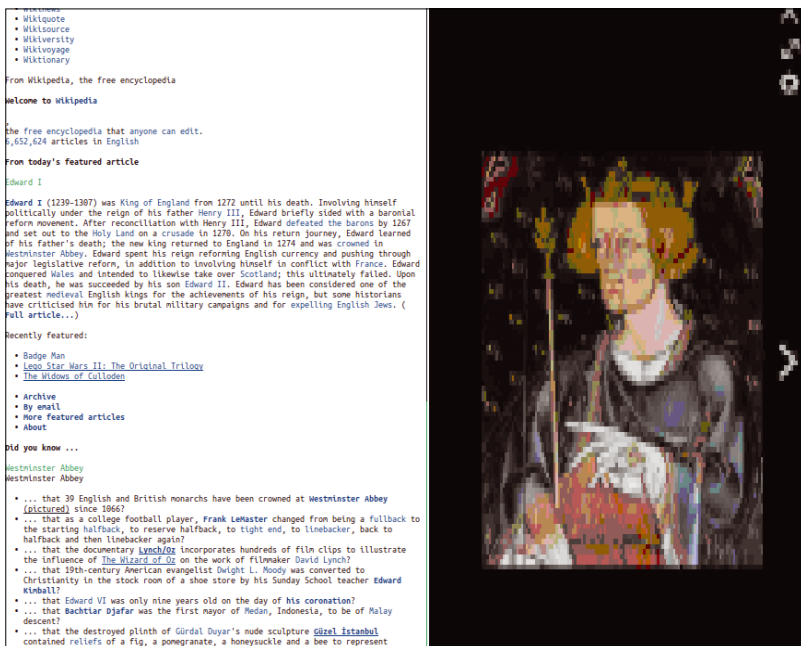
With Carbonyl many of these problems disappear, and even more will do the same, if Carbonyl maintains its promises. Sure, images and videos look awful inside Carbonyl, but does it really always matter? To answer, please look at Figures 4, 5, and 6. The first contains the homepage of the English-language Wikipedia in the same three browsers from Figure 2 (left to right, w3m, Carbonyl, and Firefox).

The other two figures show the real power of Carbonyl, which is that it lets you browse state-of-the-art websites with what is probably the smallest possible load on the hardware, but in the same exact way you, and everybody else who ever surfed the Internet since the 1990s, already know! In Figure 5, I could see the full-size version of the picture in the featured Wikipedia article just by clicking on it.

What you see in the right side of Figure 6, instead, is the pop-up window that appears when you mouse over any link to other Wikipedia pages.

Try doing that in w3m or Lynx! Of course I do not mean to disparage in any way those browsers and their developers here! Those are very different projects, with different goals and

Figure 5: In Carbonyl, you can click on pictures to see them full-size.



plenty of real-world use cases, not competitors of Carbonyl in any way.

The fact is, with Carbonyl you can scroll, point, and click whatever you want with your mouse. Including unavoidable annoyances such as GDPR notifications, cookie configuration windows, pop-ups to accept Terms of Use, or superimposed banners. With just a bit more resolution, even picture-based CAPTCHAs may be manageable.

As one last example of Carbonyl's usability, let me explain Figure 7, which is the same figure I used in another article in this issue of *Linux Magazine* [7]. The original purpose of that figure was to show that anyone can embed interactive, very complex web forms to analyze online databases in any web page using `iframe` HTML elements. In Figure 7, the web page whose source code appears on the left is visualized in Firefox, with its drop-down menu used to select certain columns of the database. Figure 8 shows that if you load the same page in Carbonyl, the embedding and all the elements of the interactive web form, including its zoomable maps tab not shown in Figure 7, will still work!

Through these and other tests, I have found that Carbonyl, while far from being a finished product, already makes complex websites usable just like a non-savvy web user would expect from a "normal" browser, with very satisfying performances on the most common Linux terminals. According to its website, there are people who managed to play Doom on Carbonyl and to use services such as Gmail and Slack.

Carbonyl's low graphical resolution may even be a bonus by helping user's avoid distractions, especially in environments such as school labs or libraries, and I am sure there are many other such opportunities. When I asked on the Carbonyl GitHub page, for example [8], another user reported that his university is using Carbonyl and `rc` (a command-line program to manage files on cloud storage, [9]) to "enable users to mount their OneDrive using a fancy script that behind the covers uses `rc` [because] the simplest method to enable the drive is to use a web-browser to grant the app access through Microsoft's login portal. It's a clunky solution but it works thanks to Carbonyl!"

### Install and Use Carbonyl

On Linux, Carbonyl is available in two formats: an executable file and a Docker container. The first is a single file without any dependency, available for AMD64 and ARM64 systems. To use it, you should download the corresponding zipped archive from the website, unpack it in a directory of your choice, and then add that directory to your path, in order to launch Carbonyl from the prompt, like any other command-line program.



**Figure 6:** With Carbonyl, your mouse will work just like in Chrome or Firefox.

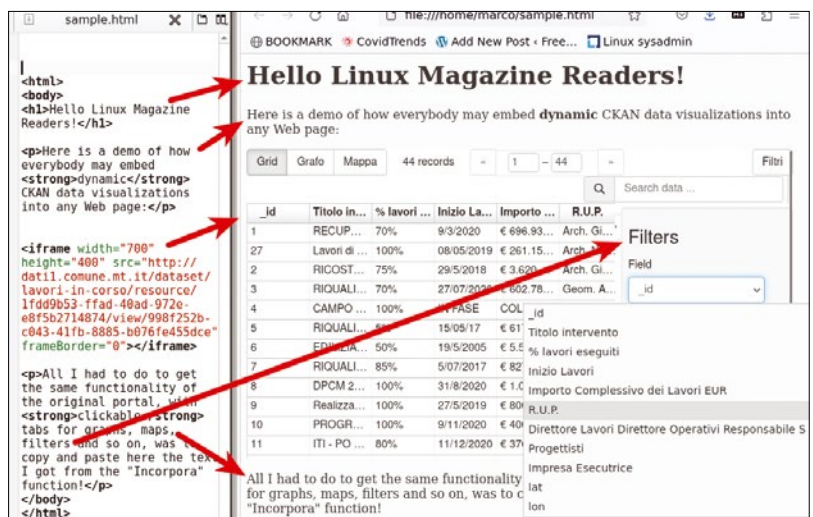
To use the container instead, first install whichever version of the Docker container manager is available for your Linux distribution (I used Docker Engine [10]), then at the prompt type:

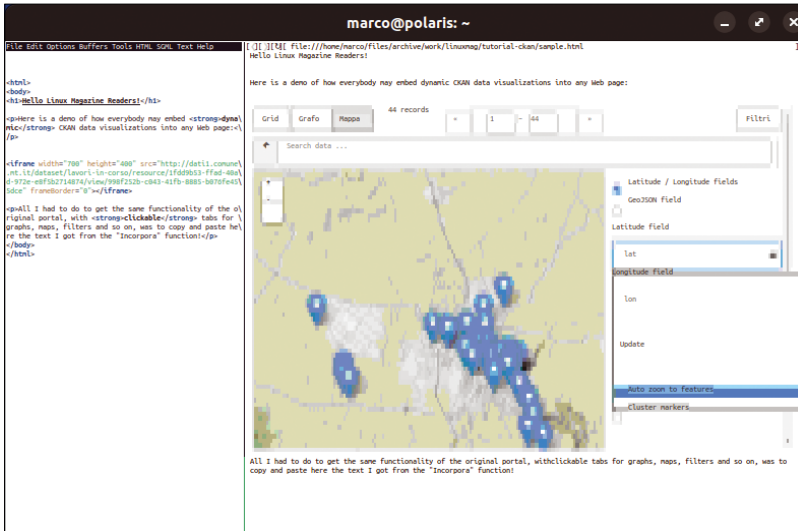
```
docker run -ti fathyb/carbonyl SOME_WEBSITE
```

The `SOME_WEBSITE` part – the full URL of the first website to visit – is absolutely optional. If you forget it, click with your mouse at the top of the terminal where the `about: blank` message is, cancel it, and type the URL there. Alternatively, you may define an alias to make Carbonyl always start with a search engine so that you can input any URL or other strings to search in the corresponding search bar.

After that, you can follow links as you want and just type `Ctrl+C` when it's time to quit the browser. The only other thing to keep in mind is that Carbonyl supports most Chromium command-line

**Figure 7:** A web page's source code (left) and the page itself shown in Firefox (right).





**Figure 8:** The web page from Figure 7 loaded in Carbonyl – its embedding and interactive web-form elements all work!

options [11], plus some of its own. The ones you must know in the latter group are `--version` and `--help`. `--version` displays which version of Carbonyl you are using, which is the first information you will need to provide should you ask for support. `--help` will display all the options available, and I suggest trying it at least once, to check what they are.

Other options to try are `--bitmap`, which disables terminal text rendering, and `--zoom`, which

sets the zoom level as a percentage (e.g., `--zoom=200` would make everything twice as big). I must report that no combination of those flags made any difference on my computer, running the i3 window manager. However, by the time you read this, the situation may be very different.

### Conclusions

Carbonyl is a browser that lets you use highly dynamic websites, in the same way you already know, while consuming much less hardware resources – and possibly with fewer distractions. On top of all that, it is also very easy to install. So just try it, I say! ■■■

### Info

- [1] Carbonyl: <https://github.com/fathyb/carbonyl/>
- [2] Chromium: [www.chromium.org/Home](http://www.chromium.org/Home)
- [3] tmux: <https://github.com/tmux/tmux/wiki>
- [4] “Command Line Browser Carbonyl,” Nextpertise: [https://nextpertise.net/posts/230305\\_carbonyl/](https://nextpertise.net/posts/230305_carbonyl/)
- [5] html2svg: <https://fathy.fr/html2svg>
- [6] Chromium PGO Profiles: <https://blog.chromium.org/2020/08/chrome-just-got-faster-with-profile.html>
- [7] “Managing Open Data with CKAN” by Marco Fioretti, *Linux Magazine*, issue 272, July 2023, p. 20
- [8] “Real-world” usage of Carbonyl, GitHub: <https://github.com/fathyb/carbonyl/issues/153>
- [9] rclone: <https://rclone.org/>
- [10] Install Docker Engine on Ubuntu: <https://docs.docker.com/engine/install/ubuntu/>
- [11] Chromium command-line options: <https://peter.sh/experiments/chromium-command-line-switches/>

### The Author

**Marco Fioretti** (<http://mfioretti.substack.com>) is a freelance author, trainer, and researcher based in Rome, Italy, who has been working with free/open source software since 1995, and on open digital standards since 2005. Marco also is a board member of the Free Knowledge Institute (<http://freeknowledge.eu>).



# LINUX NEWSSTAND

**Order online:**

<https://bit.ly/Linux-Newsstand>

*Linux Magazine* is your guide to the world of Linux. Monthly issues are packed with advanced technical articles and tutorials you won't find anywhere else. Explore our full catalog of back issues for specific topics or to complete your collection.



**#271/June 2023**

## Smart Home

Smart home solutions will save you time and energy – and, did I mention, you can amaze your friends. This month we show you how to take charge of your home environment with smart devices and open source automation software.

**On the DVD:** SystemRescue 10.0 and Linux Lite 6.4

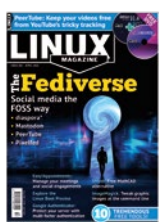


**#270/May 2023**

## Green Coding

A sustainable world will need more sustainable programming. This month we tell you about some FOSS initiatives dedicated to energy efficiency, and we take a close look at some green coding techniques in Go.

**On the DVD:** Fedora 37 Workstation and TUXEDO OS 2



**#269/April 2023**

## The Fediverse

Social media tools connect the world, bringing us the latest news and commentary from politicians, movie stars, community leaders, and remote friends. But the tracking and data mining of the commercial social media platforms has left many users searching for a better option. This month we dive down into the alternative universe for social media users: the Fediverse.

**On the DVD:** EndeavourOS Cassini 22.12 and Debian 11.6 “bullseye”

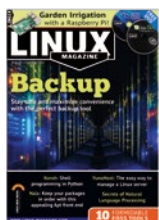


**#268/March 2023**

## Data Poisoning

Think computers don't make mistakes? If you slip some doctored-up training samples into the mix, you can get a fancy machine-learning system to think a dog is a cat or a 1 is an 8 – and you can trigger this bad behavior through a hidden signal no one else will notice.

**On the DVD:** MX Linux 21.3 and Puppy Linux FossaPup 9.5



**#267/February 2023**

## Backup

In theory, everyone could use the same backup tool, but the best way to build regular and systematic backups into your life is to find a solution that fits with your own habits and methods. This month, we preview some popular backup apps in the Linux space so you can find one that works for you.

**On the DVD:** Linux Mint 21 Cinnamon and Kali Linux 2022.4



**#266/January 2023**

## Generative Adversarial Networks

What is the secret behind the recent explosion of computer art and fake videos? One neural network lies to another neural network...

**On the DVD:** Ubuntu 22.10 and AlmaLinux 9.0

# FEATURED EVENTS

Users, developers, and vendors meet at Linux events around the world. We at *Linux Magazine* are proud to sponsor the Featured Events shown here.

For other events near you, check our extensive events calendar online at <https://www.linux-magazine.com/events>.

If you know of another Linux event you would like us to add to our calendar, please send a message with all the details to [info@linux-magazine.com](mailto:info@linux-magazine.com).



## Akademy 2023

**Date:** July 15-21, 2023

**Location:** Thessaloniki, Greece & Online

**Website:** <https://akademy.kde.org/2023/>

Akademy is the annual world summit of KDE, one of the largest free software communities in the world. This is a free, non-commercial event organized by the KDE community. The conference is expected to draw hundreds of attendees from the global KDE Community. Akademy features a 2-day conference followed by 5 days of workshops, Birds of a Feather (BoF), and coding sessions.

## GUADEC 2023

**Date:** July 26-31, 2023

**Location:** Riga, Latvia

**Website:** <https://events.gnome.org/event/101/>

GUADEC is the GNOME community's largest conference. Join us in Riga, Latvia as we bring together hundreds of users, contributors, community members, and enthusiastic supporters for a week of talks and workshops.

## RustConf 2023

**Date:** September 12-15, 2023

**Location:** Albuquerque, New Mexico and Online

**Website:** <https://rustconf.com/>

RustConf gathers Rust developers from around the world to learn and share with one another. Join us for 4 days of talks and trainings from top Rust developers, discord rooms for live discussion, digital and in-person discussions, and more. Don't miss the largest annual gathering of the Rust-language community.

## Events

Akademy	July 15-21	Thessaloniki, Greece & Online	<a href="https://akademy.kde.org/2023/">https://akademy.kde.org/2023/</a>
EuroPython 2023	July 17-23	Prague, Czech Republic + Virtual	<a href="https://ep2023.europython.eu/">https://ep2023.europython.eu/</a>
GUADEC	July 26-31	Riga, Latvia	<a href="https://events.gnome.org/event/101/">https://events.gnome.org/event/101/</a>
WeAreDevelopers World Congress	July 27-28	Berlin, Germany	<a href="https://www.wearedevelopers.com/world-congress/">https://www.wearedevelopers.com/world-congress/</a>
RustConf 2023	Sep 12-15	Albuquerque, New Mexico	<a href="https://rustconf.com/">https://rustconf.com/</a>
stackconf 2023	Sep 13-14	Berlin, Germany	<a href="https://stackconf.eu/">https://stackconf.eu/</a>
EuroBSDCon 2023	Sep 14-17	Coimbra, Portugal	<a href="https://2023.eurobsdcon.org/">https://2023.eurobsdcon.org/</a>
Storage Developer Conference (SDC'23)	Sep 18-21	Fremont, California	<a href="https://storagedeveloper.org/">https://storagedeveloper.org/</a>
All Things Open	Oct 15-17	Raleigh, North Carolina	<a href="https://www.allthingsopen.org/">https://www.allthingsopen.org/</a>
DrupalCon Lille 2023	Oct. 17-20	Lille, France	<a href="https://events.drupal.org/lille2023">https://events.drupal.org/lille2023</a>
LinuxFest Northwest 2023	Oct 20-22	Bellingham, Washington	<a href="https://linuxfestnorthwest.org/">https://linuxfestnorthwest.org/</a>
Hybrid Cloud Conference	Oct 26	Virtual Event	<a href="https://www.techforge.pub/events/hybrid-cloud-congress-2/">https://www.techforge.pub/events/hybrid-cloud-congress-2/</a>
SeaGL 2023	Nov 3-4	Virtual Event	<a href="https://seagl.org/">https://seagl.org/</a>
Open Source Monitoring Conference (OSMC)	Nov 7-9	Nuremberg, Germany	<a href="https://osmc.de/">https://osmc.de/</a>
SFSCON 2023	Nov 10-11	Bolzano, Italy	<a href="https://www.sfscon.it/">https://www.sfscon.it/</a>
SC23	Nov 12-17	Denver, Colorado	<a href="https://sc23.supercomputing.org/">https://sc23.supercomputing.org/</a>
Cassandra Summit	Dec 12-13	San Jose, California + Virtual	<a href="https://events.linuxfoundation.org/cassandra-summit/">https://events.linuxfoundation.org/cassandra-summit/</a>



**Contact Info****Editor in Chief**

Joe Casad, jcasad@linux-magazine.com

**Copy Editors**

Amy Pettie, Aubrey Vaughn

**News Editors**

Jack Wallen, Amber Ankerholz

**Editor Emerita Nomadica**

Rita L Sooby

**Managing Editor**

Lori White

**Localization & Translation**

Ian Travis

**Layout**

Dena Friesen, Lori White

**Cover Design**

Dena Friesen, Lori White

**Cover Image**

© Gloria Sanchez, 123RF.com

**Advertising**Brian Osborn, bosborn@linuxnewmedia.com  
phone +49 8093 7679420**Marketing Communications**Gwen Clark, gclark@linuxnewmedia.com  
Linux New Media USA, LLC  
4840 Bob Billings Parkway, Ste 104  
Lawrence, KS 66049 USA**Publisher**

Brian Osborn

**Customer Service / Subscription**For USA and Canada:  
Email: cs@linuxnewmedia.com  
Phone: 1-866-247-2802  
(Toll Free from the US and Canada)For all other countries:  
Email: subs@linux-magazine.com

www.linux-magazine.com

While every care has been taken in the content of the magazine, the publishers cannot be held responsible for the accuracy of the information contained within it or any consequences arising from the use of it. The use of the disc provided with the magazine or any material provided on it is at your own risk.

Copyright and Trademarks © 2023 Linux New Media USA, LLC.

No material may be reproduced in any form whatsoever in whole or in part without the written permission of the publishers. It is assumed that all correspondence sent, for example, letters, email, faxes, photographs, articles, drawings, are supplied for publication or license to third parties on a non-exclusive worldwide basis by Linux New Media USA, LLC, unless otherwise stated in writing.

Linux is a trademark of Linus Torvalds.

All brand or product names are trademarks of their respective owners. Contact us if we haven't credited your copyright; we will always correct any oversight.

Printed in Nuremberg, Germany by Zeitfracht GmbH. Distributed by Seymour Distribution Ltd, United Kingdom

Represented in Europe and other territories by: Sparkhaus Media GmbH, Bialasstr. 1a, 85625 Glonn, Germany.

Published monthly as Linux Magazine (Print ISSN: 1471-5678, Online ISSN: 2833-3950) by Linux New Media USA, LLC, 4840 Bob Billings Parkway, Ste 104, Lawrence, KS 66049, USA. Periodicals Postage paid at Lawrence, KS and additional mailing offices. Ride-Along Enclosed. POSTMASTER: Please send address changes to Linux Magazine, 4840 Bob Billings Parkway, Ste 104, Lawrence, KS 66049, USA.

# WRITE FOR US

*Linux Magazine* is looking for authors to write articles on Linux and the tools of the Linux environment. We like articles on useful solutions that solve practical problems. The topic could be a desktop tool, a command-line utility, a network monitoring application, a homegrown script, or anything else with the potential to save a Linux user trouble and time. Our goal is to tell our readers stories they haven't already heard, so we're especially interested in original fixes and hacks, new tools, and useful applications that our readers might not know about. We also love articles on advanced uses for tools our readers *do* know about – stories that take a traditional application and put it to work in a novel or creative way.

Topics close to our hearts include:

- Security
- Advanced Linux tuning and configuration
- Internet of Things
- Networking
- Scripting
- Artificial intelligence
- Open protocols and open standards

If you have a worthy topic that isn't on this list, try us out – we might be interested!

Please don't send us articles about products made by a company you work for, unless it is an open source tool that is freely available to everyone. Don't send us webzine-style "Top 10 Tips" articles or other superficial treatments that leave all the work to the reader. We like complete solutions, with examples and lots of details. Go deep, not wide.

We have a couple themes coming up that we could use your help with. Please send us your proposals for thoughtful and practical articles on:

- **Cryptocurrencies**
- **Systemd hacks**

Describe your idea in 1-2 paragraphs and send it to: [edit@linux-magazine.com](mailto:edit@linux-magazine.com).

Please indicate in the subject line that your message is an article proposal.

**Authors**

Erik Bärwaldt	30	Vincent Mealing	73
Bruce Byfield	6, 28, 42	Graham Morrison	84
Joe Casad	3, 12, 24	Dr. Günter Pomaska	58
Mark Crutch	73	Mike Schilli	52
Nate Drake	16, 80	Ferdinand Thommes	75
Marco Fioretti	20, 64, 90	Koen Vervloesem	46
Jon "maddog" Hall	74	Jack Wallen	8
Rubén Llorente	36		

**Approximate**

UK	Jul 07
Europe	Jul 21
USA / Canada	Aug 11
Australia	Sep 22

**On Sale Date**

Issue 273 / August 2023

# Podcasting

Some podcasters are devoted to a cause; others are building a resume; some are in it for the fun. Whatever your reasons, podcasting is a way of reaching out to the world – with almost no barriers to entry. Next month, we show you how to get started.



## Preview Newsletter

The Linux Magazine Preview is a monthly email newsletter that gives you a sneak peek at the next issue, including links to articles posted online.

Sign up at: <https://bit.ly/Linux-Update>

Image © aleksanderdn, 123RF.com



# AKADEMY 2023

Akademy is the annual world summit for KDE Community members, developers, translators, designers, and friends. Come join us!

Akademy 2023 will be a hybrid event, in **Thessaloniki, Greece & online**

**July 15 - 21**



# HETZNER

## SAVE NOW WITH THE **HETZNER SERVER AUCTION** THE CLOCK IS TICKING!



### DESCENDING PRICES ON THE HETZNER SERVER AUCTION!

Get yourself a dedicated server with powerful hardware at an unbeatable monthly price.

We show you to the second when the server price will decrease so you can spring into action at the right moment.

Don't wait too long; otherwise, someone else may steal the deal you want!

### e. g. DEDICATED SERVER

- ✓ AMD Ryzen 7 3700X
- ✓ 64 GB DDRx RAM
- ✓ 2 x 1 TB SSD
- ✓ IPv4 inclusive
- ✓ Traffic unlimited
- ✓ No minimum contract
- ✓ Setup Fee \$0

**AMD**  
**RYZEN**

monthly \$ **37**

All prices exclude VAT and are subject to the terms and conditions of Hetzner Online GmbH. Prices are subject to change. All rights reserved by the respective manufacturers.

[www.hetzner.com/sb](http://www.hetzner.com/sb)