



Maker Tricks: Build an electronic scoreboard



LINUX

MAGAZINE

ISSUE 273 – AUGUST 2023

Podcasting

Support your passion
and reach out to
your online
audience



NetBox: Keep network knowledge at your fingertips

Fair Play: This cool Go program divides soccer players into teams of equal skill

Fyne GUI Toolkit: Code once for multiple platforms

RawTherapee: Transform your digital images with this raw photo editor

fdupes: Clean up duplicate files

10 FOSS Gems: This is your brain on Linux



LINUX NEW MEDIA
The Pulse of Open Source



Stuck With Old Code?

We can help.

Websites, software & proprietary conversions.

Fast. Thorough. Durable. \$150/hr

Ruby on Rails

PHP

Javascript

Relational
Databases

NoSQL

Fulltext
Search

DURABLE PROGRAMMING.COM

1.617.356.8239

BLACKOUTS AND BLACK INK

Dear Reader,

The Reddit blackout is big in the news as I write this column. As you read it, you will probably know more than I know now, but for now, the disruption is starting to look significant, even though the company says it will all blow over in time.

A recap: Reddit, which is actually a company, although it has many features that make it seem more like a collection of community groups, announced that it would start charging for applications that access its API [1]. This news is significant because there are several companies that derive their revenue from apps that provide an interface with Reddit, and some of these vendors have declared that paying to access the Reddit API will drive them out of business. Apollo, makers of a Reddit client for Apple products, has already announced that the change will cost them \$20 million per year.

Reddit has been chugging along for 17 years. Why rock the boat now? It is no secret that the company is gearing up to go public with an IPO later this year [2]. The whole scenario fits with the usual story for how these tech ventures go. Venture capitalists invest in a private company to build it up (with an emphasis on growth, rather than profit), then at some point, they go public with an IPO, and everyone makes a lot of money. But the IPO only works if Wall Street likes what it is buying, and Wall Street likes profits. So if Reddit wants to make a splash, they need to transition from a company that emphasizes growth (i.e., maximum access to the API) to a company that emphasizes profit (i.e., no access unless it brings in revenue).

Reddit is reportedly the 10th busiest website in the world, and the sixth busiest in the US, with 57 million daily users, so a boycott is big news. The company maintains that the

boycott has only shut down a small percentage of its forums. Out of a total of 100,000 active subreddits, 6,000 went dark initially, with others following to a maximum of 9,000, and some coming back later in the week. As of this writing, around 3,500 are still dark. But the actual numbers might not be as important as the attention it brings to Reddit at a critical time.

It is true that some of this attention could be positive attention from investors because Reddit executives are being “tough” and demanding profits. But it is also worth remembering that Reddit is in a peculiar situation that most companies will never face. Their whole business model depends on them looking and behaving like a community. Most of the moderators on Reddit volunteer their time. According to a recent study by researchers at Northwestern University [3], volunteers save Reddit \$3.4 million per year in labor costs. Of course, Reddit is a big company (\$3.4 million was equivalent to 2.8% of revenue at the time of the 2019 study). But the line between red and black ink is quite tenuous in the Internet economy, and \$3.4 million can make a difference. Most volunteers are happy to donate their time to a worthy cause, but a corporate titan dialing up its profits to impress Wall Street might not be a cause that many of these volunteers consider worthy.

I have some sympathy for Reddit. Are they supposed to go on forever giving free access to companies that don't contribute anything to pay for the infrastructure? And yet, if they operate on free labor and talk the talk of community spirit, should they not be held to a different standard from other profit-screaming ventures? The company says it is negotiating with several of the client companies in hopes that they will continue, although two of the largest, Apollo and Reddit is Fun (RIF), have already said they will close their doors.

Reddit CEO Steve Huffman was recently asked if he thought the real answer is that the profit model for building a massive company on Internet advertising just doesn't work. His answer was, “Are you saying Facebook doesn't work?” [4].

That's the whole point, though. Companies like Google and Facebook work very well, but not just by selling ads. They also spy on people, strong-arm their customers, and play corporate hardball when necessary to protect their interests. So yes, Reddit can be just like Google and Facebook, but when that happens, they might not look like the Reddit we all remember.



Joe Casad,
Editor in Chief

Info

- [1] “Redditors Go to War with the Company as It Enforces Eye-Watering Prices for Reddit API,” *Forbes*, June 13, 2023: <https://www.forbes.com/sites/qai/2023/06/13/redditors-go-to-war-with-the-company-as-it-enforces-eye-watering-prices-vfor-reddit-api/?sh=84f371db3dba>
- [2] “Reddit Aims for IPO in Second Half of 2023,” Reuters, February 14, 2023: <https://www.reuters.com/technology/reddit-aims-ipo-second-half-2023-information-2023-02-14/>
- [3] “Unpaid Social Media Moderators Perform Labor Worth at Least \$3.4 million a Year on Reddit Alone,” *Northwestern Now*, May 31, 2022: <https://news.northwestern.edu/stories/2022/05/unpaid-social-media-moderators/>
- [4] “Reddit CEO Steve Huffman: ‘It's time we grow up and behave like an adult company,’” *Morning Edition*, NPR, June 15, 2023: <https://www.npr.org/2023/06/15/1182457366/reddit-ceo-steve-huffman-its-time-we-grow-up-and-behave-like-an-adult-company>

ON THE COVER

42 Coding with Fyne

Give your custom Go app a graphical interface.

48 NetBox

Put all those useful bits of network information in one place.

58 Programming Snapshot - Teams

Mike spins up an inventive solution for choosing sports teams based on player statistics.

64 DIY Scoreboard

We'll show you how to create your own scoreboard with a Raspberry Pi and a little Python.

78 fdupes

If you don't love housekeeping, you'll love this handy tool for finding duplicate files and folders.

88 RawTherapee Workshop

Access the raw image data to go deeper with your photo editing.

NEWS

8 News

- Fedora-Based Ultramarine Linux
- Debian 12 Has Finally Been Released
- Armbian 23.05
- Linux Mint Finally Receiving Support for Gestures
- An All-Snap Version of Ubuntu Is in the Works
- Mageia 9 Beta 2 Ready for Testing
- KDE Plasma 6 Looks to Bring Basic HDR Support
- Bodhi Linux 7.0 Beta Ready for Testing
- Changes Coming to Ubuntu PPA Usage

12 Kernel News

- Rust in Linux
- Compiler and Kernel Frenemies

COVER STORIES

16 Podcasts

If you use Linux, you already have most of the tools you need to get in the podcast game. Just plan carefully and take it a step at a time.

20 Audacity

This free, open source, easy-to-use, multi-track audio recording and editing tool is perfect for podcasts.

24 Sound Studio Workshop

Once you get your podcast operation up and running, you might decide you want a real mixer and some higher end software. We'll introduce you to Ardour and get you started with some basic audio hardware.

REVIEWS

32 TUXEDO InfinityBook Pro 16 Gen7 MK1

The next-generation laptop from TUXEDO is faster and lighter than previous business models.

36 Distro Walk – Trisquel

Rúben Rodríguez discusses Trisquel, a free Linux distro that has been in continuous development for the past 16 years.

IN-DEPTH

40 Command Line – most

The most terminal pager offers a feature-rich, better organized alternative to less.

42 GUI Apps with Fyne

The Fyne toolkit offers a simple way to build native apps that work across multiple platforms. We show you how to build a to-do list app to demonstrate Fyne's power.

48 NetBox

NetBox is a single source of information on your network where you can store all those important details that used to get lost.

58 Programming Snapshot – Go Algorithms

Instead of the coach determining the team lineup, an algorithm selects the players based on their strengths for Mike Schilli's amateur soccer team.

95 Back Issues

97 Call for Papers

96 Events

98 Coming Next Month



Podcasting

On the Internet, you don't have to wait for permission to speak to the world. Podcasting lets you connect with your audience no matter where they are. Whether you're in it to build community, raise awareness about your skills, or just have some fun, the tools of the Linux environment make it easy to take your first steps

MakerSpace

64 DIY Scoreboard

We look at a broadcast video system network that uses Python code to control a video router and check out another program that creates a scoreboard.

LINUXVOICE

75 Welcome

This month in Linux Voice.

77 Doghouse – AI

Earlier technologies have persisted despite government regulations – so will AI.

78 fdupes

The command-line tool fdupes helps you find duplicate folders and directories.

82 FOSSPicks

This month Graham looks at wallabag, Read It Later, killport, F3D, Tenacity, Cataclysm: Dark Days Ahead, botany, and more!

88 Tutorial – RawTherapee Workshop

The current RawTherapee version finally adds selective image editing, among other long-desired features, to help it compete with king of the hill, darktable.



TWO TERRIFIC DISTROS

DOUBLE-SIDED DVD!

SEE PAGE 6 FOR DETAILS

-  @linux_pro
-  @linuxpromagazine
-  Linux Magazine
-  @linuxmagazine

Linux Mint 21.1 and openSUSE Leap 15.5

Two Terrific Distros on a Double-Sided DVD!



Linux Mint 21.1
64-bit

Linux Mint is one of the most popular Debian derivatives, noted for its responsiveness to users. Linux Mint 21.1, codenamed Vera, is based on Ubuntu 22.04 and will be supported until 2027.

For this release, the Cinnamon desktop defaults to more vibrant colors and a reduced set of icons that includes only the installation icon, mounted devices, and any items moved to ~/Desktop. The Home folder, Computer, Trash, and Network icons were removed because they are readily available from the file manager. Those who prefer the old look can use the Mint-Y-Legacy theme. Complementing this updated look are new mouse and system sound options. In addition, the Drive Manager has been redesigned and no longer requires the root password. The Update Manager now includes Flatpak packages and a menu item to automate ISO image verification. The Nemo file manager and the desktop have also undergone numerous cosmetic changes for both aesthetics and efficiency. Many of these changes are minor, but together they increase Linux Mint's user-friendliness for users of all levels.



openSUSE Leap 15.5
64-bit

OpenSUSE's development has always been closely tied to SUSE Linux Enterprise (SLE). However, Leap 15.5 is the first release to use source packages from SLE. According to the openSUSE download page, this move is intended to increase the stability of this already mature distribution. The release will receive maintenance and security updates until the end of 2024. Those who require longer support can migrate to SLE commercial support.

Leap 15.5 continues openSUSE's practice of expanding its inclusion of recent technologies, such as containers, immutable systems, virtualization, and embedded development. On the desktop, the introduction of KDE Plasma 5.27 means a new welcome wizard, enhanced features for KRunner, the refinement of the color picker and the ark compression utility, the Kate editor, and the Kdenlive video editor. In addition, the new Flatpak version features enhanced security, while advanced users can explore a faster version of the Vim editor that introduces a new scripting language, as well as netavark, a tool for networking containers, and a new repository for Cisco's OpenH264 codec, whose packages are released under a BSD license. For all levels of users, Leap 15.5 is an incremental upgrade built on the stable base of a long history of development.

Defective discs will be replaced. Please send an email to 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
shop.linuxnewmedia.com



Customers in All Other Countries
sparkhaus-shop.com

Thank you for your continued support!



NEWS

Updates on technologies, trends, and tools

THIS MONTH'S NEWS

- 08 • Fedora-Based Ultramarine Linux Designed for Ease of Use
- Debian 12 Has Finally Been Released
- 09 • Armbian 23.05 Now Available
- Linux Mint Finally Receiving Support for Gestures
- An All-Snap Version of Ubuntu Is in the Works
- More Online
- 10 • Mageia 9 Beta 2 Ready for Testing
- KDE Plasma 6 Looks to Bring Basic HDR Support
- 11 • Bodhi Linux 7.0 Beta Ready for Testing
- Changes Coming to Ubuntu PPA Usage

Fedora-Based Ultramarine Linux Designed for Ease of Use

Ultramarine Linux 38 is based on Fedora 38 and is offered with either the Budgie, KDE Plasma, Gnome, or Pantheon desktop environments. This latest release adds something special to improve performance.

That boost comes in the form of the System76 CPU schedule that prioritizes processes. The one caveat to this is that only the Gnome version automatically detects an application in use. However, all four editions do include various tweaks to the system to ensure users get as smooth an experience as possible. This includes improved startup and shutdown times.

As well, Ultramarine includes the latest versions of the included software. You'll find Flatpak supported by default, and there are even non-free software titles that can be installed.

Preinstalled software includes Firefox, LibreOffice, Clapper audio/video player, the basic Gnome apps (Software, Weather, Files, Maps), and plenty of utilities to get you by. Thanks to Flatpak support, you can easily install third-party, proprietary software such as Spotify and Slack.

The big appeal to Ultramarine Linux is its performance. All of the tweaks the developers have done make a big difference, and Ultramarine Linux outperforms a lot of similar Linux desktop distributions. The second you start using it, you'll notice the speed.

Download an ISO (<https://ultramarine-linux.org/download/>) now. You also can migrate Fedora to Ultramarine Linux with the help of a script you can download with the command `wget https://ultramarine-linux.org/migrate.sh`. Give the command executable permissions with `chmod u+x migrate.sh` and then run the script with `sudo ./migrate.sh`.

Debian 12 Has Finally Been Released

Debian marches to a beat that so many other Linux distributions follow. Over the decades since it was first released in 1996, Debian has developed a reputation for being one of the most stable operating systems on the market. Part of the reason for that is the development team takes their time between releases.

Unlike Ubuntu which releases every six months, a new version of Debian happens yearly (sort of). So you can bet plenty of effort has gone into each release to make it rock solid.

With the release of 12.0 (aka "bookworm"), you'll find updates to several desktops, such as Gnome 43, KDE Plasma 5.27, LXDE 11, LXQt 1.2.0, MATE 1.26, and Xfce 4.18. No, these are not the latest versions of each but that's how Debian rolls (keeping everything as stable as possible).

Other software updates include Apache 2.4.57, BIND DNS Server 9.18, Cryptsetup 2.6, Gimp 2.10.34, GnuPG 2.2.40, Inkscape 1.2.2, GNU C Library 2.36, LibreOffice 7.4, Linux kernel 6.1, MariaDB 10.11, NGINX 1.22, OpenJDK 17, OpenLDAP 2.5.13, OpenSSH 9.2p1, PHP 8.2, Python 3.11.2, Samba 4.17, and more. Overall, there are 11,089 new packages found in Debian 12, for a total of 64,419.

There also is improved support and management of non-free firmware as well as support for UEFI on ARM64 architecture.

Debian is available for nine different architectures including 32/64-bit PC, ARM64, ARMv7, 64-bit little-endian MIPS, 64-bit little-endian PowerPC, and IBM Z.

You can read the full release notes for Debian 12 here and download an ISO from the official Debian download page (<https://www.debian.org/download>).

Armbian 23.05 Now Available

Armbian is almost at its 10-year anniversary and has announced the release of version 23.05. This release is based on Debian and created specifically for ARM and RISC-V architecture. Version 23.05, nicknamed “Suni,” is based on a completely refactored build framework that has been in the works for the past three years.

According to the project website, “...many ARM-focused Linux distributions are essentially Armbian under a different name! We are committed to streamlining complexity and offering an exceptional solution for the community.”

The latest version of Armbian offers several improvements which include Armbian bookworm-based images for the latest updates from Debian, i3 support, improvements to the `armbian-installer`, an optimized package base, consistent application packages, improved Ubuntu-based assemblies, fast and safe updates, the inclusion of the `armbian-gaming` script, and plenty of bug fixes and security patches.

Users without ARM or RISC-V hardware can download a Ubuntu-based image for AMD/Intel machines. This version has much of the Ubuntu-ness removed (such as the removal of Ubuntu Pro and Snap).

Read the detailed changelog to see what is new to Armbian 23.05 and download an ISO image for installation on the single-board computer of your choice. On the download page, you’ll also find links for AMD64 images.

Linux Mint Finally Receiving Support for Gestures

Getting multitouch gestures for touchscreen and touchpads on Linux has been problematic over the years. Although some distributions have offered it, the support has been fairly weak (at best).

Hopefully, that all changes with the release of Linux Mint 21.2 (and Cinnamon 5.8). When using the default Cinnamon desktop, users will finally get to enjoy gestures that will support media player controls, tilings, and other helpful features.

According to Clement “Clem” Lefebvre (<https://blog.linuxmint.com/?p=4518>), these gestures will be supported for touchpads, touchscreens, and tablets.

Linux Mint 21.2 (based on Ubuntu 22.04) will also enjoy a few more updates and tweaks, such as Xfce v4.18, global dark mode (thanks to an implementation of `xdg-desktop-portal`), and the Software Manager was given a UI refresh as well as better scoring/sorting algorithms for applications and a special list for tuned packages, an improved login (that includes support for multiple keyboard layouts and better touchpad support), auto-detection and enabling of tap-to-click, support for Wayland sessions, support for HEIF and AVIF images, an improved Pix image viewing app, better support for Flatpak apps, and improved support for Gnome’s `libadwaita` library.

Linux Mint 21.2 (Victoria) is slated for release in June 2023. For those wanting to test Linux Mint 21.2, you’ll have to use the Daily Builds (https://linuxmint-developer-guide.readthedocs.io/en/latest/daily_builds_ppa.html) PPA and upgrade from the stable to the unstable version, or you can wait until they release the beta version, which will probably happen a couple weeks before the final release. Make sure to check the official ISO Images page (<https://community.linuxmint.com/iso>) daily.

An All-Snap Version of Ubuntu Is in the Works

It should come as no surprise, given how Canonical is pushing Snap packages for its flagship OS, Ubuntu, that they’d take things to the next level by offering a version of Ubuntu that is comprised completely of Snap packages.

MORE ONLINE

Linux Magazine

www.linux-magazine.com

ADMIN HPC

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

Saving Storage Space

• Jeff Layton

People often turn to data compression to save storage space, but reducing precision can be a useful technique.

ADMIN Online

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

Package management tools for Windows

• Evgenij Smirnov

Chocolatey and WinGet offer full-fledged package management on Windows, but which is best for your environment?

CrowdSec crowd security service

• Thomas Joos

Threats can be detected and averted at an early stage with crowd security, in which organizations form a community to take concentrated action against cyberattacks by sharing attack data. We explain how this strategy works with the CrowdSec cloud service.

Verifying your configuration

• Ankur Kumar

Automated acceptance testing is a powerful tool for catching problems related to misconfiguration. We’ll show you how to implement your own acceptance testing environment with a free tool called goss.

Before anyone gets up in arms, the all-Snap version of Ubuntu will not be the default. When you go to download Ubuntu next year (when the all-Snap version will be available), it will default to the traditional Debian version of the OS.

So, what's the appeal of the all-Snap version of Ubuntu? It'll be immutable, which means it was designed so the operating system is mounted read-only, so it can't be changed. Once the OS has been installed, system files and directories cannot be changed. This isn't new to Ubuntu, as Ubuntu Core (another immutable version) has been around since 2015. The idea behind immutable operating systems is to make them more secure.

This announcement was subtly dropped in the comments of the CUPS Snap announcement posted on omgubuntu.com (<https://www.omgubuntu.co.uk/2023/05/cups-snap-ubuntu-23-10#comment-6196766355>). Canonical's Oliver Grawert said, "There will be a desktop release of it with the next LTS (optional though, the classic desktop install will indeed not go away)."

The next LTS version of Ubuntu is 24.04, which will be released in April 2024, so you can bet there will be a slow trickle of information about this new all-Snap version of the Ubuntu Desktop distribution.

Mageia 9 Beta 2 Ready for Testing

Mageia was forked from Mandriva Linux back in 2011 and, since then, has become something all of its own. This latest release comes three months after the first beta and the team has put in a good amount of time fixing stubborn issues, adding security fixes, and application updates.

Beta 2 features Linux kernel 6.3.3, GLib 2.36, RPM 4.18.0, Chromium 110, Firefox ESR 102.11, LibreOffice 7.5.2, KDE Plasma 5.27.4, Gnome 44, Xfce 4.18.1, LXQt 1.3.0, and Mesa 23.1.0.

Users will find three different repositories included with Mageia: Core (includes free and open-source software), Non-free (contains software that is free to use and distribute but includes proprietary software), and Tainted (contains packages released under a free license, but which infringe on patents and copyright laws in some countries). Mageia does configure 32-bit repositories, but they are disabled on 64-bit systems.

When you download a version of Mageia, make sure to select your desktop environment of choice: KDE Plasma (https://www.mageia.org/en/downloads/get/?q=Mageia-9-beta2-Live-Plasma-x86_64.iso), Gnome (https://www.mageia.org/en/downloads/get/?q=Mageia-9-beta2-Live-GNOME-x86_64.iso), Xfce (32-bit (<https://www.mageia.org/en/downloads/get/?q=Mageia-9-beta2-Live-Xfce-i586.iso>), or 64-bit (https://www.mageia.org/en/downloads/get/?q=Mageia-9-beta2-Live-Xfce-x86_64.iso)).

To find out more about the new release, check out the official Mageia blog (<https://blog.mageia.org/en/2023/05/24/the-release-of-beta2-brings-mageia-9-stable-closer-to-reality/>). At this time, there is no official release date for Mageia 9, but keep checking the official blog (<https://blog.mageia.org/en/>) for more information.



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

**Subscribe FREE
to Linux Update
bit.ly/Linux-Update**

KDE Plasma 6 Looks to Bring Basic HDR Support

When KDE Plasma 6.0 finally lands, users will find that High Dynamic Range (HDR) support has finally arrived. Because 4K screens are becoming the norm, HDR support has become a necessity.

With HDR, video content can be pushed well beyond the limits of the now outdated standards media outlets have held onto for decades. Support for HDR in KDE Plasma 6.0 came by way of an HDR hackfest that was organized by Red Hat in May 2023 as well as the Plasma Sprint.

KDE Plasma 6.0 will only offer HDR support for Wayland sessions, which, according to KDE developer Nate Graham (<https://pointieststick.com/2023/05/20/this-week-in-kde-preliminary-hdr-support/>), "lays the groundwork for color management on Wayland."

You can read a very in-depth post, written by KDE developer Xaver Hugl (<https://zamundaaa.github.io/wayland/2023/05/18/hdr-and-color-management-in-kwin.html>), about HDR and color management in KWin. In his post, Hugl says, “We didn’t do a lot of hacking at the hackfest, but I did manage to drive an HDR screen with a wide color gamut and with HDR mode enabled while having KWin do the required color conversions to make SDR content look correct.”

One thing to keep in mind is that, with KDE Plasma 6, Wayland will be the default display manager.

Also, according to Hugl, HDR support will be “quite basic at first.” Because of that, users won’t be able to jump in and immediately watch videos or play games in HDR. The development team has a long road ahead of them before HDR support in Linux matures.

Bodhi Linux 7.0 Beta Ready for Testing

Bodhi Linux has been around for some time now, and the latest release promises new versions of software, better hardware support, heightened security, improved performance, and much more to entice new users and please those who have been on board.

Front and center of the new version is Moksha 0.4.0-8 desktop, which includes a number of improvements, including completely refactored modules, a new keybindings viewer, better window snapping, Do Not Disturb feature for menu items, an Internal Notification API which was backported from e25, and much more. As well, there are new themes for both Plymouth and the Greeter application.

For the base, you’ll find Ubuntu 22.04.2 LTS, a choice of three different kernels (Standard 5.15.0-71-generic, HWE 5.19.0-41, and 6.2.6), EFL 1.26.99-2, the mozillateam PPA, kebek333’s PPA, nvidia-legacy PPA, and Ubuntu Backports is enabled.

A lot of work has also gone into improving and fixing the many modules included with Bodhi Linux, and there’ve been plenty of software updates, including Chromium (non-Snap version) 113.0.5672.63, Terminology 1.13.1-3, the Web Browser Manager (which allows you to select any number of browsers to install), and Thunar 4.16.

You can read all about version 7.0 here (<https://www.bodhilinux.com/release/7-0-0/>) and download the beta version here (<https://sourceforge.net/projects/bodhilinux/files/7.0.0-beta/>).

Changes Coming to Ubuntu PPA Usage

With the upcoming Ubuntu 23.10 (Mantic Minotaur), there will be a considerable change to how PPAs are handled. As you may know, in the current iteration of the software-properties software, when you add a PPA from the command line, a .list file is created in /etc/apt/sources.list.d/, and the associated GPG key is added to /etc/apt/trusted.gpg.d/.

When 23.10 is released, those PPAs will use the Deb822 format for .source files and their corresponding GPG keys will be added directly to the file in a Signed-By field. This means users won’t have to manage a collection of .list files.

According to the developers, this change offers one very important benefit: When a PPA is removed from a system, the GPG key will be automatically removed as well. Keys will now be unique to a PPA and cannot be used for other repositories, and other keys cannot be used to sign a PPA. These benefits will go a long way to enhance the security of PPAs. Another benefit of the new system is that users won’t have to worry about deleting .list files that can accumulate on a system.

Of course, there’s always a downside, the biggest of which is that PPAs will have root access to a system. Because of this, a program maintainer could add malicious code to a repository, and the next time you upgrade, that malicious code would be installed and have unfettered access to your machine.

Read the announcement from the ubuntu-devel mailing list (<https://lists.ubuntu.com/archives/ubuntu-devel/2023-May/042573.html>).

Zack's Kernel News



Chronicler Zack Brown reports on the latest news, views, dilemmas, and developments within the Linux kernel community.

By Zack Brown

Author

The Linux kernel mailing list comprises the core of Linux development activities. Traffic volumes are immense, often reaching 10,000 messages in a week, and keeping up to date with the entire scope of development is a virtually impossible task for one person. One of the few brave souls to take on this task is **Zack Brown**.

Rust in Linux

A fascinating aspect of Linux kernel development has been the attempts to switch away from C and assembly language to something else like C++. Linus Torvalds is notorious for caring a lot about the beauty of the kernel code as well as important features such as CPU and RAM usage and security. For him to accept any other language, it would have to be a pretty amazing language. For comparison, the developers tried adding C++ support to the kernel in the late 1990s, but it didn't pass muster.

The Rust programming language is the first serious attempt by kernel developers to add support to the kernel since then. Linus allowed a rudimentary first attempt to go into kernel version 6.1 in late 2022. Since then, many more Rust patches have been submitted. Unlike the case with C++, Rust doesn't seem to be leaving any time soon. Instead, it seems as though Linus is fairly committed to giving it a real try.

It's a surprising decision because Rust is so new. Rust was invented in 2006 while C++ was invented in 1979. Even today, Rust has hardly taken over the world. And yet it has been given this amazing pride of place in the Linux kernel. I'm not denigrating! It's fascinating to consider Rust's characteristics that would place it even above C++ as being welcome into the kernel development process. However, it's not at all surprising that Linus would evaluate a new language based on its merits rather than its popularity.

Incorporating Rust into the kernel takes a twisting and turning path. The portions of the kernel written in C and assembly need to connect with the portions written in Rust in such a way that there's no loss of speed or security, and in such a way that both Rust and C developers can write good and readable code.

For example, Wedson Almeida Filho recently posted a patch to introduce a mutex structure for Rust that, as he put

it, "allows Rust code to use the kernel mutex idiomatically."

Wedson's patch modifies both C and Rust files in the kernel source to add this support. If you're curious what the Rust code would look like for a mutex initialization routine, here is a snippet of Wedson's patch:

```
macro_rules! new_mutex {
    ($inner:expr $(, $name:literal)? ↵
    $(,)? => {
        $crate::sync::Mutex::new(
            $inner, ↵
            $crate::optional_name!($($name)?), ↵
            $crate::static_lock_class!()
        );
    }
}
```

Rust is a strange and fascinating language that is seemingly so different from C, but with features that make it a welcome addition to the kernel project.

Peter Zijlstra replied, "I despise all these stupid helpers ... but I suppose it's the best they could come up with to interface the languages :/ The only hope is that the thing can do cross-language LTO [Link Time Optimization] or something to re-inline stuff."

Wedson replied:

"One thing we could [do to] improve the situation is to convert some of the existing macros into inline functions on the header files.

"We can't do it for all cases (e.g., cases like mutex_init that declare a new static variable when lockdep is enabled) but mutex_lock is just a function when lockdep is disabled, and just calls mutex_lock_nested() when it is enabled.

"How do you feel about this?"

In reply, Peter asked, "Can rust actually parse C headers and inline C functions ? I thought the whole problem was that it can only call C ABI symbols (which inline functions are not)."

Wedson replied:

"Rust can't. We use a tool called bindgen to read C header files and generate equivalent Rust (extern 'C') declarations for functions. The tool has been

enhanced recently (<https://github.com/rust-lang/rust-bindgen/pull/2335>) to handle static inline functions by automatically generating helpers like the one above (in addition to the Rust decls).

“So the situation is improved in that we don’t need to manually write (and commit) the helpers. It may improve further in the future if we get better integration of the languages.”

Peter accepted this, saying that in that case, “feel free to convert macros to inline functions where the difference is moot. There is indeed no real reason for `mutex_lock()` to not be an inline function in that case.” In response to Wedson saying that Rust could not actually parse C headers and inline functions, Peter added, “that’s sad, I was hoping it would write the equivalent inline function in rust.”

David Laight chimed in at this point, regarding converting macros to inline functions. He remarked, “`mutex_lock()` is probably ok. But there are cases where `gcc` generates much better code for `#defines` than for inline functions. Almost certainly because the front end gets to optimise `#defines`, but inlines are done much later on.”

Marco Elver suggested using “`static __always_inline`” in C files for macros that would be better converted to inline functions. He asked, “Can bindgen deal with ‘`static __always_inline`’ functions?” And Miguel Ojeda replied, “If you mean the new feature where ‘`bindgen`’ generates wrappers automatically, it seems to handle them given ‘`__always_inline`’ = > ‘`inline`’ which is what I imagine it looks for [...], though I haven’t tried to use the feature within the kernel yet.”

Miguel, speaking for the Rust community, also said, “We initially minimized changes on the C side on purpose, but if maintainers are OK with that (modulo exceptions), it would be great.”

The conversation ended there, though this thread is absolutely part of a much larger and ongoing discussion between the C and Rust communities of kernel developers. There are many Rust patches flying around these days that are being taken very seriously by both groups.

Rust is a highly impressive language. Its designers seem to take the position that they will build protection against memory leaks and segmentation

violations into the language itself, at the cost of including a few relatively convoluted language features. They also seem to make it relatively easy to avoid these convolutions at the cost of writing slower code. So for someone who only wants the higher level language features, such as easy-to-use complex data structures in a fast compiled language, you can have that without too much language complexity if you don’t mind taking a small speed hit. For people who want code that is fully speed-competitive with the C language, you can have that if you take advantage of Rust’s more complex memory protection features.

I don’t know why Linus chose to accept Rust into the kernel, but I speculate that it was precisely because of those memory protection features. There seem to be very few powerful low-level languages that incorporate such features, and it seems like a tough needle to thread. I believe Linus probably feels that the compromise of Rust’s additional complexity as a language is a very fair trade for the protections it provides, which are also very difficult to find elsewhere. I’m fascinated to watch the further progression of Rust development in the Linux kernel.

Compiler and Kernel Frenemies

Theodore Ts’o posted a patch to update the ext4 filesystem code, in the course of which he mentioned that another patch by Matthew Wilcox needed to be applied on top of his, making a particular function return an error pointer instead of simply the null value.

The details of Matthew’s patch (and Ted’s patch) are not necessary to know, and both patches were accepted into the kernel. However, Linus Torvalds commented:

“[I]t would be wonderful if we could mark the places that return an error pointer as returning ‘nonnull’, and catch things like this automatically at build time where people compare an error pointer to NULL.

“However, it sadly turns out that compilers have gotten this completely wrong.

“gcc apparently completely screwed things up, and ‘nonnull’ is not a warning aid, it’s a ‘you can remove tests against NULL silently’.

*“And clang does seem to have taken the same approach with ‘returns_nonnull’, which is really really sad, considering that apparently they got it right for ‘_Nonnull’ for function arguments (where it’s documented to cause a warning if you pass in a NULL argument, rather than cause the compiler to generate sh*t buggy code).”*

“Compiler people who think that ‘undefined behavior is a good way to implement optimizations’ are a menace, and should be shunned. They are paste-eaters of the worst kind.”

“Is there any chance that somebody could hit compiler people with a big cluebat, and say ‘undefined behavior is not a feature, it’s a bug’, and try to make them grow up?”

*“Adding some clang people to the participants, since they at least seem to have *almost* gotten it right.”*

Nick Desaulniers replied to Linus, saying, “Good. I can feel your anger. Strike me down with all of your hatred, and your journey to the dark side will be complete. Your hate has made you powerful. Let the hate flow through you!”

Nick added that Clang’s `_Nonnull` attribute did catch comparisons between null pointers and error pointers at build time, but that “it’s not toolchain portable, at the moment. Would require changes to clang to use the GNU C `__attribute__` syntax, too (which I’m not against adding support for).”

He also remarked, regarding Linus’s statement that Clang got `nonnull` right in one spot but not in another:

“I just had this conversation maybe within the past month with Bionic (Android’s libc) developers.”

“Yeah, the nonnull attributes != _Nonnull ‘attributes’. (Quotes because IIUC _Nonnull doesn’t use the __attribute__ GNU C extension syntax). My understanding (which may be wrong) is that nonnull is implemented for compatibility with GCC, while _Nonnull was likely implemented by Apple (my guess; did not check) (so compatibility with GNU C __attribute__ syntax probably wasn’t considered in code review).”

Nick added, “The Bionic developers are deploying `_Nonnull` throughout the codebase and intentionally not using `nonnull` which is dangerous (a teammate used the term ‘Developer Hostile Behavior’). `nonnull` has implications on

codegen, `_Nonnull` only affects diagnostics.”

Linus responded to Nick’s statement that Clang’s `_Nonnull` attribute did indeed catch comparisons between null pointers and error pointers. He said:

“No, that’s a warning about using it, not a warning about testing for NULL when it’s there.”

“I actually tested _Nonnull. It seems to work for arguments. But it does not work for return values.”

“Of course, maybe there’s some other magic needed, but it does seem to be sadly not working for us.”

In terms of toolchain portability, and specifically the need to modify Clang to use the GNU C `__attribute__` syntax, Linus said, “No need for using the `__attribute__` syntax at all, that would `_not_` be a show-stopper.” He added, by way of letting the Clang developers know what would be workable:

“While it’s true that it’s the common syntax, and we sometimes use it explicitly because of that, it’s by no means the only syntax, and we actually tend to try to have more legible wrappers around it.”

“So, for example, we prefer using ‘__weak’ instead of writing ‘__attribute__((__weak__))’.”

“And no, it very much doesn’t have to use __attribute__ at all. We already have things like

```
#define __diag(s) ⚠
#pragma(__diag_str(GCC diagnostic s))
```

“so we already use other syntaxes.”

“End result: if it actually worked, I’d happily do something like

```
#define __return_nonnull _Nonnull
```

“in <linux/compiler-clang.h>, with then <linux/compiler-gcc.h> then just having

```
#define __return_nonnull
```

“along with a big comment about how __attribute__((nonnull)) is horrible garbage that should never every be used.”

Nick replied, “I see your point; it would be nice to flag that the comparison against NULL seems suspicious.” He acknowledged the rest of Linus’s explanation as well.

The discussion began to meander here. At one point, Linus offered the following perspective:

“The advantage of compiler warnings really is that they get caught quicker and developers will react to them much better. They might cause the code to be properly re-organized or rewritten to be much nicer, for example.”

“The ‘trivial tree’ kind of fixups for random other issues that get noticed separately tend to be much more about ‘work around issue’. It might be the proper fix, of course, but it didn’t end up being taken into account when writing the code, so it often ends up being just a ‘papering over the warning’ kind of fix. Particularly since the person trying to fix the problem generally isn’t the main developer of that code.”

In fact, at this point, the conversation began to get very technical – Nick posted some raw code as a sample of what he thought might satisfy Linus. And they (plus Theodore Ts’o) started getting into the real nitty gritty of things before the thread finally petered out.

I’m fascinated by the strange relationship between the kernel project and the various compiler projects. It’s such a chicken-and-egg type problem when these projects have disagreements over compiler features. I personally feel that the Linux kernel is one of the most important software projects (or the actual most important software project) in the world, and that therefore the compiler exists almost as a service tool for Linux. It makes perfect sense to me that any compiler would want to adopt the features desired by the Linux kernel project.

However, there is an absolutely legitimate point on the other side: The compiler is truly the fundamental software, and all other software projects are the beneficiaries that need to have some faith that the compiler people will make good decisions for everybody.

The validity of both of these perspectives creates an amazing situation when kernel and compiler projects find themselves at odds. As an aside, this same situation arises when hardware projects and compiler projects find themselves at odds. In such situations, the manner in which the debate proceeds, and in which the conflicts and disagreements are ultimately resolved, seem to me like they contain the secrets of the universe. ■■■

ALL THINGS OPEN 2023

OCTOBER 15-17

Downtown Raleigh, NC USA

≈5000
attendees

150+
speakers

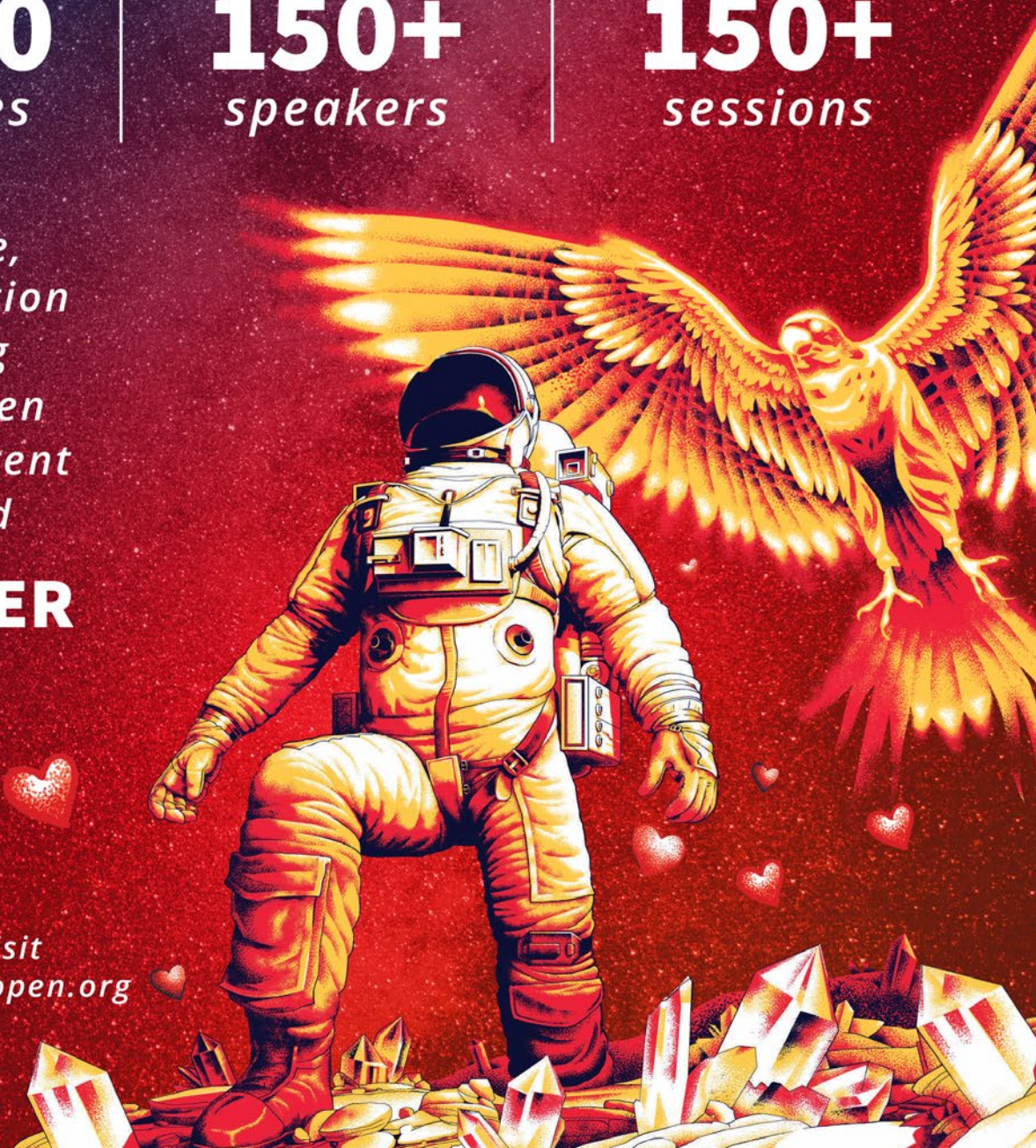
150+
sessions

*New decade,
same tradition
of featuring
the best open
source content
in the world*

**REGISTER
TODAY**



*Scan code or visit
2023.allthingsopen.org*



An inside look at creating a podcast

Showtime

If you use Linux, you already have most of the tools you need to get in the podcast game. Just plan carefully and take it a step at a time.

By Graham Morrison

Many of us have sat around discussing something with friends, maybe after a beer or a kombucha, only for someone to say, “This would make an amazing podcast!” Everyone says “Yes!” and nods enthusiastically, and the conversation shifts to talking about what to call the podcast, where it should be hosted, and whether recording something to a smartphone is good enough. Everyone goes to bed excited, only to wake up the next morning wondering why a podcast about mushroom varieties in *Zelda* sounded like such a good idea.

Or not. Sometimes a podcast idea does survive the next-day test, and there’s never been a better time to start your own podcast. As a Linux and open source user, your technical skills are already well suited to the task, and as a talking point, knowledge related to Linux and open source technologies is perennial, always interesting, and applicable across a huge range of other subjects, from politics to cooking. This gives you a significant advantage over most other people who wake up with a good podcast idea, and it’s an advantage that will help you overcome many of the initial barriers that stop people from making even their first podcast.

Have a Plan

The first thing to consider is a plan, and the first part of your plan should be deciding who should host the podcast. This can be even more important than the strength of the initial idea, because a compelling group of personalities can talk about almost anything. Many listeners will also want to feel like they’re listening to friends, or to people who understand them and what they’re interested in. It’s harder, but you can do this alone too by communicating your passion for a subject, making it infectious enough for listeners to not want to miss an episode. A wonderful example of this is the *History of English Podcast* [1], started by Kevin Stroud in 2012 and originally intended to last a mere 100 episodes. His idea was to discuss the etymology of English words and phrases, tracking their use from tribes wandering the Iranian plateau through to the Norman conquest of England, and on to the constant language flux of the modern world. Ten years later, Keith has just released episode 168, and he’s just made it to the late 1500s. He may

never be able to catch up, but his passion for the subject overwhelmed his modest ambitions, and that of his listeners too, which is a great sign of a good idea.

As with the *History of English Podcast*, it’s most likely that you’ve thought of a subject you know well and you want to share with other people – and you need to break this down into segments that will fit into individual episodes, or work as ongoing segments across multiple episodes. You might want to talk about news stories or things you and your co-hosts have discovered. It helps if these topics fall into predictable category segments, even if you intersperse them with more ad-hoc segments that are specific to their respective episodes. Magazines do the same thing for similar reasons. They too have predictable sections like news, reviews, and tutorials, but also more free-form and adaptable features and interviews that are part of a cover theme.

Another thing to consider is whether your podcast is going to be family friendly. A family friendly podcast will steer clear of swearing and certain subjects. This could limit your scope or spontaneity, or make editing harder if you need to remove segments or certain words, but it can broaden your audience because platforms like Apple Podcasts and YouTube demarcate content that isn’t broadly consumable. More practically, it can also help if people want to listen in the car with their families.

Duration and Cadence

Preparing and setting up for a podcast is like packing for a holiday – you pack almost the same for a weekend as you do for a week. In podcast terms, that means it’s the same effort to prepare for a 30-minute recording as for a 2-hour recording. But you must resist the temptation to make your podcasts too long, especially with such fierce competition for your listeners’ attention from other podcasts. It’s much easier to fit a 30-minute podcast into your listening schedule than it is a 2-hour podcast, and a 2-hour marathon session is much easier to skip over in your queue when looking for something to listen to while vacuuming. A longer recording and shorter duration will also give you more editing choices, leading to higher quality content. And as the saying goes, always leave your audience wanting more.



How often you record is closely related to episode duration. It's far better to produce two 30-minute podcasts than a single podcast lasting an hour. Generally, the more frequently you can produce a podcast the better. Once a week is ideal. This keeps your podcast in the minds of your listeners and helps them to remember you. Once every two weeks is another good option, especially to start with, but once a month might be too little. Either way, the more frequent your podcasts, the larger your monthly download figures will be, and this helps hugely if you ever want to sell advertising sponsorship.

It's essential you communicate clearly when a podcast episode is expected – and stick to it. Rain or shine, PulseAudio failure or hard drive crash. It's the one thing that differentiates amateurs from the professionals; professionals understand that listeners need an absolutely predictable routine. Predictable releases are a sign of strength and stability, because no one wants to invest time in something that may ultimately peter out, and the more your listeners need to do to work out which was your last episode and which is your next, the fewer listeners you'll have.

People Power

Three or four co-hosts is an ideal number because it allows listeners to get to know your personalities, and it allows them to feel they're among friends (Figure 1). Co-hosts can help share the burden of being engaged, doing the research, playing devil's advocate, and approaching a discussion from another perspective, or by simply compensating for another host who can't make it. It helps if you already know each other and are already comfortable and relatively unguarded in each other's company. Even for well-planned or scripted podcasts, it's the unpredictable parts that listeners enjoy, and spontaneous, divergent conversations are an important part of this. It's important that differing opinions on

topics are reflected by the hosts. A podcast should not become a vanity project for the strongest personality, or worse, the person who puts the podcast together.

The role of the main host is the most important and under-appreciated in the whole podcast enterprise. The main host is the person asking questions on behalf of the listener, sensitive to whether what's being talked about is likely to be understood. This is important whenever you have experts talking – an expert might assume a level of knowledge that you can't also assume of your listenership, and it's the main host's job to ask for clarification on points or challenge an assumption when needed. The main host should also try to make sure all hosts are equally included in the conversation, perhaps by asking a quieter host their opinions, or breaking into the monologue of a verbose participant to give someone else a chance to share an opinion.

Whoever drives the podcast needs to be subtle and transparent. If they're doing a brilliant job, no one will know. This is a thankless task that might even weaken your position within the hosting dynamic as you play more than one role from more than a single angle, but it's essential that someone adopts this persona, and it might as well be the podcast founder.



Figure 1: The rapport between co-hosts is sometimes more important than the podcast idea itself, although mixing up personalities and opinions is also vital.



Figure 2: Performing and recording a podcast live can be a lot of fun, but it doesn't necessarily result in the best quality episodes.

Publishing and Analytics

With a plan agreed, recordings made, and the important step of editing complete (Figure 2), you're ready to share your podcast with the world. We'd obviously recommend releasing your podcast under a Creative Commons license, and many Linux-based podcasts choose either the Creative Commons Attribution-NonCommercial (CC BY-NC) or the Attribution-ShareAlike (CC-BY-SA) licenses. The CC-BY-SA license is ideal if you're not concerned about people deriving monetary value from your work. Choosing either will also help with music and logo choices, because you're then free to use similarly licensed creative works within your own podcast.

Using music on the intro, the outro, and segment transitions adds a professional production quality to your podcast. Although music and design work can be commissioned cheaply through platforms such as Fiverr, choosing from Creative Commons-licensed media gives you much greater control. Many Creative Commons music tracks, for instance, are provided as a pack of *stems*, bundling the individual tracks used to produce the final mix. This gives podcasters a lot of flexibility, letting them repeat a section of the music, remove vocal tracks, or create an entirely different mix suitable for embedding within the podcast.

The next big question is how you intend to host your podcast. A 30-minute mono podcast is usually about 30MB. A moderately successful podcast might have 10,000 listeners, and this requires around 300GB of bandwidth per episode. Self-hosting is usually good enough to start with, especially if you can grab a domain name related to your podcast, and you can either set up a dynamic CMS such as Wordpress for posting content related to each episode, or you can use a static site generator updated for each release. Another important aspect is community feedback, and it's worth

offering several options, including email or web-based comments, alongside cheaper and easier-to-maintain options like an IRC presence, a Telegram group, or a Discord server. Spotify will do all this for you, but you lose control over how you might publish or aggregate your podcast later.

Of course, a podcast wouldn't be a podcast without an RSS feed. RSS remains central to how you publish your podcast and how people listen to it. An RSS file is an XML-formatted text file that podcast players parse to understand where the podcast is located and which episodes are available (see the box entitled "Podcast RSS Format"). RSS is simple enough to create manually, and you'll find plugins for Word-

press and other sites that can make the process even more painless. It is also easy to get your podcast included in Apple's Podcasters Program [2], Spotify's equivalent [3], or YouTube, although you'll need to agree and accept certain content, RSS-formatting, and encoding requirements for Apple and Spotify. You can bypass all of this using a podcast aggregating service or host, who will happily serve your podcasts, submit them for inclusion on the most popular services, and even negotiate advertising on your behalf. Aggregating services usually cost very little up front, although they will take a substantial cut from any advertising revenue.

On the subject of money, making any from a podcast is difficult and shouldn't be your principal motivation. You might, however, make enough money to help with hosting bills and with equipment. For open source and Linux-related podcasts, you might want to consider using Patreon [4], which enables listeners to make a small contribution to the running of the podcast. You can encourage this with additional content for backers, early access to episodes, or access to a podcast feed without any advertising. You can also sell merchandise (Figure 3) or sell advertising for the



Figure 3: One way to help fund a podcast is through merchandise, which can often be made and dispatched on demand, with the supplier taking a certain percentage of the overall price.

Podcast RSS Format

The RSS file includes links to each episode, metadata such as duration and size, and a section that links back to your general web presence (Figure 4). Listeners subscribe via the RSS URL and are pushed to new episodes as soon as they're released. The easiest way to create your own RSS feed is to either use a plugin for your CMS, such as Wordpress, or to copy one for a similar podcast. It can help to find an XML editor, such as Red Hat's XML extension for Visual Studio Code, and use it to remove all but one podcast episode, then simply replace the metadata with your own. The podcast-specific content is held with the <channel></channel> elements, with the first section used to describe the global data for your podcast, such as its name, a description, and links to images and logos. Beneath this, each episode is encapsulated within <item></item> tags that contain similar details specific to each episode.

podcast. Selling advertising is a unique skill, but if you're up for the challenge, nothing is stopping you from getting in touch with companies you think might be interested to negotiate advertising, sponsorship, or product reviews. These are usually negotiated in advance per-quarter or per six-months, depending on a company's advertising budget, and advertisers will want to see consistent listener numbers, listening hour numbers, and a certain level of engagement with their adverts – either through affiliate links or through podcast-specific coupon codes or similar.

Providing meaningful statistics is surprisingly difficult and quite different from the web-based metrics you might be familiar with. Many podcast players will open multiple streams for a single listener, cache podcasts offline, or never make it through a single episode. You can carefully craft analytics for unique IP addresses and megabytes downloaded, but it's usually easier to rely on the far more advanced tools provided by Apple Podcasts, Spotify, and YouTube, which can usually highlight the most popular

sections and also help promote engagement through reviews or comments. These numbers can then be extrapolated to include self-hosted downloads. But try not to think about the numbers too much. They can be a useful metric for growth, but they naturally fluctuate throughout the year, and it is often difficult to derive meaningful data from them. Stay true to your original plan and vision, but most importantly, have fun. Everything else will look after itself. ■■■

Info

- [1] *History of English Podcast*: <https://historyofenglishpodcast.com>
- [2] Apple Podcasts for Creators: <https://podcasters.apple.com/>
- [3] Spotify for Podcasters: <https://podcasters.spotify.com>
- [4] Patreon: <https://www.patreon.com>

Author

Graham Morrison is a distinguished Linux author and editor. He writes the FOSSPicks column for *Linux Magazine* and has been podcasting nearly every two weeks (and now every week) for 14 years, first on *TuxRadar*, which he edited, produced, and published, and more recently on *Late Night Linux*.



```

1  <?xml version="1.0" encoding="UTF-8"?>
2  <rss version="2.0"
3    xmlns:content="http://purl.org/rss/1.0/modules/content/"
4    xmlns:wfsw="http://wellformedweb.org/CommentAPI/"
5    xmlns:dc="http://purl.org/dc/elements/1.1/"
6    xmlns:atom="http://www.w3.org/2005/Atom"
7    xmlns:sy="http://purl.org/rss/1.0/modules/syndication/"
8    xmlns:slash="http://purl.org/rss/1.0/modules/slash/"
9    xmlns:itunes="http://www.itunes.com/dtds/podcast-1.0.dtd"
10   xmlns:podcast="https://podcastindex.org/namespace/1.0"
11   xmlns:rawvoice="http://www.rawvoice.com/rawvoiceRssModule/"
12   xmlns:googleplay="http://www.google.com/schemas/play-podcasts/1.0"
13 >
14 <channel>
15   <title>Late Night Linux</title>
16   <atom:link href="https://latenightlinux.com/feed/mp3" rel="self" type="application/
17   <link>https://latenightlinux.com</link>
18   <description>Late Night Linux is a podcast that takes a look at what's happening wi
19   <lastBuildDate>Fri, 09 Jun 2023 14:56:30 +0000</lastBuildDate>
20   <language>en-US</language>
21   <sy:updatePeriod>hourly</sy:updatePeriod>
22   <sy:updateFrequency>1</sy:updateFrequency>
23   <generator>https://wordpress.org/?v=6.2.2</generator>
24 </channel>
25 <image>
26   <url>https://latenightlinux.com/wp-content/uploads/2016/12/cropped-favicon-32x32.pn
27   <title>Late Night Linux</title>
28   <link>https://latenightlinux.com</link>
29   <width>32</width>
30   <height>32</height>
31 </image>
32   <atom:link rel="hub" href="https://pubsubhubbub.appspot.com/" />
33   <itunes:summary>Late Night Linux is a podcast that takes a look at what's happening
34   <itunes:author>Late Night Linux</itunes:author>
35   <itunes:explicit>yes</itunes:explicit>

```

Figure 4: The XML used within a podcast RSS feed is easy to understand, but using an XML-aware editor, such as Visual Studio Code, will help you catch any errors that might creep in.

Live Mic!

Audacity is a free, open source, easy-to-use, multitrack audio recording and editing tool perfect for podcasts.

By Ken Hess

Audacity [1] is one of those “must-have” tools for anyone who records or edits audio files. It is cross-platform (macOS, Windows, Linux), making sharing files and collaboration easier. The Audacity audio editor and recorder comes with an easy-to-use interface and works with a wide range of hardware. Audacity supports direct microphone or through-a-mixer recording in mono or stereo, and it includes some convenient keyboard shortcuts for more efficient editing. You can record directly into Audacity or open most audio and some video file types for editing. In this article, I cover direct recording into Audacity using a single source (microphone) and some basic editing. You will only use a handful of effects and features for simple podcasting. The effects I describe in this article are the ones I use most often. I’ve done some advanced editing, but those few times are exceptions.

Pro Tip: Make your podcast more appealing to listeners by removing silence, adding intro and outro clips, and adding background music.

One of the most-often-asked questions about podcasting is, “How much does it cost to start a podcast?” The answer sounds sarcastic, but it ranges from \$0 to thousands. It’s \$0 if you have a functional computer running Linux, use an on-board microphone, and install Audacity. Hosting might also be free if you use one of the free online aggregation services. These services take your podcast and distribute it to seven or eight of the big podcast providers, such as Google Podcasts, Apple Podcasts, Amazon Music, and others. You could also purchase a new computer, a new high-end microphone or set of microphones, a mixing device, and you get the idea. If you have a computer you can use, install Linux on it, install Audacity, and buy a good microphone. The microphone’s price often reflects its quality. However, a microphone at the \$100 price point will serve you well. You can



always upgrade later should you acquire sponsors or followers who donate to your show.

Audacity’s official features include the ability to:

- Record live audio
- Convert tapes and records into digital recordings or CDs
- Edit WAV, AIFF, FLAC, MP2, MP3, or Ogg Vorbis sound files
- Export and import AC3, M4A/M4R (AAC), WMA, and other formats using optional libraries
- Cut, copy, splice, or mix sounds together
- High quality sound recording at 16-bit, 24-bit, and 32-bit
- Add numerous effects, including changing the speed or pitch of a recording, as well as Fade In, Fade Out, and Truncate Silence
- Use LADSPA, LV2, Nyquist, VST, and Audio Unit plugins
- Write your own custom plugins

Using Audacity

I’ve used Audacity since 2008 when I first started podcasting. It was already a mature product by then and had every feature and plugin I needed to create and edit podcasts and other audio files. The two most important aspects of Audacity for podcasting are recording and editing. Everyone will tell you that sound is the most important part of a film or video, but even more so for a podcast with no visual cues to derive meaning or context. Your sound quality must be superior. You can improve the quality of poorly recorded audio using Audacity’s features and plugins, but it will always sound less than ideal.

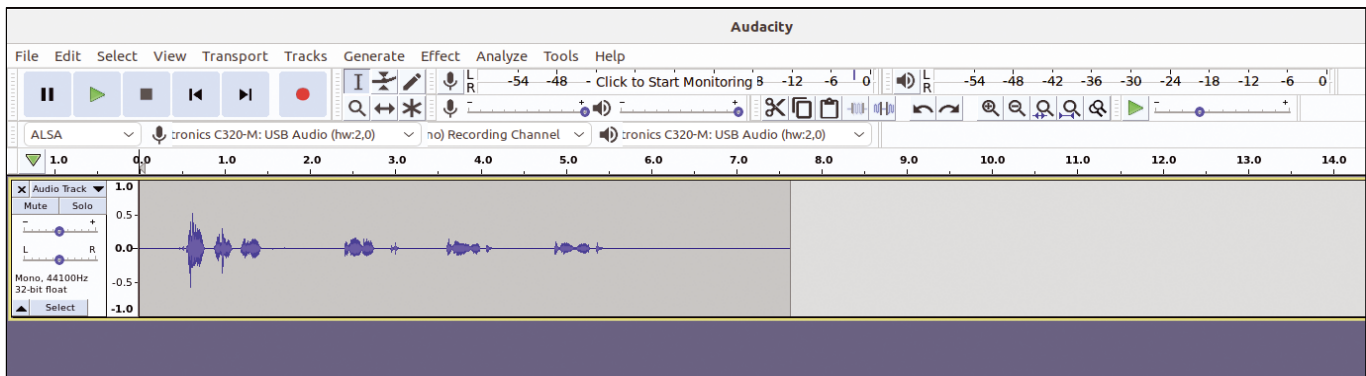


Figure 1: Creating a new recording in Audacity.

Quality sound begins with quality equipment, and I cover recording hardware choices in the next section.

Operating Audacity is simple. Because most laptop computers have an internal microphone, you can start recording when Audacity opens on your system. Press the Record button and start talking. You'll see the sound waves appear on the screen as you speak (Figure 1). Voila! You're a podcaster! Well, not quite, but you're on your way.

As a podcaster, you'll likely want or need to use some type of effect, such as Truncate Silence, Amplify, or Noise Reduction. To apply an effect on an entire file, press **Ctrl + A** to highlight the entire file, click *Effect* from the menubar (Figure 2), and then select your effect. Most effects have their own settings that you can adjust to suit your needs. Before you apply an effect, click the *Preview* button to listen to a sample of the applied effect using your custom settings. Adjust and preview until you are satisfied with your selections, and then click *OK* to apply the effect. You can apply effects to a portion of your audio file, too. To do so, highlight the area you want to adjust, and then repeat the above steps to apply an effect. Remember the universal undo key sequence, **Ctrl + Z**, if you want to reverse a change.

A word of caution if you use Noise Reduction to remove hiss or static in a file. Audacity does a good job of removing these anomalies, but there is a cost. Your audio might sound "tinny" or unnatural if you must significantly reduce noise. It's better to record your audio again rather than annoy your listeners with a

robotic voice – unless that's what you're going for.

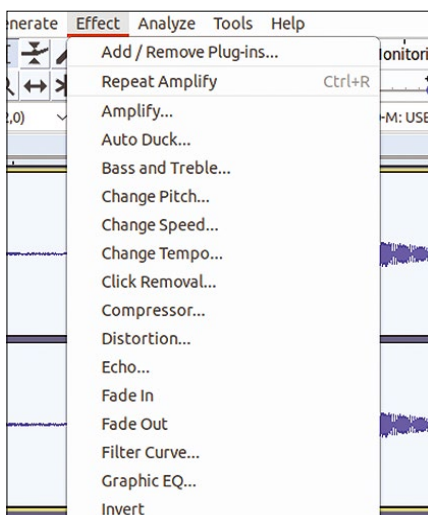


Figure 2: A partial list of Audacity's default effects.

Recording Hardware

Although I have used a standalone microphone and speakers, a headset with microphone and lapel microphones, and a headset or earbuds with no microphone, I prefer a headset with a built-in microphone for recording my podcasts. I currently use a C320-M headset by Plantronics (see Figure 3), which is now part of the electronics vendor Poly [2]. It is a well-built, low-noise headset that allows me to hear myself speak. Noise-canceling headphones often distance me from the conversation because I can't fully hear myself speaking, so I don't know how to modulate my voice for the recording.

To see a list of your attached devices, select *Edit | Preferences* from the main menu options.

Pro Tip: Remember your branding. Mention the name of your podcast multiple times during your podcast.

I also have a Focusrite Scarlett 2i2 audio interface [3] (shown as an available device in Figure 4) that handles two input sources. Audacity sees this device as a single source. If you have multiple speakers, each needing a microphone, you'll need an interface device to connect multiple inputs. Using this device, I can attach two microphones and record them simultaneously. An interface device is a versatile choice because you can use XLR inputs or 1/4-inch TRS connectors, and various adapters exist for each.

I like cardioid lapel microphones or headsets with built-in microphones for multiple-speaker podcasts. The speaker can relax more and speak normally rather than trying to stay an inch from a stationary or handheld microphone. The lapel microphone picks up the speaker's voice without them having to

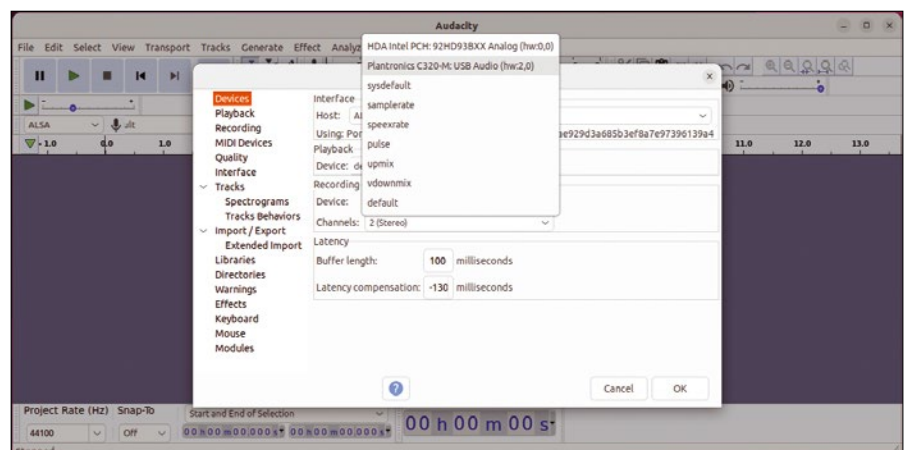


Figure 3: The Plantronics headset in Audacity's device list.

focus on the distance and positioning, which gives the podcast a more natural and conversational tone. Additionally, without so much focus on microphone mechanics, you'll likely have fewer pauses and fillers to remove from the recording.

Audacity is compatible with almost every piece of recording equipment compatible with your computer. Generally, if your computer recognizes it, so will Audacity. Remember to make careful and thoughtful hardware choices by reading reviews and asking questions about how well the components work with Audacity.

Dealing with Audacity's Quirks

Audacity is a superb audio program with some interesting odd-ball features. The most notable quirk is that you must attach record and playback hardware before you start Audacity. Audacity won't recognize any hardware plugged in while it's active. Your system will recognize and make the new hardware available to you, but if Audacity is open when you plug in the device, you won't see it in the device list. Attach microphones, headsets, or other audio devices and then open Audacity. The new hardware will appear in the device list. Once attached, you can switch between audio devices without closing and reopening the application. For example, if you attach an external microphone and then start Audacity, you can use the external microphone for a recording, switch to the onboard microphone, and continue recording without closing and reopening. Be careful when selecting Playback and Recording devices from your lists. You won't damage anything by selecting a microphone as a playback device but selecting the correct playback device will prevent hours

of frustration and troubleshooting.

Another significant quirk is that you can't record over a particular spot in a record – or at least I have never figured out how to do it. For example, suppose you record a conversation and want to record over a mistake. In that case, you might attempt to place your cursor to the left of the piece

you want to record over, hit record, and expect your new recording to be inserted where you've placed the cursor. You'll find that your recording begins at the end of your current recording. You must copy, cut, and paste your new clip into the recording to replace the bad audio. However, you can highlight your new clip, copy it (Ctrl + C), highlight where you want to replace the bad clip, and press Ctrl + V. Your new clip will paste into your original clip replacing your bad audio.

If you press the Record button again and begin talking, Audacity adds your new recording at the end of your previous clip. My cursor is near the 2.0 second mark but the new clip was added to the end of the first one (Figure 5).

Pro Tip: Sponsorships are great, but long spots make an audience lose interest. Two or three 10-second spots are better than a single 30-second one.

The final quirk I'll mention is that when listening to an audio file, you might need to stop and come back later and pick up where you left off. If you expect to resume listening, do not press the Stop button. Press the Pause button instead. If you press the Stop button, Audacity returns you to your cursor, which could be several minutes or hours from where you started. If you press Pause to stop listening temporarily, press Pause again to resume listening. I have listened to my podcast recordings, some stretching to over an hour, only to press stop and then forget where I stopped. It is an annoyance that only needs to happen a few times to train you to press Pause instead.

Advanced Features

Audacity has some interesting advanced features. You can find a few of these under the *Generate* menu (Figure 6), including Chirp, DTMF Tones, Noise, Silence, and Click Track.

The one I use most is Silence. I often add a few seconds of silence at the beginning of an audio file I know I'm handing off to another person for production to allow for intro music, sponsored ads, or voice-over mixing. To add silence to a track, click the mouse to set the cursor. Then, select *Generate | Silence* and provide a duration (Figure 7). An interval of silence will appear in the track (Figure 8).

Saving Projects and Exporting Audio

After you edit your audio file, you'll want to save it. You have some options for doing this. If you need to save your project and come back to it later, preserving your edits and history, select *File | Save Project | Yes*. This option will save your

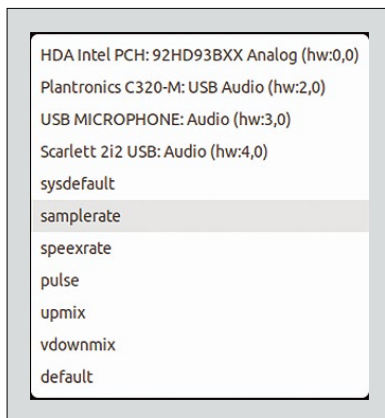


Figure 4: Viewing the list of available devices.

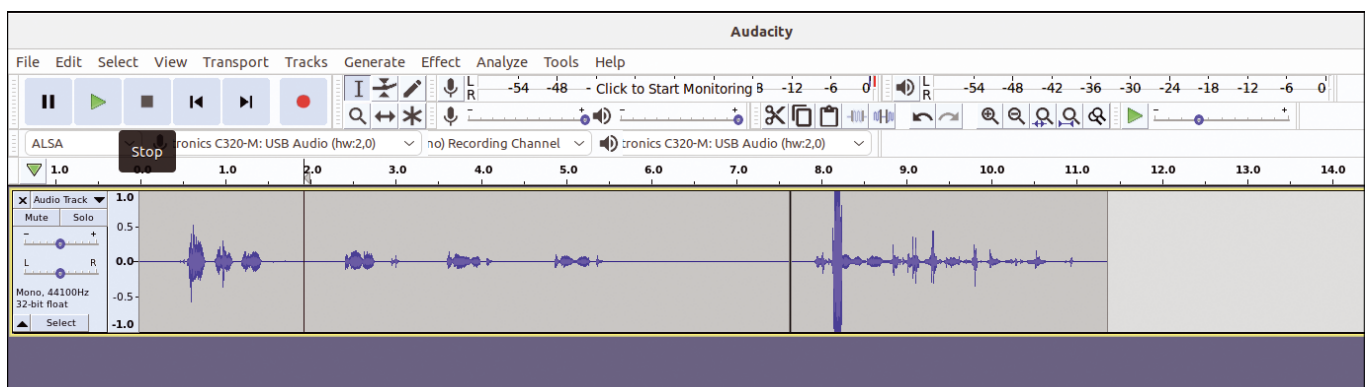


Figure 5: Attempting to record over part of a clip.

tracks just as they are now and preserve your edit history. The file saves with an AUP extension. You can save your project and still export the audio for playback on other devices. You will receive a notice stating this is an audio file, not a project. Click *OK* to acknowledge the message and continue saving your project file.

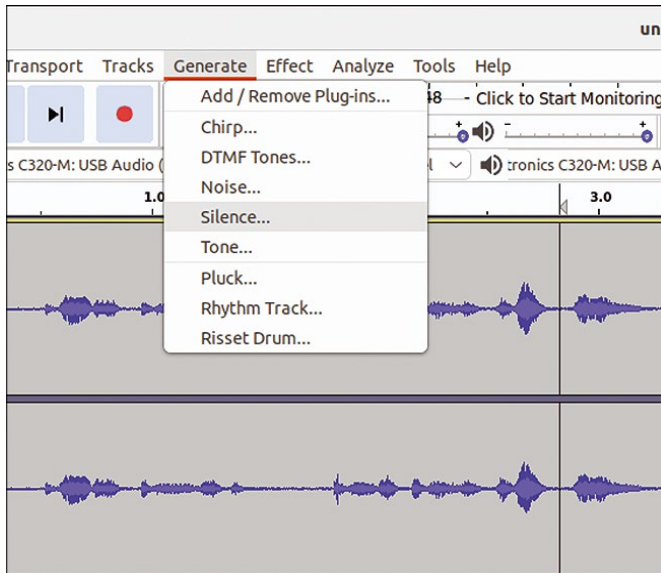


Figure 6: Audacity's *Generate* options.

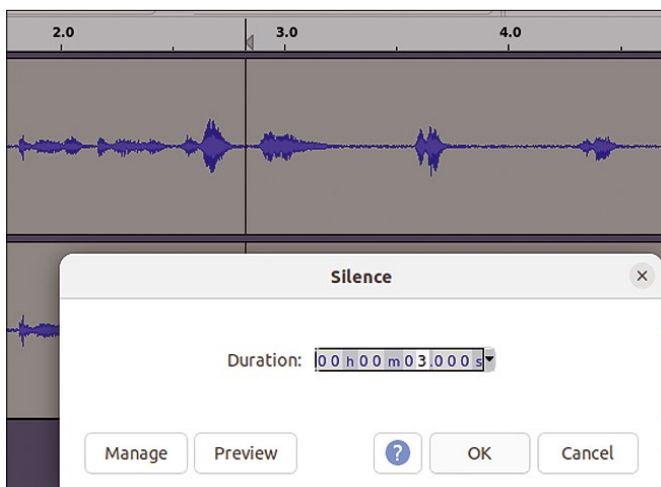


Figure 7: Inserting three seconds of silence at the cursor's position in a track.

To save your file as a completed audio file, select *File | Export*. A dialog opens, prompting you to select a file type. I always export as MP3, but that's a personal preference.

You may also download the source code for Audacity [4] and compile it on your system to enable or disable options and customize your installation. I have never needed to do this because the standard features and locations have always worked for me.

Summary

Audacity is a professional audio application that supports a broad range of recording and playback hardware. It is a free, open source, easy-to-use application with advanced features, dozens of effects, plugins, and output options. It's an excellent choice for podcasters who need an uncomplicated solution that works equally well on Linux, Windows, and macOS systems. The Audacity development team releases regular updates and security and bug fixes. Working with Audacity is fun and productive. It will serve you well for many years. Happy podcasting! ■■■

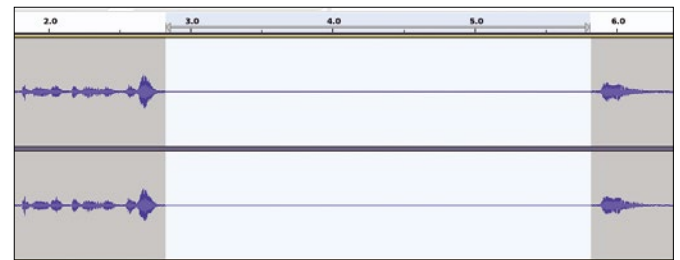


Figure 8: Three seconds of silence added to the track.

Info

- [1] Audacity: <https://www.audacityteam.org/>
- [2] Poly: <https://www.poly.com/us/en>
- [3] Focusrite Scarlett 2i2: <https://focusrite.com/en/usb-audio-interface/scarlett/scarlett-2i2>
- [4] Audacity source code: <https://www.audacityteam.org/download/source/>

Author

Ken Hess is a freelance technical writer and journalist. He covers a variety of open source topics, including Linux databases, and virtualization. You can reach him via his website at www.kenhess.com.

Sound Explorer

Once you get your podcast operation up and running, you might decide you want a real mixer and some higher-end software. We'll introduce you to Ardour and get you started with some basic audio hardware. *By Hartmut Noack*



Podcasting often occupies a space on the low end of the audio recording spectrum. All you really need is a computer, suitable software, and a microphone. Some users, however, might wish to avoid this most basic scenario. You might ask, “If I’m going to all the trouble to set up a studio for podcasting, what else can I do with it?” Or you might want your studio to grow into something more sophisticated. Perhaps you are a musician yourself? Or you wish to host live music – or other performance alternatives that require mixing and multiple mics.

In these scenarios, you could easily outgrow the simple mic + Audacity configuration. If you’re looking for something more, try Ardour.

Ardour [1] is still considered the reference application as a free tool for recording and mixing on Linux. A variety of alternatives also exist – for instance, Muse is a good option if you are composing music with MIDI, and Qtractor offers some innovative features for working loops and samples – but when it comes to mixing and mastering, Ardour is unsurpassed.

This article describes how to get past the most basic podcast configuration and equip your studio for live recording.

I’ll also describe some of the hardware you might want to incorporate into your studio. Although the setup described in this article might be more detailed, and more sophisticated, than a basic Audacity configuration, it is still relatively simple and inexpensive. Professional recording studios spend *way* more money on hardware.

Mastering Live Performance

Mastering, putting the final touch to a finished mix, requires something more than “it sounds pretty good on my headphones.” If you want to make sure that the mix still sounds good enough on a single smartphone speaker while performing well on various stereo systems, you need to test it before releasing it.

Recording live performances raises another issue. Laptop concerts are commonplace, and these kinds of performances can easily be managed today using free software on Linux and the stereo output of a simple USB interface. But today’s technology gives you far greater opportunities. The options range from simple pre-monitoring options for DJs that let you sync the sound source precisely with the beat to analog mixers as musical instruments.

Hardware

The step from a simple USB microphone to a mixer is affordable nowadays. I used the Mackie ProFX10v3 [2] mixer for this article, priced at around EUR250 (Figure 1). The device is the result of more than three decades of experience in building analog mixers. In addition to all the qualities you can expect from it, there is also a built-in USB interface that goes about its tasks without treading on anybody's toes.

Thanks to the JACK audio server built into most Linux systems, generic USB devices of this type now work well,



Figure 1: Mackie's ProFX10v3 mixer only has dials instead of channel faders, but it can sit next to a normal laptop on your desk.

but the generic ALSA driver does not provide controls for the device's software mixer for some devices. The ProFXv3 also shows up as a mixer section in QasMixer; in other words, everything this piece of hardware can offer on a computer is fully available (Figure 2). For the specs, see the "ProFX Mixer Details" box.

The first thing you notice about the ProFX10v3 mixer is its very solid build. All of the controls, switches, and connectors are easy to reach and leave the user with a reliable impression. Nothing wobbled, and I didn't encounter any annoying noise

during operation. Everything is clearly legible and sensibly labeled. Thanks to illuminated mute switches, the ProFX10v3 mixer is also useful for live gigs in poor lighting conditions.

The 10 in the name ProFX10v3 stands for the analog inputs whose stereo output is routed to the two PCM inputs of the USB interface. The mixer's master

ProFX Mixer Details

The mixer offers a great deal on a small footprint. To use the inserts in the first four channels, you need a stereo jack plug to route one channel out with the other receiving the processed signal (Figure 3). The FX in the mixer's name stands for digital effects computed by the GigFX engine in the mixer. These effects can be inserted into each channel. For the complete version 3 series manual, visit the Mackie product page [3].

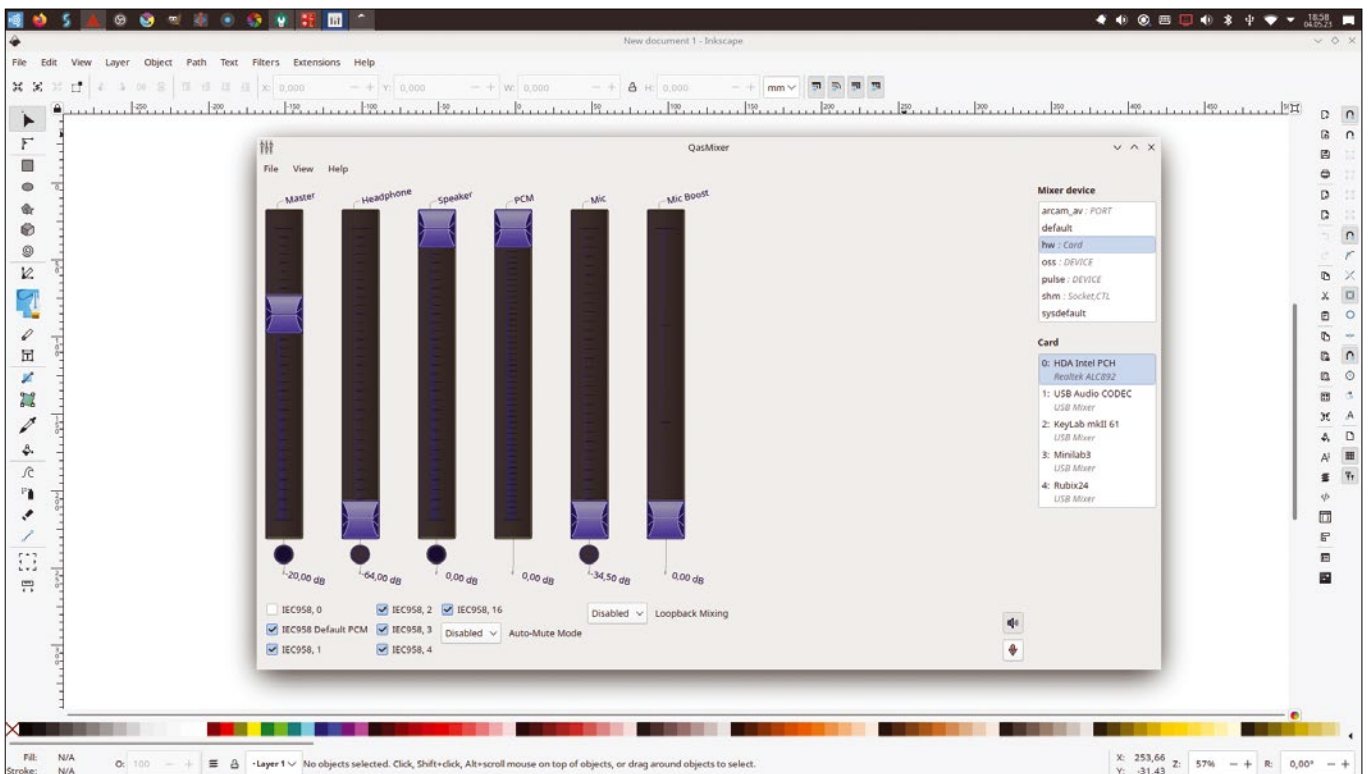


Figure 2: Immediately after connecting, the ProFX shows up with two inputs and four outputs for *jackd* (with the Carla audio plugin host here), which you can control with QasMixer.



Figure 3: The inserts cannot be controlled; they only loop the signal into the channel. The switch below is responsible for the Low Cut EQ.

control does not affect the signal's volume, which means you can record the overall mix of a band, or two channels at once, discretely by turning the balance controls all the way to the left or right.

Scenarios

The first thing you need to find out is how to use the hardware in combination with Linux, the *jackd* package, and the application stack. The 2x4 interface lets you output the sound sources to two separate targets: *Main Mix* and *Control Room*. The *Control Room* target also serves the headphone output, to which you could connect another amplifier and speakers if required (Figure 4). When you are working on a recorded mix, this lets you use different output systems at the same time. I normally use

promise even if you route the main mix out 2 to 3 meters away using simple jack cables.

Many of today's popular sound manglers combine incoming stereo signals into a single channel either physically or in the software. Many errors that you will not notice in stereo playback do not sound right if output in mono. Once your mix sounds good on decent stereo monitors, set up the mono output in the same way, switching between the two output modes. If your work also sounds acceptable in mono, you can assume that it will sound okay on a smartphone speaker.

The live performance scenario is far more complex. Whether you want to stream live on Twitch and add music, voice chat, or mix a band to perfection, the options that a mixer like the ProFX offers for controlling the signal flow quickly become

two active near-field monitors by Tannoy for tests. These classic speakers allow very accurate and analytical listening, although they can also be a bit unforgiving because they reproduce the sound as naturally as possible. On the ProFX, you can test what the same mix sounds like on a stereo system at the push of a button. You can also connect Bluetooth speakers and similar playback devices, as long as they have an analog input.

The main mix is sent from the mixer via balanced XLR cables, which guarantee an interference-free signal at distances of 50 meters or more. One of Mackie's trademarks is that the mixers are totally silent if no signal is being transmitted. The ProFX delivers on this

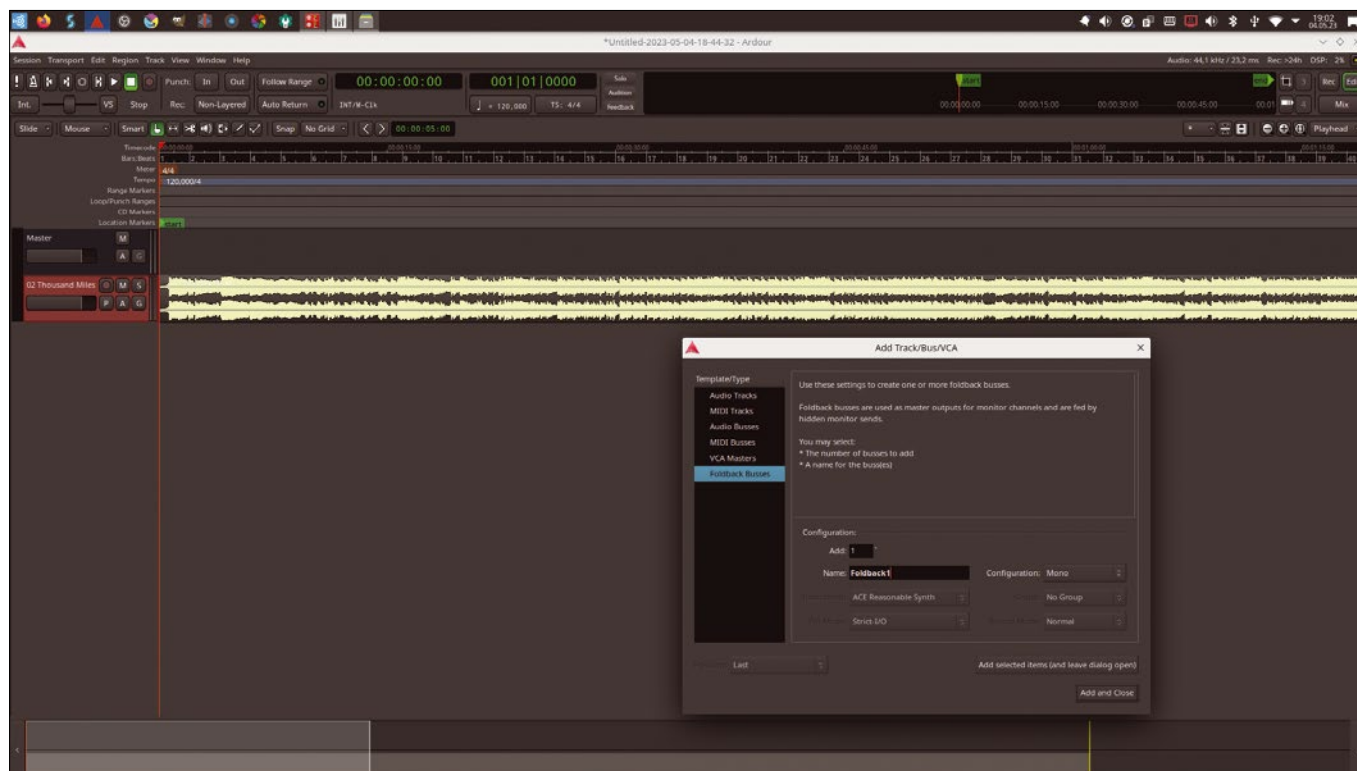


Figure 4: In Ardour, a foldback bus effectively gives you a second master output in the mixer that you can route to another output.

indispensable. You will soon discover that it is much more satisfying to work with real dials and buttons than with virtual buttons on the screen.

The insert connectors in the first four channels of the ProFX are a rare feature in this price range. The connectors allow the signal to be routed to external devices, which then process it

and feed it back into your mix. All of this reaches Linux via JACK as a stereo mix and can be recorded in Ardour.

But you can also send signals from any JACK-enabled Linux software to the sound systems connected to the mixer output. This means that a band can use, say, the Hydrogen drum computer while simultaneously playing a soft synth like Yoshimi or

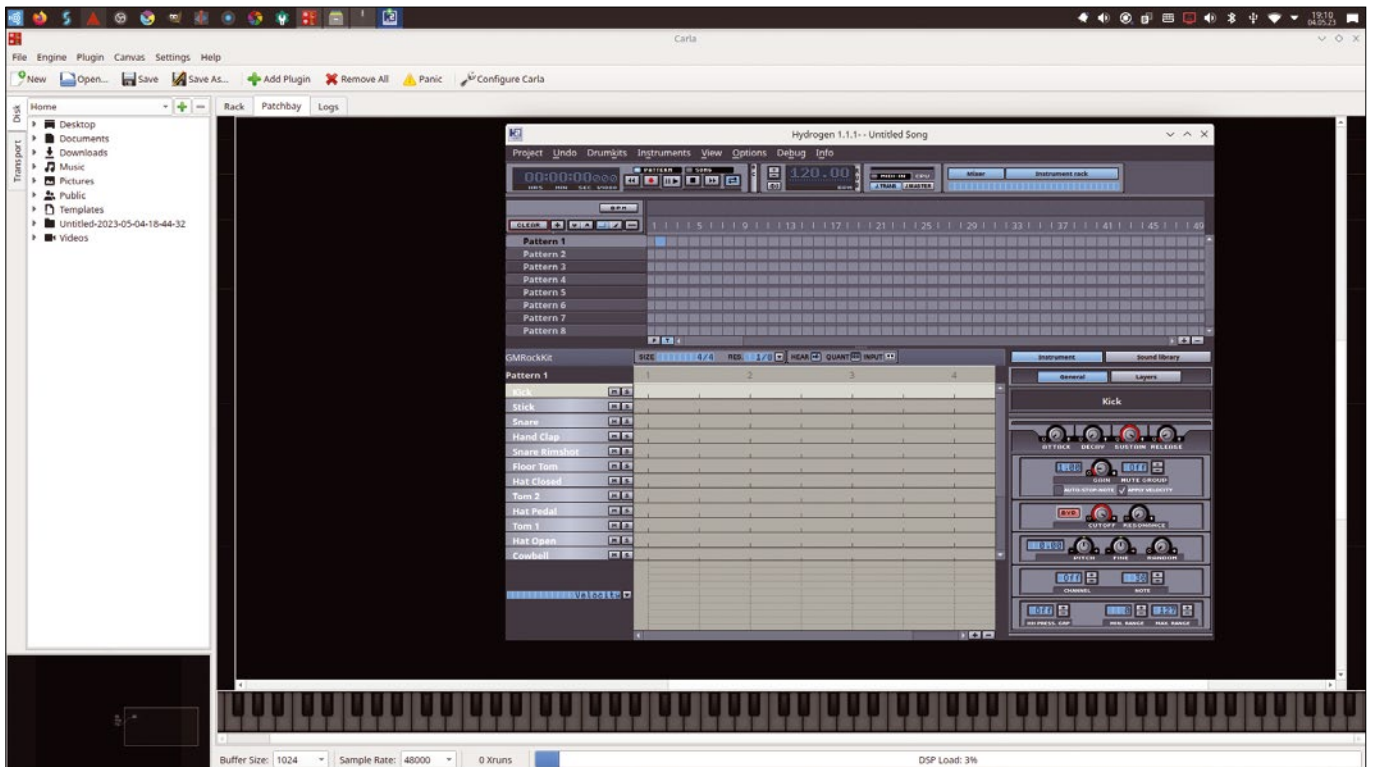


Figure 5: Ardour has much work to do here; Hydrogen and Yoshimi are wired in parallel in Carla.

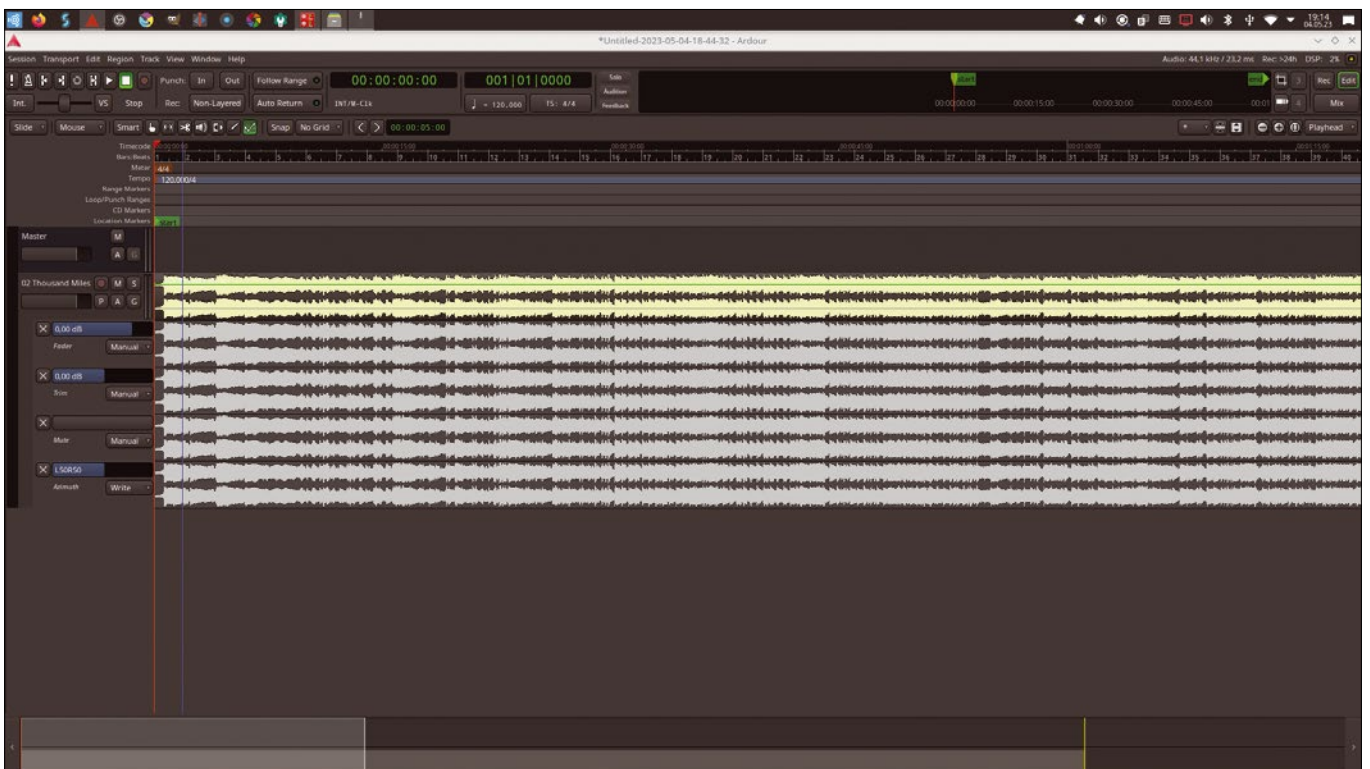


Figure 6: No matter what you want to adjust at a particular point in time, you automate it with intuitive curves in Ardour.

VeeOne via MIDI keyboard – along with signals entering the mixer via microphone or guitar (Figure 5).

Mix and Master

Ardour offers extensive automation options for mixing and mastering. Every single parameter of both the mixer itself and all plug-ins can be programmed in parallel with the audio using intuitive curve tracks. You can draw the curves directly in the software interface; a MIDI controller or an OSC-enabled smartphone app are also suitable for this task.

In the Editor, clicking A opens a menu that lets you show the meta-tracks for the track; you can then draw in curves for virtually any parameter, from the volume to totally obscure plug-in settings (Figure 6). In addition, Ardour's signal routing, which fully relies on JACK, gives you flexible control of practically any output channel; this feature is not found in any other, even commercial, software for Linux. A second (or more) master channel can also be implemented as buses with relatively little overhead (Figure 7).

For mastering, you mainly need plugins for the equalizer and compressor/limiter. Much has happened in this sector in terms of free extensions for Linux in the last two or three years. In addition to simple classics from the LADSPA era like the FooLimiter and high-end convenience plugins from Calf, you will find new modules that offer even more settings and elaborate, well-designed graphical interfaces of very high quality (Figure 8).

Hear, Hear!

One way to listen to audio while mixing and recording is to use headphones. For tests, I often use AKG's classic K240 DF and a fairly new offering by the name of MC-350 (Figure 9) by Mackie.

The MC-350 headphones differ significantly from AKG by offering noise-canceling. The design is much like that of earmuffs

for hearing protection, and it prevents the signal mixing with the ambient noise. When recording voices or guitars, this gives you a sound that you can't achieve at all with speakers and only to a limited extent with open headphones. You can hear the recorded signal without it mixing with the sound output in the room at the same time.

The result is sometimes amazing. A guitar voice amplified by guitarix (a guitar amplifier for Linux running on JACK), which seems beautifully round and musical in the test with Tannoy's speakers, sounds totally distorted and lacking distinguishable harmonies through the Mackie headphones, and this is exactly what the recording actually sounds like. This is definitely not because the Mackie somehow can't handle the signal. The headphones don't distort anything, they just relentlessly reproduce the settings selected in guitarix. If you listen to the track through the speakers, it mixes with the internal sound of the Ibanez RG guitar being played, with these well-defined acoustic sounds merging with the fat crunch of guitarix.

In a nutshell: Especially when you are setting up effects for acoustic signals, it is a good idea to use headphones. This revealing effect is less extreme with the open AKG headphones. But you can also discover the error by simply turning up the volume a bit.

The Right Speakers

If you produce audio yourself, you should be careful not to be too swayed by equipment trimmed for good looks. Mixes made with mid-range hi-fi equipment have a strong tendency to sound poorer, duller, too spiky, and thus unlike what the artist intended.

Monitors for sound engineers sound analytical. They don't emphasize what the listener finds particularly brilliant, and

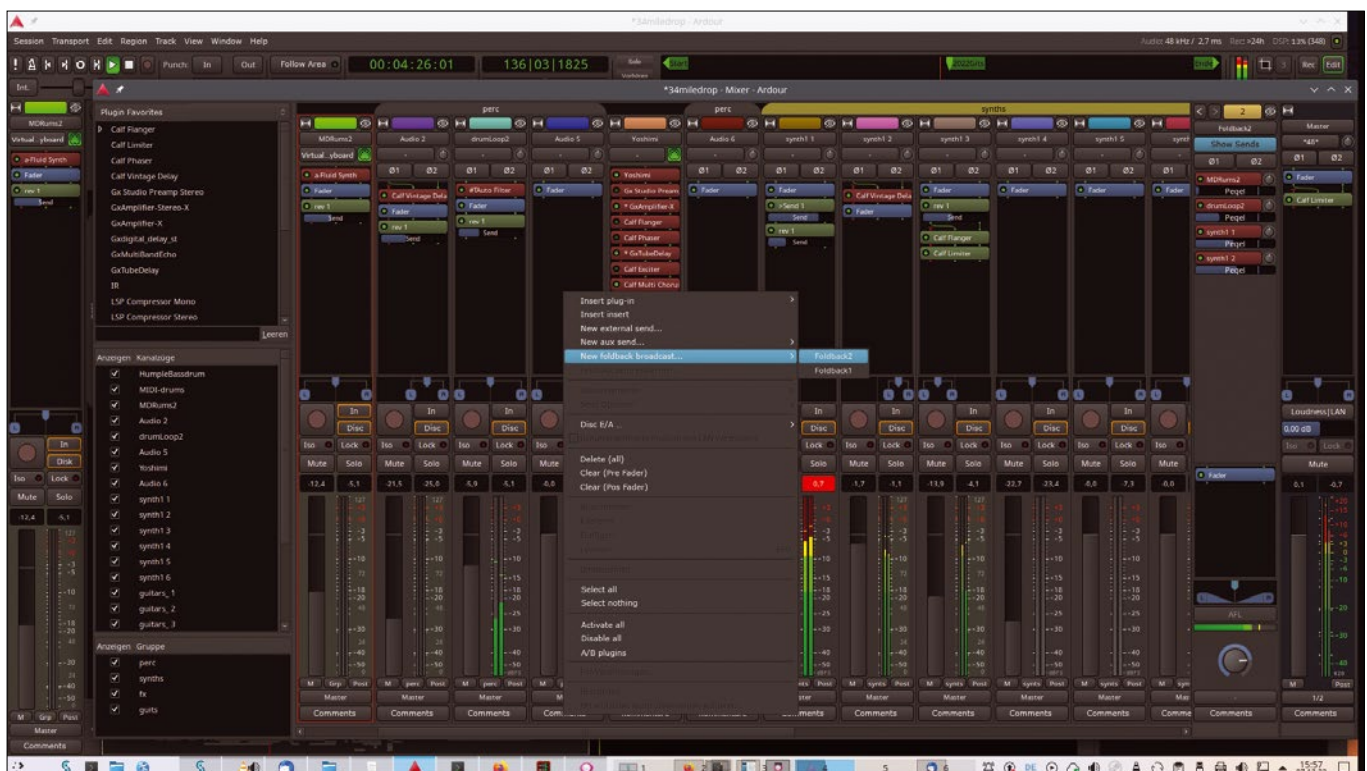


Figure 7: The Ardour mixer follows a philosophy that says the most annoying feature is the one that's missing. In terms of signal routing, it's hard to imagine more possibilities.



Figure 8: If you are missing any adjustment options in Calf's mastering plugins, take a look at LSP's EQs and dynamics processors. In terms of signal routing, it's hard to imagine more possibilities.

they don't exaggerate the thrust that bass produces. To achieve this, they avoid distortion and coloration to the extent possible and reproduce sounds in a particularly well-defined manner – spatially, so that the individual voices can be precisely balanced in the mix.

I tested these capabilities with a classic pair of speakers, the 2-way Tannoy Reveal Near-Field Studio Monitor system with built-in 50-watt amplifiers (Figure 10). I set up the boxes in an triangle at a distance of about 1.20 meters from the workplace (see the "Speakers Set Up Correctly" box). This way, you can hear everything, including the errors. The spatial representation seems noticeably stronger than with the stereo system

setup in the same room featuring a Denon amplifier and two 3-way hi-fi speakers by Canton.

The Tannoy speakers are available second hand for about EUR200. Together they weigh 10 kilos and expect a signal from a jack

cable fed in by a decent interface. Can a small, lightweight compact device like the StealthBar by Mackie for

Using the StealthBar Correctly

Fortunately, the small compact system supports every possible connection option. The StealthBar can be used as a good USB master directly with JACK, as well as a plain vanilla output device for another interface. Integrated as a master via USB with JACK, it is unobtrusive. On the back, there is a mini-jack input that feeds the stereo-in of the USB interface. As a nice extra, the StealthBar offers a Bluetooth interface. No problem for PulseAudio: Immediately after pairing, sound output from the browser is audible. Unfortunately, this Bluetooth port is not integrated into JACK. Even if you ignore the additional latency of PulseAudio, you can't listen to sounds from Ardour and other JACK applications in this way. But what if you want to use JACK with a different interface? In this obvious case, you need to connect the stereo output of the device to the back of the StealthBar via the mini-jack. Then, just like with Tannoy's classic monitors, the StealthBar works as an active analog monitoring speaker.

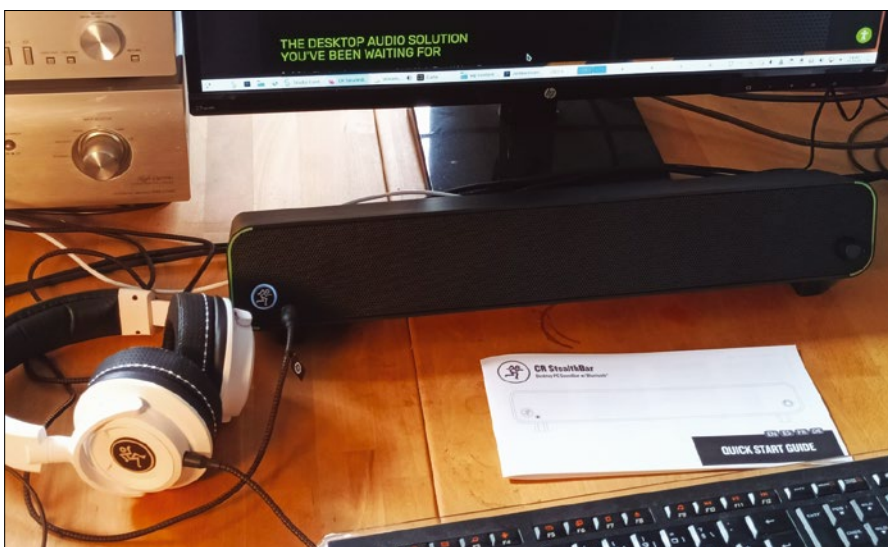


Figure 9: The Mackie headphones, shown here next to the StealthBar speaker, are also available in white.



Figure 10: Classic near-field monitors usually sound better and almost always more powerful than small compact solutions, but they weigh about 5 kilos each.

EUR100 match this? As the name suggests, the StealthBar is a compact bar that is best positioned under the screen. The device is a USB interface and a Bluetooth speaker at the same time (see the box entitled “Using the StealthBar Correctly”). Although the StealthBar is only 50 centimeters wide, it delivers good stereo mapping, even if it requires a bit more concentration to implement the spatial impression with the small speaker.

Analyzing the Room Acoustics

If your own space is acoustically quirky and not representative of a general listening environment, tailoring the sound so it

Speakers Set Up Correctly

One of the biggest advantages that headphones offer is their ease of use: No matter where you put them on, they always sound the same. Compared with this, the sound waves output by speakers into the room are reflected by the walls, and the sound also strongly depends on the positioning. The most important rule of thumb is that you need to sit exactly in the center between the speakers. The ideal is be an isosceles triangle, the size of which you need to adjust to the physical properties of the speakers. The StealthBar, on the other hand, is optimized for a short distance; you get the best spatial impression at a distance of about one meter. It still sounds OK at a greater distance, but locating voices in the stereo mix is more difficult. For Tannoy’s near-field monitors, the manufacturer recommends an isosceles triangle at head height with a side length of about two meters. In the absence of a very large room and proper tripods, I have the Tannoys set up at opposite ends of a desk with a width of two meters. To listen really intensely, I have to roll my chair away from the desk slightly.

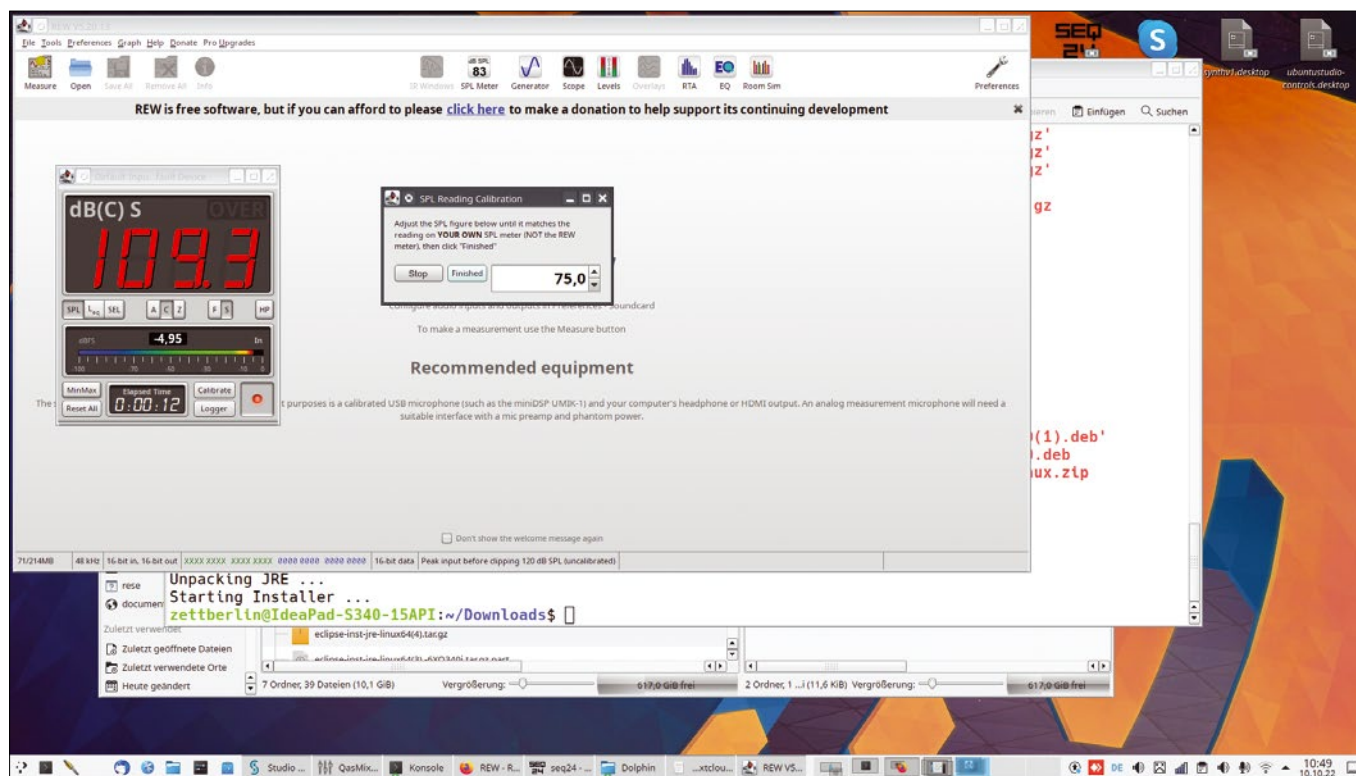


Figure 11: REW measures the capabilities of your microphones so that the analysis functions can also be used with the existing equipment.

sounds perfect in your studio will not make it better for everyone else. You can find software solutions that will let you measure and analyze the sound output in the room accurately. I used the Java Room EQ Wizard (REW) app in the test; you can download it free of charge [4]. You will also find a commercial Pro version of the REW proprietary freeware [5], which – above all – enables more precise measurements, as needed when building loudspeakers. Having said this, the free version comes with everything you need to optimize a home studio. It generates the required measurement tones and meaningful graphs of the measured signals.

On first launch, REW offers to calibrate the connected microphone (Figure 11). It then measures the sound waves that the speakers spread in the room and graphically displays whether and how the results deviate from an expected ideal result. You can reduce these deviations by removing obstacles near the speakers and dampening reflections in the room. Reflections not only create reverb effects that wash out the sound image, they also cancel out or boost frequencies, and that distorts the sound entirely.

Optimizing room acoustics is an expensive discipline if you want everything to be perfect. Fortunately, you can get very close to perfection even with simple means. Soft carpets and heavy curtains often already work wonders, and you can easily prevent the sound being wiped out in the corners of the room using wall hangings. As a last resort, you could use an

equalizer upstream of JACK's master output. This setup is tricky, and it generates more system load.

Conclusions

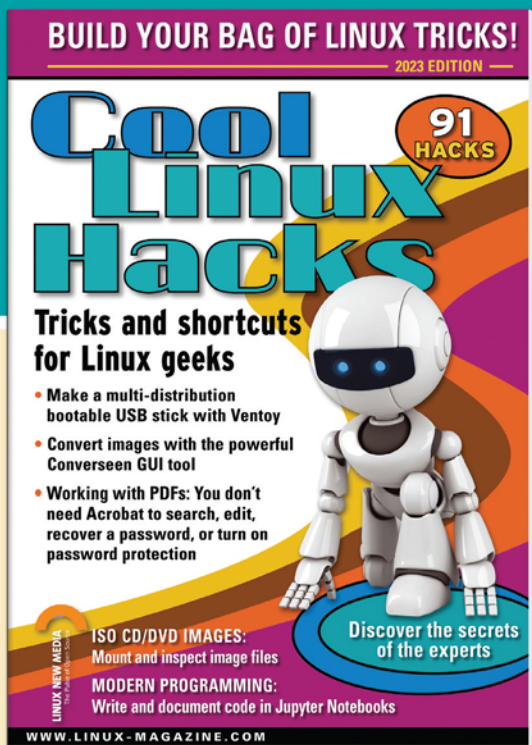
If you get past the simplest podcast scenarios, you will find that USB mixers work like a charm on Linux today. Devices that offer complex signal routing are easy to wire thanks to JACK and Carla. Finally, Ardour implements all of these options in a way that is typical of premium music production and performance. ■■■

Info

- [1] Ardour: <https://ardour.org/>
- [2] Mackie ProFX10v3: <https://mackie.com/en/products/mixers/profxv3-series/ProFX10v3.html>
- [3] Manual for the Mackie ProFXv3: https://mackie.com/img/file_resources/ProFXv3_OM.pdf
- [4] REW: <https://www.roomeqwizard.com/>
- [5] REW license terms of the proprietary freeware version: <https://www.roomeqwizard.com/eula.html>

Author

Hartmut Noack works as a lecturer, author, and musician in Celle and Hannover, Germany. You will find some CC-licensed results of his work with free music software on <http://lapoc.de>.



SHOP THE SHOP
sparkhaus-shop.com

GET PRODUCTIVE WITH
COOL LINUX HACKS

Improve your Linux skills with this cool collection of inspirational tricks and shortcuts for Linux geeks.

- Google on the Command Line
- OpenSnitch Application Firewall
- Parse the systemd journal
- Control Git with lazygit
- Run Old DOS Games with DOSBox
- And more!



ORDER ONLINE:
sparkhaus-shop.com/specials



Exploring the TUXEDO InfinityBook 16 Gen7 MK1

DRESSED UP

The next-generation laptop from TUXEDO is faster and lighter than previous business models. *By Joe Casad*

In the old days, we all thought nothing of buying a retail Windows laptop, reformatting (or shrinking) the partition, and setting it up with Linux. Outcomes varied, but eventually you could get it working. But how much time did you spend removing an operating system you didn't want just to install the system you did want? When Linux started to reach the mainstream, it became clear the world needed reliable, stable, secure options for Linux users who want to boot their computer into their preferred system without the hassles and complications.

The list of retailers selling pre-built Linux laptops includes giants like Lenovo and Dell, but also includes some smaller vendors who specialize in Linux systems and therefore make Linux the sole focus of their attention. We've reviewed a number of the products in this sector in previous issues of *Linux Magazine*. Recently we received the TUXEDO InfinityBook Pro 16 Gen7 MK1 system [1], which is listed for EUR1,224 at the TUXEDO website.

TUXEDO has been popular in Europe for years, and it is becoming more of a presence within the North American market. Like some of the other Linux special-

ists, TUXEDO maintains its own Linux distro, which they call TUXEDO OS.

TUXEDO says their InfinityBook MK1 system (Figure 1) is for "content creators, software developers, or business users." The review system runs at 4.6GHz, with 14 processor cores, 31GB of memory, and a Samsung SSD 980 1TB hard drive. Side ports on the

ultra-thin InfinityBook include (2) USB A, (1) USB C, an SD card reader, 2-in-1 audio (headphone + mic), and HDMI 2.0, as well as a Kensington lock connection and a Thunderbolt 4 port. An Intel Mesa integrated graphics card works very well for the kind of basic gaming I do. If you're looking for something more, you can upgrade to an NVIDIA GeForce RTX 3050 or buy their "maximum performance" model, which comes with the GeForce RTX 3060.

Ready for Work

I must admit there is a certain joy in booting a system and watching it come up in Linux, without reformatting any drives or working around the UEFI boot secure boot protections. There is even more joy in booting up the TUXEDO system because it boots *fast* – faster than a Windows system and faster than most Linux alternatives. Some commentators report TUXEDO systems booting in as little as six seconds. In my case, it was more like 16 seconds to the login screen, and then fast completion after entering credentials.

A neutral Linux designed to boot on any hardware must drag along components, features, drivers, and settings you don't really need if you have full knowledge of the hardware. Consequently, a pre-built system like the InfinityBook that delivers hardware and Linux together often comes with better performance and more efficient power usage. The engineers at TUXEDO have prioritized fast boot, and their close knowledge of the hardware allows them to



Figure 1: The InfinityBook Pro 16 Gen7 MK1 is big enough to work on all day, but still very light and easy to carry around.

optimize the startup in ways that aren't possible with a generic system.

The InfinityBook MK1 is designed to be light and easy to lug around. It weighs in at only 1.5kg, and it is only 17mm thick. It feels more like a tablet in my laptop case than a full-sized computer with a 16-inch 2560x1600 pixel display.

Software

The review system ships with TUXEDO OS 2, a major upgrade of TUXEDO's in-house operating system. TUXEDO OS 2 is based on Ubuntu with the KDE Plasma 5.27 desktop. The system comes with a standard collection of Linux office tools and desktop utilities, including the LibreOffice suite, the Okular document viewer, and the VLC media player. The default browser is Firefox, and the mail client is Thunderbird. If you're a gamer, you'll want to install the Lutris open gaming platform, which is available in the Tuxedo OS repositories. You'll need to set up the Steam client separately if you want to play Steam games through Lutris.

Of course, the distinguishing features are not the common KDE and Linux tools bundled with the system but TUXEDO's own value-added contributions. The TUXEDO Control Center is a central user interface for managing the hardware environment (Figure 2), and it is one of the best examples of the close coordination between hardware and software. You get a quick dashboard view of CPU speed, system temperature, and fan usage. You can also choose a predefined

hardware profile to instantly tailor the system for different use cases. For instance, you can choose a profile for minimum power usage or one "mid-tier performance" profile for office usage. One profile that was close to my heart was the "Quiet" profile. As one who has never liked the noise of computer fans, I would have been delighted with this option in the past, but the fan on the TUXEDO laptop is so quiet that I hardly noticed it. You can also create your own custom hardware profile. The control panel also provides settings for controlling the keyboard backlight, the webcam, the shutdown timer, and hard disk encryption. Of course, there are other ways to do many of the things that the TUXEDO control panel does, but the interface is quite sensible and well designed, and keeping these settings in easy reach is certainly a benefit.

Another interesting feature built into TUXEDO OS is the Tomte configuration service [2]. The developers call Tomte "a service that automatically recognizes the customer's device and checks for missing drivers and required packages." The system accesses Tomte automatically during the install and at other important moments, but you can also interact manually with Tomte using the `tuxedo-tomte` terminal command.

Test Drive

The MK1 is a pleasure to work on. The keyboard is easy to use (North American

lefties, see the box entitled "Left Shift"). The battery offered up to 10 hours of run time, which means you could theoretically work all day in an airport without having to hunt for an outlet.

I was able to play videos easily and with no hitches. The system had everything I needed, and the picture looked great. The built-in speakers were functional for a movie or occasional YouTube clip, but if you listen to a lot of music tracks, you'll want headphones or external speakers.

The system also performed well for audio recording. I fired up Audacity and found that everything worked without any futzing or reconfiguration – not just for voice recording but for acoustic music on multiple tracks.

The clickpad caused me some confusion at first. It is a very large clickpad that is divided in two halves. You can either use the whole clickpad as one unit, with the left mouse-click on the left half and the right mouse-click on the right half, or you can disable one half and use the other for both the right and left mouse buttons.

Left Shift

When I first got the system, I started making more mistakes than usual when typing. It took me a while to notice the difference in the shift keys. The left shift key is noticeably smaller than the right; as a left-handed person who types on my computer all day, I missed the full-sized left shift. It turns out that the computer I reviewed came with an ISO keyboard, which is common for Europe but not so common where I live in the US. Right-handed people wouldn't notice the difference, but for lefties, the ISO keyboard can take some adjustment [3]. I got used to it after a while and stopped typing so many typos.

Luckily, if you're buying a new computer in Tuxedo's online shop, you can choose the ANSI or ISO keyboard option (Figure 3). Even if you're right handed, you might want to take a moment to consider which keyboard layout you are accustomed to and choose it, rather than floating with the default. The ISO keyboard is used more frequently in Europe and around the rest of the world because it is an international standard and is better for accommodating other languages. ANSI is an American standards organization, and the ANSI layout was created specifically for English.

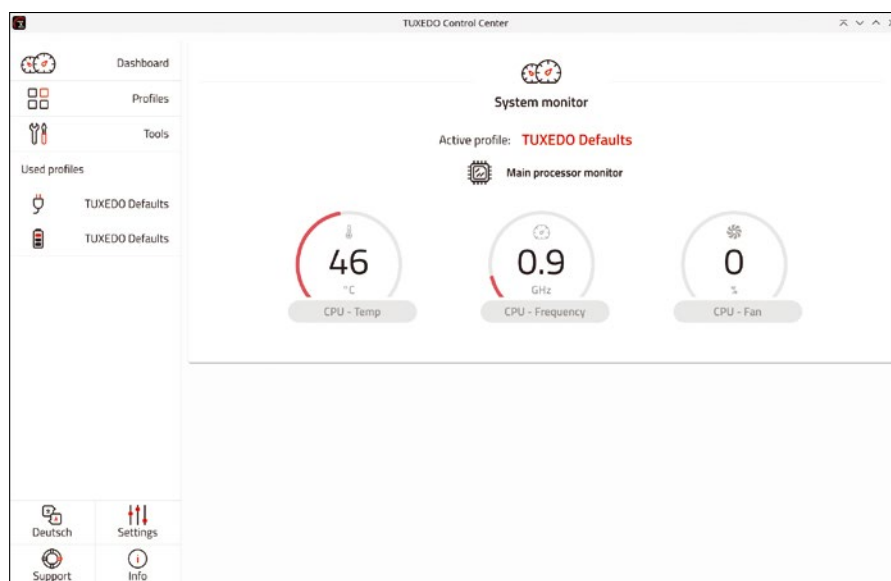


Figure 2: The TUXEDO Control Center lets you stay close to the hardware without leaving the desktop interface.

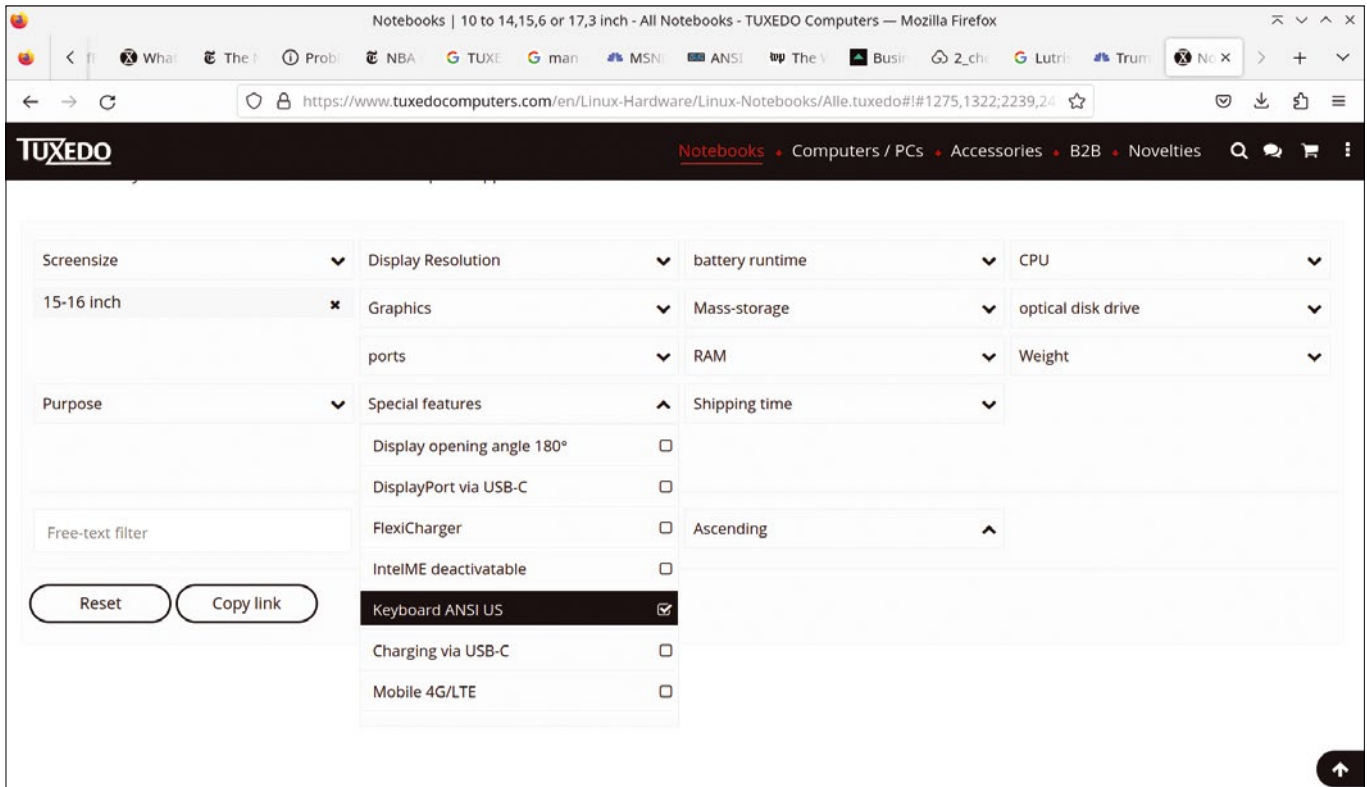


Figure 3: You can choose the ANSI or default ISO keyboard layout if you purchase online.

Once I got used to this feature, I quite liked it. There is no reason why the clickpad needs to be in the dead center of the keyboard – why not put it closer to the hand that uses it? That said, it did take me some time to figure it out. The controls for activating each half, or putting it in a mode where both halves work

together, were not well documented, and once in a while, the clickpad would appear to turn itself off unexpectedly. I eventually got it working the way I needed it to. (Tapping twice in the upper corner of the pad activates the half-pad mode.) Once I understood this, I could finally understand the cryptic symbols printed on

the top of the clickpad (Figure 4). Cool feature, but a worthy reminder that I was still in the Linux world, where innovation comes first and documentation often follows later. Buying a pre-built system saves you from most of the tinkering, but not all – you still have to tinker a little bit.

Conclusion

I'm a prototypical business user, and I need something that isn't a lot of trouble – light, convenient, easy to type on, and easy to plug things into. The TUXEDO laptop gave me everything I needed and more. ■■■

Info

- [1] TUXEDO's Infinity Pro 16 Gen7 MK1: <https://www.tuxedocomputers.com/en/TUXEDO-InfinityBook-Pro-16-Mk1-Gen7.tuxedo#>
- [2] Tomte Configuration Service: <https://www.tuxedocomputers.com/en/Infos/Help-Support/Frequently-asked-questions/What-is-TUXEDO-Tomte-tuxedo>
- [3] ANSI vs ISO Keyboard Layout: <https://switchandclick.com/ansi-vs-iso-layout/>

Author

Joe Casad is the editor in chief of *Linux Magazine*.



Figure 4: Once I figured out what was going on, the hieroglyphics on the clickpad made perfect sense.

The
NOV 12-17
INTERNATIONAL CONFERENCE for
HIGH PERFORMANCE
COMPUTING
NETWORKING, STORAGE, & ANALYSIS
[SC23.SUPERCOMPUTING.ORG](https://sc23.supercomputing.org)



#iamhpc

SC, the premier HPC conference, is a week of technical and professional events where thousands of researchers, engineers, technologists, educators, and students gather to learn, share, and grow.



+ EXPLORE the
POSSIBILITIES

Orient yourself with the four main components of SC and discover what the conference has to offer.

Program



Attend the leading technical program for HPC professionals and students, celebrated for its high-impact speakers, presentations, tutorials, panels, and more!

Exhibits



Engage the largest HPC expo and discover the latest technological innovations from industry, research, startup, and academic organizations, all under one roof.

Students



Find your HPC future! Students@SC provides special student-oriented programming to promote career success with a little help from mentors and friends.

SCinet



Experience the world's fastest network (for the week of SC) and learn how SCinet advances the frontiers to support all of the conference's networking demands.

Register on or before
October 12, 2023 and save!



REGISTER for
SC23 IN-PERSON IN DENVER
or THE DIGITAL EXPERIENCE



LEARN MORE & REGISTER!

SPONSORED BY



sighpc



TCHPC

The veteran of free distros

Trisquel



Rúben Rodríguez discusses Trisquel, a free Linux distro that has been in continuous development for the past 16 years. *By Bruce Byfield*

The Free Software Foundation (FSF) maintains a list of free distributions that contain no proprietary software [1]. Although DistroWatch lists 274 Linux distributions, the FSF lists only 10 free ones, and many of those are no longer developed. A notable exception, Trisquel has released a version every one to three years since 2007 [2] (Figure 1). Trisquel was founded by Rúben Rodríguez, who remains a leading developer for the distribution. Recently, Rodríguez took the time to discuss the distribution that has been such a large part of his life for the past 16 years.

Linux Magazine (LM): How did Trisquel get started? What were the original goals and was there a decisive moment in the project?

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/>.

Rúben Rodríguez (RR): The project started almost 20 years ago, when I was at the University of Vigo in Spain. My university wanted to make a customized GNU/Linux distribution, as it was quite trendy at the time, and I was approached along with other friends that were interested in that technology. I ended up

leading this project and eventually (when the trend passed and the university's interest faded) continuing it as an independent project.

When I first got involved in this field, most of my knowledge was technical, from experimentation and some classes and from hacking with my peers in the

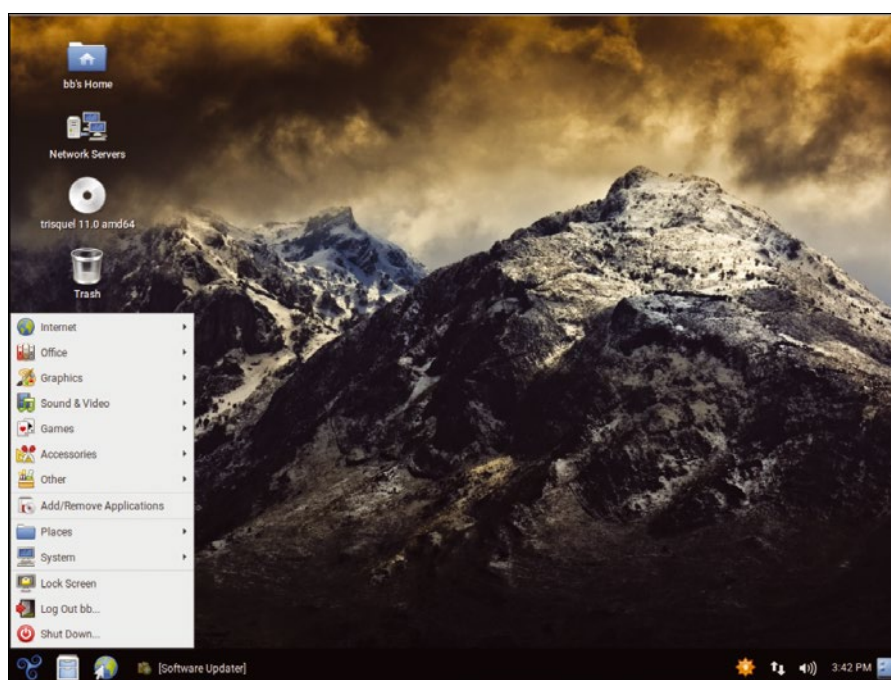


Figure 1: The Trisquel desktop.

Photo by Joshua Earle on Unsplash

local user group. It wasn't until a year after the project started, when my school hosted a talk by Richard Stallman, that I learned about the ideas of free software, licensing, and the GNU project. This changed my approach, as I recognized then that we should only provide and recommend software that could be used, studied, modified, and shared freely – more so in the context of a school project.

As a consequence, Trisquel moved from being a Debian derivative with some non-free packages added to a project based on Ubuntu with its proprietary packages removed. The choice was made because of the increasing popularity of Ubuntu (which was released roughly at that time), which pointed to a need for a version of Ubuntu without proprietary components, and because Ubuntu's fixed release cycle would facilitate our work.

LM: What changes to Ubuntu are needed to provide a completely free distribution?

RR: There are three main categories of work: removing non-free packages, removing non-free components from otherwise free packages, and rebranding for trademark compliance.

Removing a proprietary package is relatively simple. We do that through the ubuntu-purge scripts [3] (Figure 2). Currently, we have 150 source packages filtered out, which results in a larger number of binary packages. Adding a package to the list can sometimes require a detailed analysis to investigate licensing or other legal reasons and to make sure there is a free alternative when possible.

Removing proprietary components from a free package is more involved. It usually requires continuous maintenance to make sure the package remains functional, as well as to minimize the amount of work we need to do when the package is upgraded upstream. The largest of these kind of cases is Linux, for which we use the Linux-libre deblob scripts, which we incorporate into the Ubuntu kernel packaging workflow [4] (Figure 3). Another large example is Firefox, which, among many other customizations, is modified to not use *addons.mozilla.org* to search for extensions, as that repository contains both free and non-free software.

Firefox is also a good example of the rebranding task category. Some packages have a trademark license that has separate requirements from the software license itself. In some cases, it requires that the package must be rebranded if changes are made, as it happens with Mozilla products and Ubuntu itself. Our custom version of Firefox is called Abrowser.

LM: Besides the ethical choice, are there any advantages to a free distro?

RR: Free distros usually make better choices for their default settings in terms of privacy. For example, Abrowser is configured with a more strict privacy configuration than Firefox's default. Trisquel is also configured to minimize automatic (thus unrequested) network

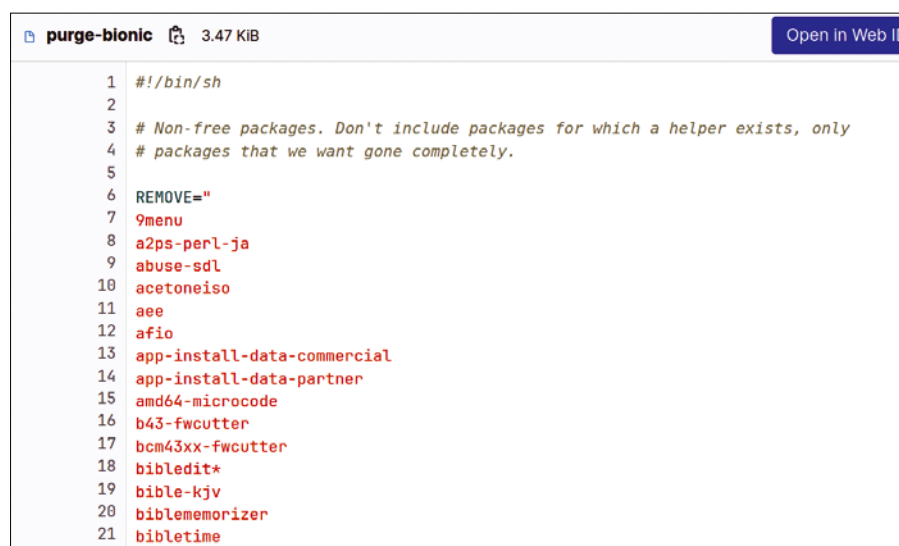
connections from different system components and opened network ports.

LM: Are there any disadvantages to a free distro?

RR: We try to make as many hardware devices work as possible, but some require proprietary drivers and will not work on Trisquel as-is. The user can always make the choice to install those components, but we leave that to them and don't recommend (or document) how to do it, as we find it more important to not advertise such software. We recommend resources like *h-node.org* to check for compatibility.

LM: How does Trisquel compare to other free distros listed by the FSF?

RR: All distributions on the list meet the FSF's Free System Distribution

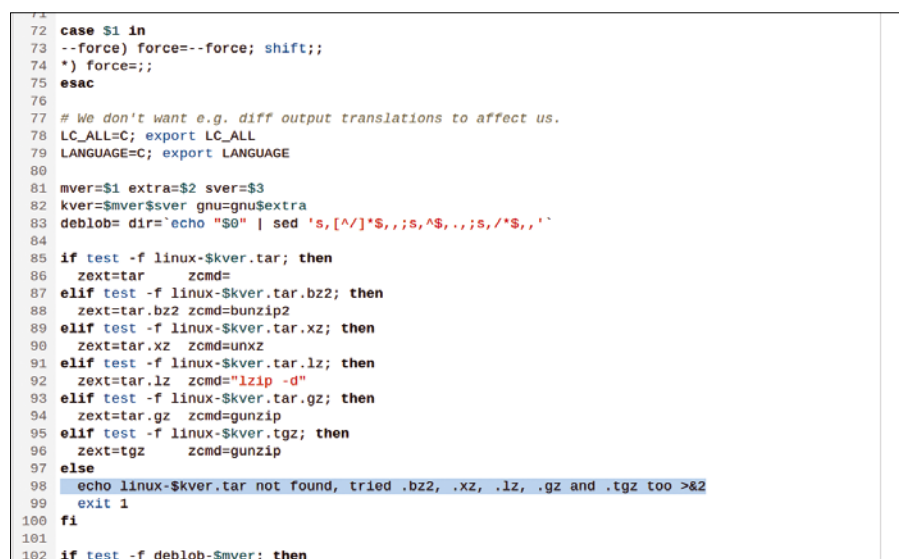


```

1  #!/bin/sh
2
3  # Non-free packages. Don't include packages for which a helper exists, only
4  # packages that we want gone completely.
5
6  REMOVE="
7  9menu
8  a2ps-perl-ja
9  abuse-sdl
10 acetoneiso
11 aee
12 afio
13 app-install-data-commercial
14 app-install-data-partner
15 amd64-microcode
16 b43-fwcutter
17 bcm43xx-fwcutter
18 bibleedit*
19 bible-kjv
20 biblememorizer
21 bibletime

```

Figure 2: Part of a ubuntu-purge script used to remove non-free elements from Ubuntu.



```

72 case $1 in
73   --force) force="--force; shift;;
74   *) force=;;
75   esac
76
77 # We don't want e.g. diff output translations to affect us.
78 LC_ALL=C; export LC_ALL
79 LANGUAGE=C; export LANGUAGE
80
81 mver=$1 extra=$2 sver=$3
82 kver=$mver$sver gnu=gnu$extra
83 deblob= dir="echo "$0" | sed 's,[^/]*$,,;s,^$,.,;s,/^$,,'
84
85 if test -f linux-$sver.tar; then
86   zext=tar zcmd=
87 elif test -f linux-$sver.tar.bz2; then
88   zext=tar.bz2 zcmd=bunzip2
89 elif test -f linux-$sver.tar.xz; then
90   zext=tar.xz zcmd=unxz
91 elif test -f linux-$sver.tar.lz; then
92   zext=tar.lz zcmd="lzip -d"
93 elif test -f linux-$sver.tar.gz; then
94   zext=tar.gz zcmd=gunzip
95 elif test -f linux-$sver.tgz; then
96   zext=tgz zcmd=gunzip
97 else
98   echo linux-$sver.tar not found, tried .bz2, .xz, .lz, .gz and .tgz too >&
99   exit 1
100 fi
101
102 if test -f deblob-$sver; then

```

Figure 3: The start of the script used to remove proprietary blobs from the Linux kernel.

Guidelines (FSDG) [5]. This is different from the FSF’s Respects Your Freedom (RYF) list, which is a program for the certification of hardware products – that is, devices that don’t require non-free components such as proprietary firmware or drivers to work [6]. The FSDG is a set of recommendations given by the GNU project to produce distributions of GNU that follow the principles of the project. Distributions that follow the guideline are recommended by the FSF and GNU and are used (and sometimes produced) by vendors that are RYF certified, so there is an overlap.

The distributions recommended by the FSF and GNU are mostly listed from their compliance with the FSDG, so it includes a variety of projects with different practical goals. Some are more experimental or hacker-oriented, while some are more specialized (like distributions for embedded devices). Trisquel aims to be very user-friendly, focusing on having strong accessibility features out of the box, a simple design, and good language support. Having a regular release cycle, Trisquel also aims to be a good choice for server infrastructure.

LM: Are free distros becoming more or less popular?

RR: I don’t have specific statistics, as we don’t track our users. Our server traffic indicates a steady interest in our project, and we have a quite active community participating in forums, mailing lists, and IRC.

LM: Describe the work involved in becoming FSDG-compliant.

RR: In addition to the cleanup of non-free components I mentioned, there are other requirements like not recommending proprietary software in the documentation and on community communication channels like forums and such. An important requirement is self-hosting; it is not enough to make a free installer (like a live image) for a distro that has non-free components in their repositories. Mainly, it focuses on the responsibility that comes from distributing software to others.

LM: Trisquel supports several desktops, such as MATE, LXDE, and Sugar, the desktop originally designed for the

educational purposes by One Laptop per Child. Do any of the supported desktops involve special work?

RR: All of the desktops that we distribute as a live environment are quite polished, but the MATE desktop (our recommended default) is the one that takes the most work. MATE is a very good choice for accessibility, due to its good compatibility with screen readers and with keyboard navigation. Our target is that the system must be installable without help for people with disabilities including blindness. It is also a good choice for computers that have no 3D acceleration (which in many cases can only be achieved with non-free drivers), as it is an optional requirement for that desktop.

LM: What other features make Trisquel stand out?

RR: I’ve mentioned most of the practical advantages, as many stem from the project values in the first place. A goal that goes hand-in-hand with accessibility is the simple design and the ease of use for which we aim.

One thing that is remarkable is how easy it is to contribute improvements. We use quite simple Bash scripts (we call them package-helpers) to apply modifications to upstream packages, which get built and published for testing automatically through our CI infrastructure and then published for production after review.

LM: Do you have any stats on downloads, commits, and developers?

RR: We only track downloads partially, since there are many mirrors that we don’t get statistics from. Our main site has counted over a million downloads. Our most active repository is the previously mentioned package-helpers, with over 2,000 commits affecting 300+ source packages. That project has gotten contributions from 25 developers. We also have volunteers helping with the documentation, translations, build systems, and other sub-projects.

LM: How is Trisquel governed? How are decisions made?

RR: Trisquel is established as a non-profit organization registered in Spain, which takes care of the legal and tax required obligations. The project is quite

horizontal in its structure, and we make most decisions during the developers’ meetings I mentioned. In terms of community management, we are lucky to have a good amount of long-term participants in our forums and lists that are excellent at encouraging newcomers, keeping the conversation on track, and notifying moderators the few times intervention is needed.

LM: Who is the target audience for Trisquel?

RR: We try to make it as wide as possible. Trisquel is not a specialized distro. It is not particularly hacker-oriented, but you can dig into the code and technical details to any depth, so it can be used by people of very different levels of expertise. It targets both desktop and server users, and recently we added ARM and IBM POWER architecture support to expand on the possible use cases. Ultimately we want it to be a very flexible platform for people that are walking up the “freedom ladder” and are exploring and learning about free software.

LM: Are there any future plans?

RR: Our main goals at the moment are to find more developers and other contributors, to renew our main website, and to continue to grow our build infrastructure.

LM: Is there anything else readers should know about Trisquel?

RR: I invite people to try it out, check out our community, and explore the many projects we work on and that always need more contributors. Also, Trisquel is sustained 100 percent from donations from our community [7]. ■■■

Info

- [1] FSF free distributions: <https://www.gnu.org/distros/free-distros.html>
- [2] Trisquel: <https://trisquel.info/>
- [3] ubuntu-purge: <https://gitlab.trisquel.org/trisquel/ubuntu-purge>
- [4] Linux-libre: <https://www.fsfla.org/ikiwiki/selibre/linux-libre/>
- [5] FSDG: <https://www.gnu.org/distros/free-system-distribution-guidelines.en.html>
- [6] RYF: <https://ryf.fsf.org/>
- [7] Donations: trisquel.info/donate



DrupalCon

LILLE 2023
17-20 OCTOBER

Looking forward to seeing you in Lille!

DrupalCon comes back to France in 2023 between 17-20 October! This time the Lille Grand Palais is our host venue in Lille. Easily accessible with only a 35-minute train ride from Brussels and 80 minutes from London by train.



Register now to
get the best rate!



Take a look
at the Schedule



Become
a Sponsor

Visit our website <https://events.drupal.org/lille2023> for more information





A modern terminal pager

New Kid on the Block

The most terminal pager offers a feature-rich, better organized alternative to less. *By Bruce Byfield*

Pagers are commands for viewing files one page at a time in the terminal. Roughly speaking, pagers serve the same function at the command line as Plasma's Okular, although pagers are usually used for text files or as a pipe to make screen output from other commands more readable without scrolling. At times, a

version of cat can be used for the same purpose, but technically, the pager name is reserved for three commands: less, more, and – most recently – most [1], names that are a sequence of puns, less being more, as the popular saying goes, and more being greater than less. As the newest pager, most is by far the most usable (Figure 1), because the

alternatives are limited or awkward to use as soon as you go beyond the bare command.

The trouble with cat is that, strictly speaking, it is not a pager at all. Its main function is to concatenate or join files together. It can function as a pager, especially for a short file, simply by specifying a single file, but cat cannot scroll or do more than number lines and display non-printing characters. A cat clone, bat, offers all the functions of cat, plus backwards scrolling, range, and the ability to

```
One Stop Distro Solutions
By traditional definition, distributions are software. Yet increasingly, companies are offering distributions as part of a bundle that is
The advantages of one stop solutions are obvious. Vendors can offer software tailored to their hardware, and win customer loyalty. For$
Whatever the reason, one-stop solutions have only started to come into their own in the last few years. Many are derivatives of Ubuntu$
Tuxedo Computers
Tuxedo is a rapidly expanding company that is just starting to become well-known in North America. While offering an extensive product$
Purism
Purism began in 2014, and immediately made headlines when its PureOS became one of the few distributions to win the Free Software Fou$
A disturbing note is that both the hardware and distribution appear not to have been updated for at least a year, perhaps closer to tw$
MX Linux
MX Linux has had the most page views on Distrowatch since 2019. Partly, its popularity may be due to the fact that is is a collaborati$
Starlabs Systems
Starlab's website does not allow for easy browsing, but does have the option of displaying prices in dozens of currencies. Alone among$
-- MOST: one-stop-distros.txt (1,1) 0%
Top of buffer.
```

Figure 1: Displaying a text file with most.

display in a pager with `--pager PAGER`. However, `bat` is at best a basic pager.

As for `more`, it is 45 years old. Its age shows in the fact that some versions of `more` do not support backwards scrolling, even today, as well as in the continued existence of `--print-over (-p)`, which instead of scrolling, clears each screen as the next one is printing, and of `--plain (-u)`, which would suppress underlining, except that today it is left unenabled. The best that can be said for `more` is that it has a handful more options than `bat`.

Such limitations leave `most` users with `less`. While highly functional, `less` sometimes seems an over-reaction to `more` with `less`'s dozens of options, including those for search and navigation. In general, `less` is a solid foundation, which is why it continues to be the standard pager, despite `most`'s longer feature list.

Making the Most of most

By taking advantage of today's readily available memory, `most` offers features that `less` does not. In particular, if compiled that way, `most` can open multiple files of at least two lines, a feature that is ideal for file comparison. Multiple files are opened by listing them in a space-separated list after the command and any options:

```
most OPTIONS FILE1 FILE2
```

The name of the file is displayed at the bottom of a window in the status line. Below the status line is any current activity or a set of basic instructions (Figure 2).

Within each window, commands can be given simply by typing the letter, without the one- or two-hyphen prefix required by an option when starting `most`. Note, though, that which option is used for each command depends on how `most` is compiled, so if one option listed in the man page fails to work, another may be successful (the options covered here are the ones that worked for me, sometimes after trial and error). Each window can be scrolled independently, or synced with other windows using `l` or `L`, which adds an asterisk (*) to the status line. By default, `most` does not wrap lines; long lines are indicated by a dollar sign (\$) in the right margin of the window. You can see the rest of the line by scrolling with the right arrow key or by expanding the

width of the terminal window. However, adding the `-w` option to the command wraps lines, marking the right margin with a back slash (\). Conveniently, by adding the `-e` option to the command, you can edit the current window, which means that `most` can also act as a text editor. To move to another window, enter `n` and the name of the next file entered in the command appears at the bottom of the window; you can access that file by pressing the Enter key. However, you might prefer instead to split the current window with `Ctrl + X + 2`.

Like `less`, `most` has an extensive list of keyboard shortcuts for navigation. The `most` man page lists several dozen, many of which have two or even three alternatives. Mercifully, though, within a window, you do not need to learn all of the shortcuts to navigate efficiently. That said, `t` and `b` are handy for moving to the top and bottom of the window. So are `NUMBER J` to jump to a specific line number or `NUMBER %` to jump to a percentage of the way through a file. In a long file, you might also opt for `NUMBER U` to jump back a set number of screens. More simply, you can just use the Up and Down arrow keys to scroll, either by the default single line or by your choice of lines by entering a number before you press the arrow key.

A set of search functions are also available with `most`. When starting `most`, you can add `-c` to make searches case sensitive. Within `most`, `f` prompts for a string for which to search forward. Similarly, a question mark (?) prompts for a string for which to search backwards. With both `f` and `?`, `Ctrl + G` cancels the search. Once you have located a string, `m` sets a bookmark. From there, `Ctrl + X` then `Ctrl + x` (with the same keys used twice) can return you to another bookmark, so you can toggle back and forth, a handy feature as you edit.

Simplifying Learning

Another advantage of `most` over `less` is that it is better organized than `less`. For one thing, `most` has far fewer options to add to the command, and most of those turn features on or off that the majority of users are unlikely to need. Although `most`'s man page is still unusually long, it is shorter and far better organized than that for `less`.

```
- MOST: one-stop-distros.txt
Next File (0): coreboot-firmware.txt
```

Figure 2: At the bottom of the current main window, you see that `most` is about to switch to another open file.

The same is true of `most`'s environmental variables. There are fewer of them than in `less`, and several are designed to minimize the need to memorize options and commands. `MOST_SWITCHES` sets options, one per line. For example,

```
define MOST_SWITCHES "-s"
```

stops the display of more than one blank line in a row. `MOST_INITFILE` is even more convenient because it defines a configuration file, looking for it in `/etc/most.conf` or in `.mostrc` in the current account's home directory. No matter where it is found, this configuration sets the default keybindings, using for each the format

```
setkey "FUNCTION" "KEYBOARD SHORTCUT"
```

Taking the time to create a configuration file can save regular `most` users the need for memorization, as well as involved typing. `MOST_HELP` can also be used to set a custom help file with details not mentioned in the default one. As well, those who edit in `most` may also want to set their editor of choice in `MOST_EDITOR`. The advantage of `most` lies not only in features and organization, but in these time-saving devices. Like templates in a word processor, these environmental variables take time to set up, but in the end can bring users quickly up to speed and save time. ■■■

Info

[1] `most`: <https://linux.die.net/man/1/most>

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/>.



Designing cross-platform GUI apps with Fyne

Platform Agnostic

The Fyne toolkit offers a simple way to build native apps that work across multiple platforms. We show you how to build a to-do list app to demonstrate Fyne's power. *By Andrew Williams*

One of the biggest challenges in modern app development is the requirement to develop a separate codebase for each platform that you want to support. Fortunately, the Fyne graphical user interface (GUI) toolkit lets you program your app once and then deploy it rapidly to multiple devices.

Fyne is an open source GUI toolkit built for the Go programming language, a popular general-purpose compiled programming language created by Google a little over 10 years ago. Go is frequently used for cloud services and infrastructure projects. However, the Fyne toolkit makes it possible to quickly build native GUIs for desktop and mobile devices. One of a number of GUI toolkits for Go, Fyne aims to be the simplest to use, appealing to both beginners and experienced GUI developers.

Fyne's idiomatic design makes it easy for Go developers to rapidly learn its principles. Fyne allows developers to create their own custom widgets thanks

to its rich widget set and extensibility. With graphical elements inspired by Material Design, Fyne will feel familiar to many end users, while remaining open to theming by app developers. Fyne is fully unit tested and supports rigorous test-driven development. Released under the same permissive BSD licence as Go, Fyne is supported by an active development community.

In this article, I will show you how to set up your system to program your first app with Go and Fyne. I will cover the basic APIs for showing windows, laying out content, and using the standard widgets, along with storing data and working with event handlers. By the end of this article, you will have a complete graphical application running on your computer and smartphone device.

Setup

To get started with Fyne, you need to install a few developer tools including Git, Go, and a C compiler. Assuming you are

working on a Linux desktop with an X11 setup, you also need to install some libraries. Don't worry – this only applies to developing Fyne apps. Your users won't need any of these dependencies to run the finished apps. Fyne (like Go) focuses on single binary compiler output with zero dependencies. As long as your users have a working graphics driver, your apps will work without any additional setup.

You should use your computer's package manager to install the necessary tools. All distributions are different, but the most common systems will need to use the following commands. For Debian and Ubuntu, use:

```
sudo apt-get install golang gcc \
libgll-mesa-dev xorg-dev
```

On Fedora, use:

```
sudo dnf install golang gcc \
libXcursor-devel libXrandr-devel
```

```
mesa-libGL-devel libXi-devel
libXinerama-devel libXxf86vm-devel
```

For Arch Linux, use:

```
sudo pacman -S go xorg-server-devel
libxcursor libxrandr libxinerama libxi
```

You also need at least Go v1.14 for Fyne code to compile. You can adapt the above commands for your package manager or find the full list in the Fyne documentation [1].

Getting Started

Once you've installed the necessary tools, you are now ready to build a basic Go app to make sure that everything is working. First, make a new folder and set it up as a Go project using `go mod init` (Listing 1).

You now have a folder that contains a `go.mod` file. You can test to see if the Go compiler is working by quickly sending some basic file content to `main.go` and then building the app (Listing 2). All you need is the package name (`main`) and a simple `main()` function to let the compiler know you are building an application instead of a library.

If your build completes without error, then it has succeeded. You will see a file called `todoapp`. You can run `todoapp` at this stage, but you will not get output because you have not built your app yet!

Showing a Window

I'll now demonstrate how to write some code to display a window onscreen.

First, you need to install the Fyne library into your project with `go get`. The following command will download the Fyne project's source code from GitHub and store it in the Go cache. It then marks Fyne as a dependency of your project so that your code editor or IDE

Listing 1: Create a Project

```
> mkdir todoapp
> cd todoapp
> go mod init todoapp
go: creating new go.mod: module todoapp
```

Listing 2: Test the Install

```
> echo "package main
func main() {}" > main.go
> go build
```

can make sensible suggestions for auto-completion and compilation:

```
> go get fyne.io/fyne/v2@latest
go: added fyne.io/fyne/v2 v2.3.4
```

After installing the Fyne library, you next must create a new application instance, which is the variable that will be used to create new windows, access storage, and use other important features. You can create a new app by inserting the following into your main `func`:

```
a := app.New()
```

However, it is best practice to identify your app, which can be done using:

```
app.NewWithID("com.example.myapp")
```

The code should be added to the top of your main function. This app package is from the import path `fyne.io/fyne/v2/app` (your IDE will probably automatically add this).

In the new app, you can create a new window by simply requesting it with:

```
w := app.NewWindow("TODO")
```

This sets up the variable `w` for the new window. The window will have the title "TODO" which may be displayed in your window border. At this point, you could build the app. However, without any content, the window would be fairly useless, so I'll show you how to add a widget.

Adding a Widget

In Fyne, an app can be created by drawing basic types (called `CanvasObjects`) to a position within the window. These items are available in the `fyne.io/fyne/v2/canvas` package. However, you can use a collection of pre-made widgets (`fyne.io/fyne/v2/widget`) that provides several interactive widgets for the standard user controls and visual items found in most apps.

You can create a new label widget to display the text "TODO App" by simply calling `widget.NewLabel("TODO App")`. Then use the `SetContent` method to place this widget inside the previously created window. The line of code now becomes:

```
w.SetContent(
    widget.NewLabel("TODO App"))
```

Finally, you must show the window. Because GUI toolkits also have an event loop that manages how the app runs and listens for user input, you also need to run the app. Conveniently, Fyne has a method that simultaneously shows the window and runs the application:

```
w.ShowAndRun()
```

That is all there is to creating your first graphical app.

Running the App

To run the app, you can use `go run`, or you could build it using `go build` and then later run the executable that was created. However, because you have added a new library, you should run `go mod tidy` first to ensure the project metadata is up-to-date. After this, you can run the app with:

```
> go mod tidy
go: finding module for package
fyne.io/fyne/v2/app
go: found fyne.io/fyne/v2/app in
fyne.io/fyne/v2 v2.3.4
> go run .
```

Telling the Go compiler to run "." means it runs all the files in the current directory (`main.go` for your app). Your app should now look something like Figure 1. Depending on your desktop settings, the app may appear with a light or a dark theme.

Building the User Interface

Before you start building more complex content for your app, I need to explain `Container` types. Containers are what allow multiple elements to be arranged in an area, and they do so according to a layout. There are many standard containers for common types of positioning in Fyne. You can also

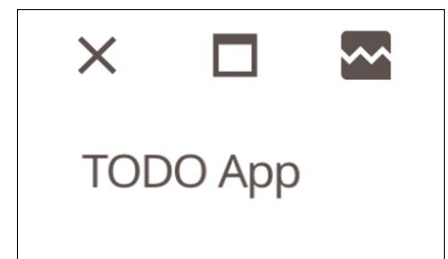


Figure 1: The app is shown here using a light theme, but a dark theme is also possible depending on your desktop settings.

add your own layout code. (I won't cover that in this article, but you can find out more online [2].)

These container types include `VBox` and `HBox` (for packing child items into vertical or horizontal boxes), `Split` (to allow users to drag a split bar separating two child items), or `Stack` (to allow multiple items to be drawn on top of each other). The most popular, and flexible, container is `Border`, which you will use twice in the to-do list app. The `Border` container takes four required fields for items to be positioned along the top, bottom, left, and right edges, and an optional fifth parameter for the content that should fill the remaining (central) space.

The app is split into two main areas: the top bar (where you will position the "New item" entry widget) and the to-do list area, which fills the main area. In essence, the app's entire layout is captured as:

```
container.NewBorder(head, nil, nil, ?
nil, list)
```

The app uses a new package called *container*, which is from `fyne.io/fyne/v2/container`.

To make the top bar, you will use another `Border` container, which will have an `Entry` widget for the user input (Listing 3) and a new `Button` widget using an icon for adding content, which is available in the standard theme. (I will add code to respond to user input later.)

Listing 3: Input Widgets

```
01 input := widget.NewEntry()
02 input.SetPlaceholder("New item")
03 add := widget.NewButtonWithIcon("",
04 theme.ContentAddIcon(), func() {
05 log.Println("new item tapped")
06 })
07 head := container.NewBorder(nil, nil, nil, add, input)
```

Listing 3 defines the app's user interface except for the to-do list in main area.

Displaying a Data List

To complete the user interface, you need to define the main to-do list. The `List` widget is ideal for this task, but it is a little more complex than the previously discussed widgets. `List` is a "collection widget" (along with `Table` and `Tree`) designed to handle lots of data at high performance.

I will create a list using `widget.NewList`. Rather than passing static data, `widget.NewList` uses callbacks to understand how content should be displayed. The three callbacks it uses define the number of list items, how an item is created, and what data each row should contain. This more complex setup means that Fyne can reuse widgets as a user scrolls, which is much faster and more efficient for CPU and memory.

For now, I will use dummy content in the sample to-do list. It will

display five items, and each one will show "TODO Item" (as the data for the third callback doesn't yet exist to display). For each row in `List`, I use a `Check` widget to give each item a checkbox, which will allow users to mark completed items and display the associated text. This code, together with the earlier items, comprises the entire user interface. For simplicity, the graphical setup code is moved into a new function called `loadUI` as shown in Listing 4.

Launching the GUI

Because the code is now in the `loadUI` function, you also need to update the `main` function to set the content to be the result of calling `loadUI`.

A list can be very small, but you want to see multiple items at the same time. To do this, you call `Resize` on `Window` with a suitable size (not pixels because a

Listing 4: ui.go

```
01 package main
02
03 import (
04     "log"
05
06     "fyne.io/fyne/v2"
07     "fyne.io/fyne/v2/container"
08     "fyne.io/fyne/v2/theme"
09     "fyne.io/fyne/v2/widget"
10 )
11
12 func loadUI() fyne.CanvasObject {
13     list := widget.NewList(
14         func() int {
15             return 5
16         },
17         func() fyne.CanvasObject {
18             return widget.NewCheck("TODO Item",
19                 func(bool) {}))
20         },
21         func(id widget.ListItemID, o fyne.CanvasObject) {
22             })
23
24     input := widget.NewEntry()
25     input.SetPlaceholder("New item")
26     add := widget.NewButtonWithIcon("",
27         theme.ContentAddIcon(), func() {
28             log.Println("new item tapped")
29         })
30     head := container.NewBorder(nil, nil, nil, add, input)
31
32     return container.NewBorder(head, nil, nil, nil, list)
33 }
```

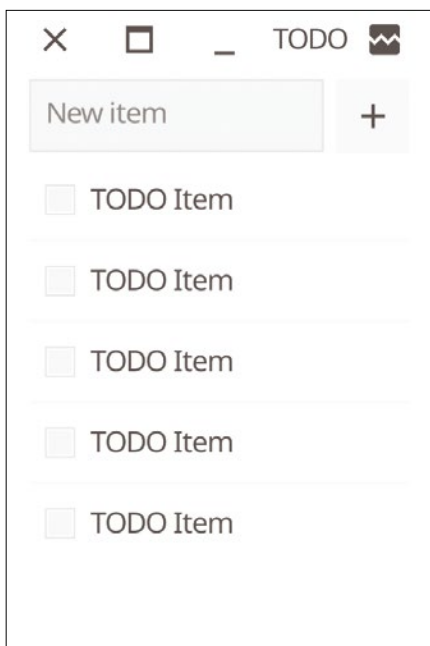


Figure 2: The full graphical layout for the to-do list app.

Fyne size will be the same across different output devices). The last three lines of the `main` function become:

```
w.SetContent(loadUI())
w.Resize(fyne.NewSize(200, 280))
w.ShowAndRun()
```

When you run the updated code, you will see the new user interface appear on your screen (Figure 2).

Listing 5: Storing Data

```
var (
    todos []string
    list *widget.List
)
```

Listing 6: Adding a To-Do Item via Tapping

```
add := widget.NewButtonWithIcon("",
    theme.ContentAddIcon(),
    func() {
        addTODO(input.Text)
        input.SetText("")
    })
```

Listing 7: Adding a To-Do Item via the Keyboard

```
input.OnSubmitted = func(item string) {
    addTODO(item)
    input.SetText("")
}
```

Listing 8: data.go

```
01 package main
02
03 import "fyne.io/fyne/v2"
04
05 func addTODO(todo string) {
06     todos = append(todos, todo)
07     list.Refresh()
08 }
09
10 func deleteTODO(todo string) {
11     for i, text := range todos {
12         if text != todo {
13             continue
14         }
15     }
16     todos = append(todos[:i],
17         todos[i+1:]...)
18     break
19 }
20 list.Refresh()
21 }
```

Remembering To-Dos

You want to be able to add items and mark them as done in your app. To do this, you will use a new global variable, `todos`, which is a slice of strings (denoted as `[]string`). You will manipulate this slice when adding or deleting items, and your list will always draw this data's current state. First, you add a new `todos` variable to `main.go`. You also need to keep a reference to the `list` variable created earlier so it can be updated (shown in Listing 5).

Adding Items

When the user taps on the add button created in Listing 3, a new item will be added to the to-do list. To do this, you make a more useful tapped handler for the button. Instead of logging to the console, you will use the code in Listing 6, which asks for a new item to be added and then clears the input text to get ready for another to-do item.

Because some users prefer to run an app from the keyboard, you will also want this functionality when the user hits Return or Enter. Thankfully, the Entry widget has an `OnSubmitted` callback that you can set as shown in Listing 7.

Both handlers call a new `addTODO` function, which simply appends a new item to the slice and refreshes the list (see Listing 8).

Populating the List

Now that you have some data to display, you need to tell the list how to update accordingly. To do this, you add some code to the third callback of the List

Listing 9: Displaying To-Do Content

```
01 func(id widget.ListItemID,
02     o fyne.CanvasObject) {
03     check := o.(*widget.Check)
04     check.SetChecked(false)
05     check.Text = todos[id]
06     check.Refresh()
07 }
```

Listing 10: Deleting To-Do Items

```
check.OnChanged = func(done bool) {
    if !done {
        return
    }

    deleteTODO(check.Text)
}
```

widget. This code will set the Check widget's text and make sure that it is not checked (because the list currently only has pending to-do items). Listing 9 achieves this using a replacement callback function.

Line 1 of Listing 9 makes sure you can use the Check widget's functionality. The list does not know what type of content it displays, so you need to do a type cast to match the widget you created in the second callback function. With this in place, you will see your to-do list – all that is left is to delete them!

Deleting Items

The to-do list is only keeping track of pending items, so you want to remove any to-do item once it is checked. For this, you can set the `OnChanged` callback to your Check widget (Listing 10), much like you set `OnSubmitted` on the Entry widget.

The `OnChanged` callback will be called if the item is checked or unchecked – which is why you first test that it is being marked as completed using the `bool` parameter. Assuming it should be deleted, you call the helper function `deleteTODO` from Listing 8.

The method shown in Listing 8 is a little complicated because there is no built-in way to delete an item from a slice in Go. Listing 8 performs “re-slicing,” which sets the to-do list to a join of two other lists (a list for before and one for after the item is deleted). The code then refreshes

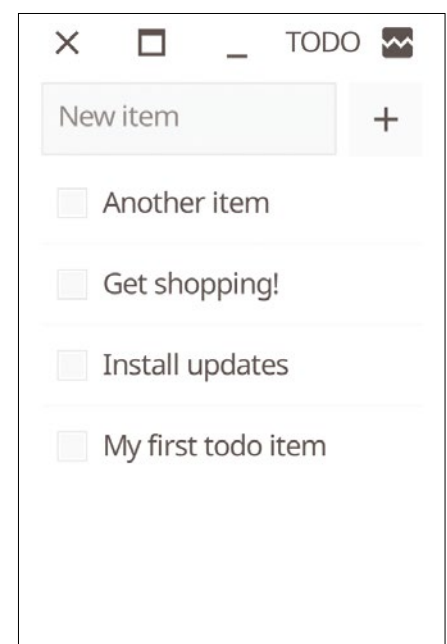


Figure 3: You can now input your to-do items.

the list as before, so your item disappears. If you run your app once more, you will see that it is empty, but you can add new items and mark them as done as often as you like, as shown in Figure 3.

Storing Data

You now have a fully functional app to track your to-do items, but the app will forget the items when you exit. To remedy this, you can connect to a database or store data in a file. However, Fyne lets you store your data in Preferences, a simple key-value store used to easily keep track of user interface state and small data storage without worrying about file storage or connection handling. You can access this feature using the Preferences() method on the App type created earlier.

Loading To-Do Items

You need to ensure that any existing to-do items are loaded when the app starts. To do this, you can simply set up the todos variable to be the result of a load call. Immediately before calling loadUI, you need to call

```
list = loadTODOs(a.Preferences())
```

This call will pass your application preferences to a new loadTODOs function that will return a slice of all stored to-dos.

Saving the Data

Before you have data to parse, you need to save it! To do this, you will use

Listing 11: storage.go

```
01 package main
02
03 import "fyne.io/fyne/v2"
04
05 const joiner = "|"
06
07 func loadTODOs(p fyne.Preferences) []string {
08     all := p.String("items")
09     if all == "" {
10         return []string{}
11     }
12     return strings.Split(all, joiner)
13 }
14
15 func saveTODOs(items []string, p fyne.Preferences) {
16     allItems := strings.Join(items, joiner)
17     p.SetString("items", allItems)
18 }
```

another new method, saveTODOs, which will take the current slice to save and a reference to the preferences you will be using:

```
saveTODOs(todos, p)
```

Be sure to call this method inside deleteTODO and addTODO after updating the data.

The methods that actually do the work in this example are really simple – you just concatenate all the items to a single string using a joiner constant and then split the items out on load, as shown in Listing 11.

Local Storage or Cloud

Regardless of your operating system or device, Listing 11 will store the data for you. However, many apps now want to have centrally stored information on a cloud service. The great news is that Fyne supports this too! Using exactly the same code from Listing 11, you can have a cloud back end for your data, which is set up by simply calling SetCloudProvider for your app with a suitable provider description. You can find more information about setting this up in a recent Fyne-Conf video [3].

Packaging and Installing

Once you have a completed your app, the next step is make it more readily available either for yourself or others.

First, you need to add an icon before installing or sharing the app.

Although Fyne is a vector graphics-based toolkit, you will need a bitmap image for the app icon. For greatest compatibility, create a 1024x1024 PNG file, name it Icon.png, and save it to the project folder. Once created, this will be used as the app icon. The best way to bundle apps is to use the fyne command-line tool:

```
> go install fyne.io/fyne/v2/cmd/fyne@latest
```

Once installed you can call fyne package to set up an app bundle. It will use the appropriate format for the current operating system. You can also install it at the same time using fyne install. This will package the app and put it into your standard app location (/usr/local/bin for most Linux distributions).

If you get a “command not found” message, be sure to check that the Go binary location (usually ~/go/bin) is in your \$PATH. Once complete, the app will be available in your favorite app launcher (Figure 4).

Sharing with Others

Before sharing your app, you should consider adding a tutorial for new users. This is really easy for a to-do list app because you can just add items to the UI when the app has not been used before by simply replacing the call to Properties.String with Properties.StringWithFallback. The fallback case will be used when the requested preference key has not previously been saved. Try the following:

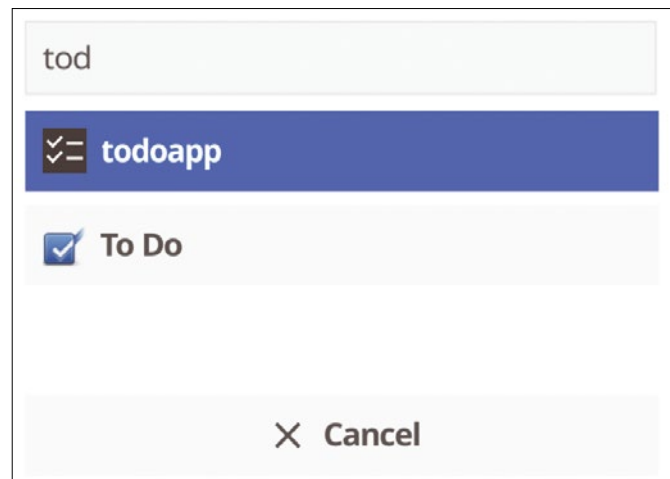


Figure 4: You can launch the to-do app from your favorite launcher.

```
p.StringWithFallback("items",
    "Do this item"+joiner+"Learn Fyne!"
    +joiner+"Build an app")
```

Now, you can package the code for distribution. The command `fyne package` will prepare a bundle (for a Linux host, this will create `todoapp.tar.xz`). The bundle will contain your binary, an icon file, and the metadata needed to install on other computers. These binaries are relocatable, so it will work for computers with a different configuration. New app users will see the welcome screen shown in Figure 5 with a tutorial and an icon!

Building for Other Platforms

Another great feature of the `fyne` command-line tool is that it can package code for other platforms, just like the Go compiler. You can specify the operating systems that you want to bundle the app for, one at a time. For example:

```
> fyne package -os windows
```

Due to the graphical libraries required to compile (like the ones you installed at the beginning of the article), there are some additional requirements for cross-platform compilation. See the Fyne documentation for more information [4].

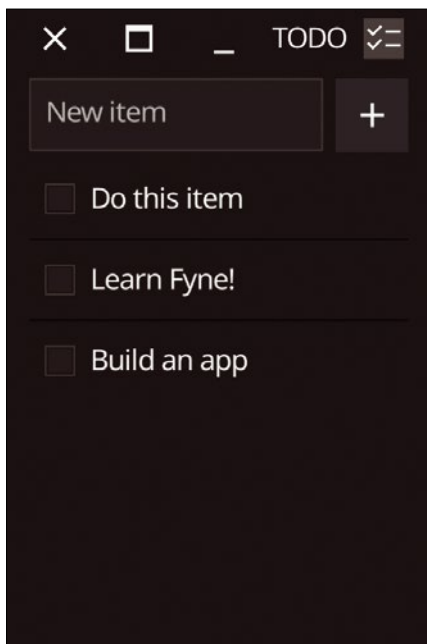


Figure 5: The welcome screen now contains a tutorial and an app icon.

If you are familiar with a Docker or podman install, then you can use the `fyne-cross` [5] tool, which will handle all the complex developer setup process inside containers. As long as you have a container engine running, it should be as simple as:


```
> go install 
github.com/fyne-io/fyne-cross@latest
> fyne-cross android
```

Figure 6 shows how the same app will look when running on an iOS or Android mobile device. The package was created using `fyne-cross` and installed using standard developer tools.

Finally, if you are interested in automating your build for multiple platforms along with having the downloads hosted for you, check out build platform tools such as Geoffrey by Fyne Labs [6].

Conclusion

In less than 100 lines of code, you now have a graphical application that will

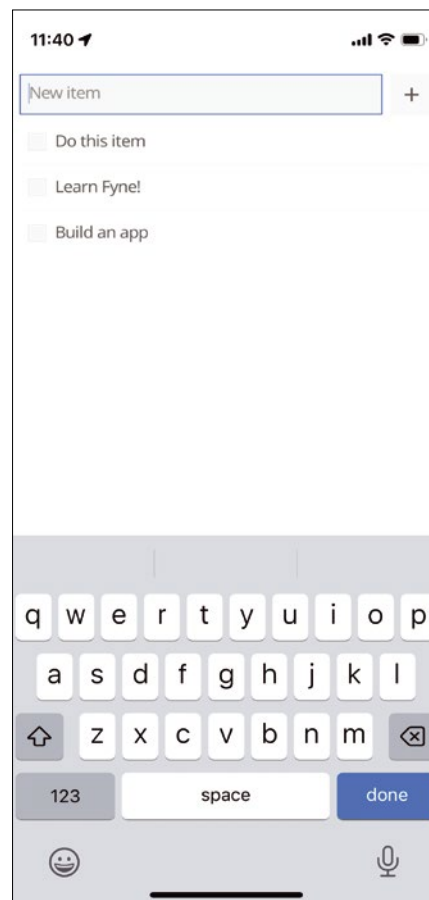


Figure 6: The same app code shown here running on iOS.

work on the desktop or mobile device of your choice. You can find the full application source code on the project's GitHub page [7]. Development with Fyne is easy to get started with. I hope you will be inspired to explore the potential of this GUI toolkit further.

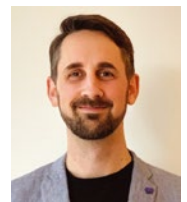
Visit the Fyne apps website [8] to see a list of other applications that have been developed using Fyne. To find out more about programming with Fyne, check out the Fyne documentation and tutorials [9]. As a Linux enthusiast, you may also be interested in trying out the full desktop environment that was built using Fyne, FyshOS [10]. ■■■

Info

- [1] Getting Started documentation: <https://developer.fyne.io/started/>
- [2] Creating your own layout code: <https://developer.fyne.io/explore/container>
- [3] Cloud storage with Fyne: <https://youtu.be/lzm7l5SXmN8>
- [4] Cross-platform compiling: <https://developer.fyne.io/started/cross-compiling>
- [5] `fyne-cross`: <https://github.com/fyne-io/fyne-cross>
- [6] Geoffrey: <https://fynelabs.com/geoffrey>
- [7] To-do app source code: <https://github.com/andydotxyz/linuxmagazine-todoapp>
- [8] Other Fyne apps: <https://apps.fyne.io>
- [9] Fyne tutorials and documentation: <https://developer.fyne.io>
- [10] FyshOS: <https://fyshos.com/desktop/>

Author

Andrew Williams is a software engineer and entrepreneur based in Scotland, UK, with experience in many open source technologies, having been a core developer in large projects such as Enlightenment, EFL, Maven, and Fyne. He is the founder of the Fyne toolkit and CEO of Fyne Labs where they work to expand the possibilities of platform-agnostic app development. He is also the author of *Building Cross-Platform GUI Applications with Fyne* and *Hands-On GUI Application Development in Go*.



Network knowledge at your fingertips with NetBox

Truth Teller



NetBox is a single source of information on your network where you can store all those important details that used to get lost.

By Adam Dix

As networks grow and increase in complexity, it becomes more and more difficult to document all of the devices that you manage and their relationships. In the worst of cases, knowledge of the network may be scrawled on a series of spreadsheets or hand-drawn sketches – or maybe even stored inside the heads of a few overworked IT staffers. A number of free and proprietary solutions provide a means for visualizing and documenting the network, often combined with additional management features. Many of these tools are useful, but some may be missing a critical feature that you need or, on the other end of the spectrum, might suffer from feature bloat. Others will lock you into a specific management or monitoring environment. With all of this in mind, it would be nice to use a software package with a solid base plus add-ons, allowing you to build out the best environment for documenting your specific network.

If you're looking for a versatile solution that embraces complexity while still "keeping it simple," you might be interested in NetBox [1]. NetBox claims to be "the premiere network source of truth." It lets you model and document networks by combining traditional Data Center Infrastructure Management (DCIM)

and IP Address Management (IPAM) with extensions and APIs. You can check out an always-running NetBox demo [2] if you want to try it out before taking the plunge and investing your organization's resources.

Information Hub

NetBox serves as a hub of information about your network (Figure 1). At my organization, NetBox has become much more than simply a way to manage IP addresses. Thanks to NetBox's flexibility, you can add notes (using Markdown), add product images (as well as images of the product's location), and much more. If you need to document which device is which in a rack, where that rack is located, and what it looks like, just take a photo of that specific device and upload it with your phone right then and there. Voila, everyone with NetBox access can see the exact machine in its exact location. If all of your devices are the same model and were bought at the same time, installed in the same place, and nicely and neatly aligned in endless rows of racks that go on for miles and miles, this might not be a big deal. For me, dealing with racks, offices, classrooms, libraries, boiler rooms, hallways, etc., and years upon years of undocumented

changes, this ability is very useful. If I am uncertain which machine I need to work on once I get to a room or office, a quick glance at NetBox tells me without requiring me to power anything up, look for an asset tag, or see which port on a switch the device is plugged into. No need to search for product information either, because it's right there, along with links to the drivers.

Speaking of racks, NetBox lets you populate racks with your devices regardless of type and builds the rack view in the web GUI. NetBox uses your provided images of the device type's front and back to populate the front and back elevation views of the rack, which I found extremely handy. Looking for a specific device isn't always as simple as looking for, say, the fifth 4U chassis from the bottom. Instead, I may be looking for a domain controller somewhere in between two NASs, three switches, and a workstation on a shelf on its side with none of the asset tags or stickers remaining. These elevation views, as well as the way that these racks are created, is a very powerful feature, even without considering all of the rest of what the software can do.

NetBox's nicest feature, in my opinion, is that it contains whatever information you choose to include, all in one place and extremely easy to find. For instance, a search for "245" gives me every room with "245" in the name, each device with those digits in its identifiers or comments, and anything with that model

Lead Image © Orson, 123RF.com

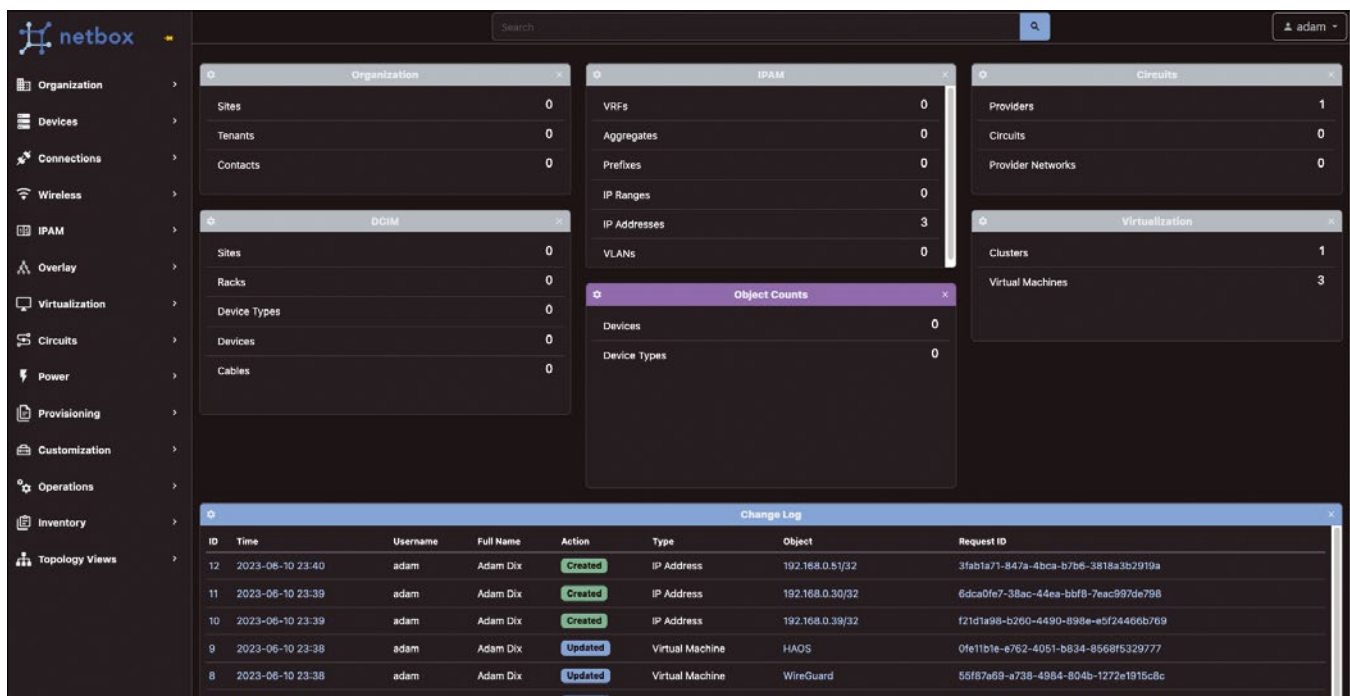


Figure 1: The NetBox overview landing page provides information about your network in one place.

number. My mantra for software like this is that I ought to be able to find what I am looking for within three steps. For instance, I can type “245” and find the room, click on it to see the device in that room, and click on that to get not only the IP address but also links to the owner’s manual, installation guide, generic and specific images of that device, which point on that device connects to its corresponding port on which switch, how much power it requires, and so on.

With that in mind, NetBox is certainly an example of you get out what you put into it. However, I can tell you that a well-executed NetBox installation can quickly become the one-stop shop that your employees go to when they receive a work order or need to know about a specific machine or device type. A rich set of APIs means that NetBox can be extended to serve as an information source for automated provisioning and management tools.

NetBox can even serve as a backup to your inventory management. Because new devices can be added easily by importing CSV files, big projects are also easy to incorporate into NetBox. Personally, I’ll wait until new items are inventoried into the inventory management software and then pull a report of newly added items, format it as needed for NetBox, upload, and then add IPs (Figure 2). The entire process takes perhaps 15 minutes from start to finish to add what is basically an unlimited number of devices of one device type, if you know the IPs and locations where they will be installed.

I have also found advantages to using NetBox that I didn’t expect or even imagine. My team recently needed to answer approximately a half zillion questions pertaining to our fleet as a cybersecurity exercise. NetBox put the answer to nearly all of those questions at our fingertips – no guessing and no estimates. How many desktops are in the high school library? Three steps needed to find out. Where is our domain controller located? Three steps. How many endpoints do we have that are no longer supported by a manufacturer’s warranty? Click, click, click. NetBox has

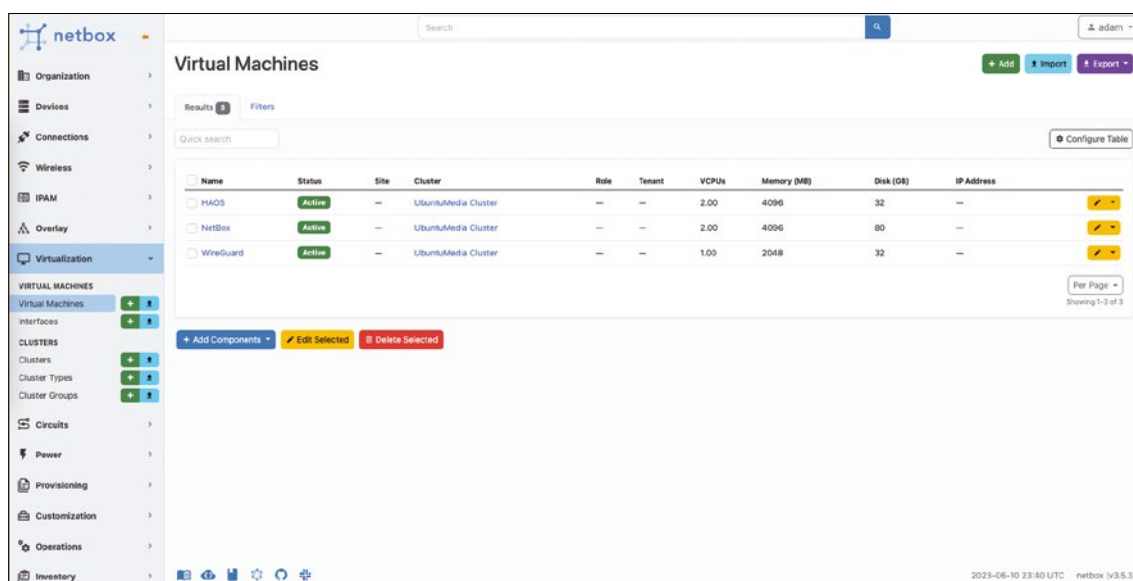


Figure 2: Managing virtual machines in NetBox.

been a godsend in terms of information management. NetBox doesn't require someone to have intimate knowledge of the systems or even be in the IT department. Searching and navigating NetBox is intuitive and simple for all levels of users.

Installation

While you can find the commands for installing NetBox in the NetBox documentation, I will walk you through them here and add some of the caveats, quirks, and counsel that I pieced together when deploying and configuring my particular installation.

I am running NetBox on a bare metal Ubuntu install [3] with a single solid-state drive (SSD) with the database being backed up daily to a network server using a cron job and a script lifted from the Level1Techs forum [4]. In addition, I recommend having at a minimum a backup machine on-site and another off-site, or, better yet, a virtual machine (VM) running in the cloud for failover (setting up these backups is outside the scope of this article). I'll assume that you have one machine to install NetBox and another for a backup (based on your organization's typical operating procedures and standards).

PostgreSQL

First, you need to install PostgreSQL [5] with the following commands:

```
$ sudo apt update
$ sudo apt install -y postgresql
$ psql -V
$ sudo -u
```

The third command above verifies that you have at least version 11 of PostgreSQL installed. Once you are in the PostgreSQL shell (the last command), run the following commands:

```
$ CREATE DATABASE netbox;
$ CREATE USER netbox WITH PASSWORD 'PutYourSuperSecurePasswordHereDONT COPYPASTETHIS';
$ ALTER DATABASE netbox OWNER TO netbox;
```

To leave the PostgreSQL shell, enter:

```
$ \q
```

To double-check that the installation was successful, enter:

```
$ psql --username netbox --password --host localhost netbox
```

You will then enter your *SuperSecure-Password* for the NetBox user password. Please do not use *SuperSecure-Password* for your password; it is neither super nor secure. Follow your own organization's guidelines keeping in mind that, if implemented to its fullest extent, NetBox may well be the keys to your kingdom.

Once you enter your password, you should see the following output:

```
psql (14.8)
Ubuntu 14.8-0ubuntu0.22.04.1)
SSL connection (protocol: TLSv1.3,
cipher: TLS_AES_256_GCM_SHA384,
bits: 256, compression: off)
Type "help" for help.
```

Now complete a connection info check with:

```
netbox=> \conninfo
```

Success: You are connected to database netbox as user netbox on host localhost (address 127.0.0.1) at port 5432 with the SSL connection described above. You can now quit by entering:

```
netbox=> \q
```

Redis

Your next installation step is to install Redis [6] using:

```
$ sudo apt install -y redis-server
```

You need to verify that you are now running at least version 4.0 of Redis with:

```
$ redis-server -v
```

The output you see should be something like this:

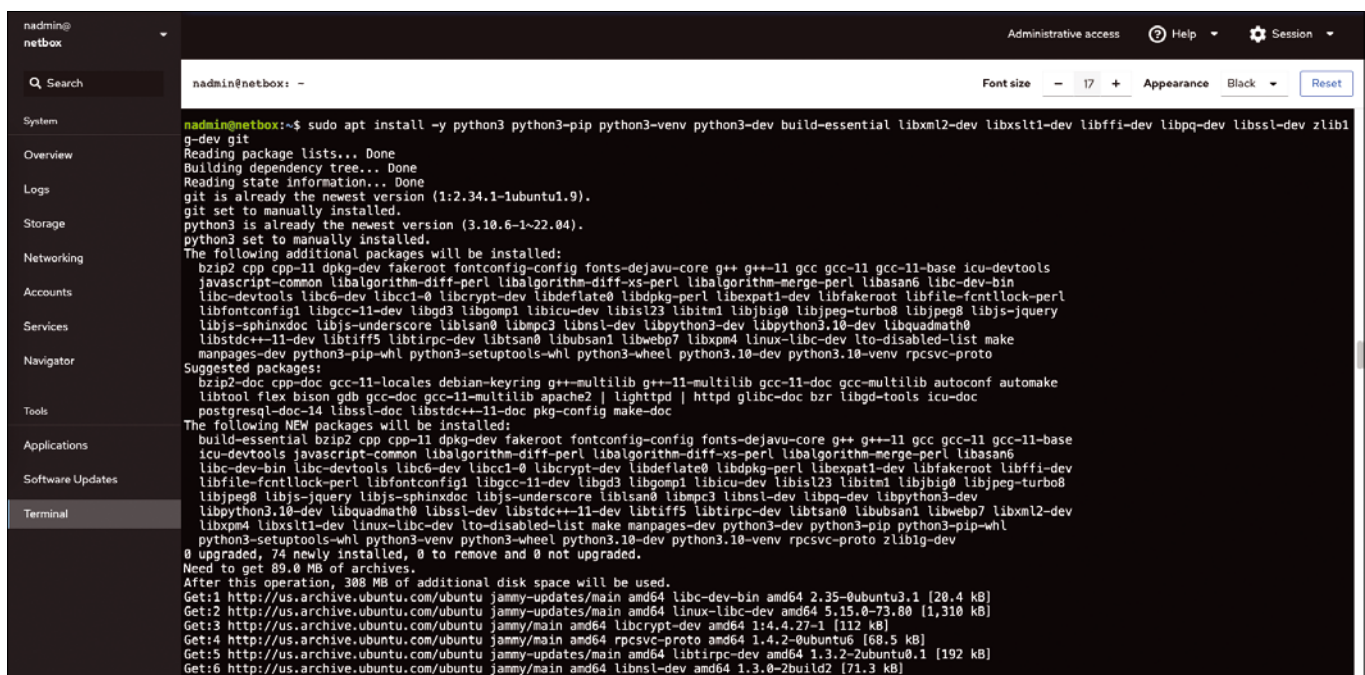


Figure 3: You can install the NetBox dependencies using Cockpit's terminal.

```
Redis server v=6.0.16 sha=00000000:0
malloc=jemalloc-5.2.1 bits=64
build=a3fdef44459b3ad6
```

Lastly, you need to verify that the Redis service is working properly. Enter the following command:

```
$ redis-cli ping
```

which should result in PONG as the response.

NetBox and Dependencies

You will need to also install the NetBox dependencies (as shown in Figure 3) using:

```
$ sudo apt install -y python3
python3-pip python3-venv python3-dev
build-essential libxml2-dev
libxslt1-dev libffi-dev libpq-dev
libssl-dev zlib1g-dev git
```

Double-check that you have at least version 3.8 of Python [7] installed by running:

```
$ python3 -V
```

Now you need to create the installation directory and `cd` (change directory) into it and download NetBox with:

```
$ cd ~
$ sudo mkdir -p /opt/netbox/
$ cd /opt/netbox/
$ sudo git clone -b master --depth 1
```

Listing 1: Modifying Database Parameters

```
01 DATABASE = {
02     'ENGINE': 'django.db.backends.postgresql', # Database engine
03     'NAME': 'netbox', # Database name
04     'USER': 'netbox', # PostgreSQL username
05     'PASSWORD': 'SuperSecurePassword', # PostgreSQL password
06     'HOST': 'localhost', # Database server
07     'PORT': '', # Database port (leave blank for default)
08     'CONN_MAX_AGE': 300, # Max database connection age (seconds)
09 }
```

```
https://github.com/netbox-community/
netbox.git .
```

Note: The period (.) at the end of the last command is required for the command to work properly.

Next you need to create the NetBox user and change ownership of the installation directory to that user with:

```
$ sudo adduser --system --group netbox
$ sudo chown --recursive netbox
/opt/netbox/netbox/media/
```

Configuration

If you read the NetBox documentation, you will notice that my configuration instructions vary somewhat from the documentation. The documentation does mention doing what I recommend, but it doesn't make it obvious, so I will try to make these steps a bit easier to understand.

First, I'll edit the NetBox configuration file using Cockpit [8] on Ubuntu Server (see the "Config Editors" box) as follows:

```
$ cd /opt/netbox/netbox/netbox/
$ sudo cp configuration_example.py
configuration.py
$ sudo nano configuration.py
```

Config Editors

I did the install for this article using Cockpit on Ubuntu Server because (in my experience) it makes administration very easy. I installed a 45Drives add-on for Cockpit called `cockpit-navigator` [9], which greatly improves running servers in general, especially if you need to edit lots of config files and create configs from templates and perform other common configuration actions regularly. You can also do all of this in the terminal (which Cockpit also includes) if you prefer to use something like nano [10] or Vim [11].

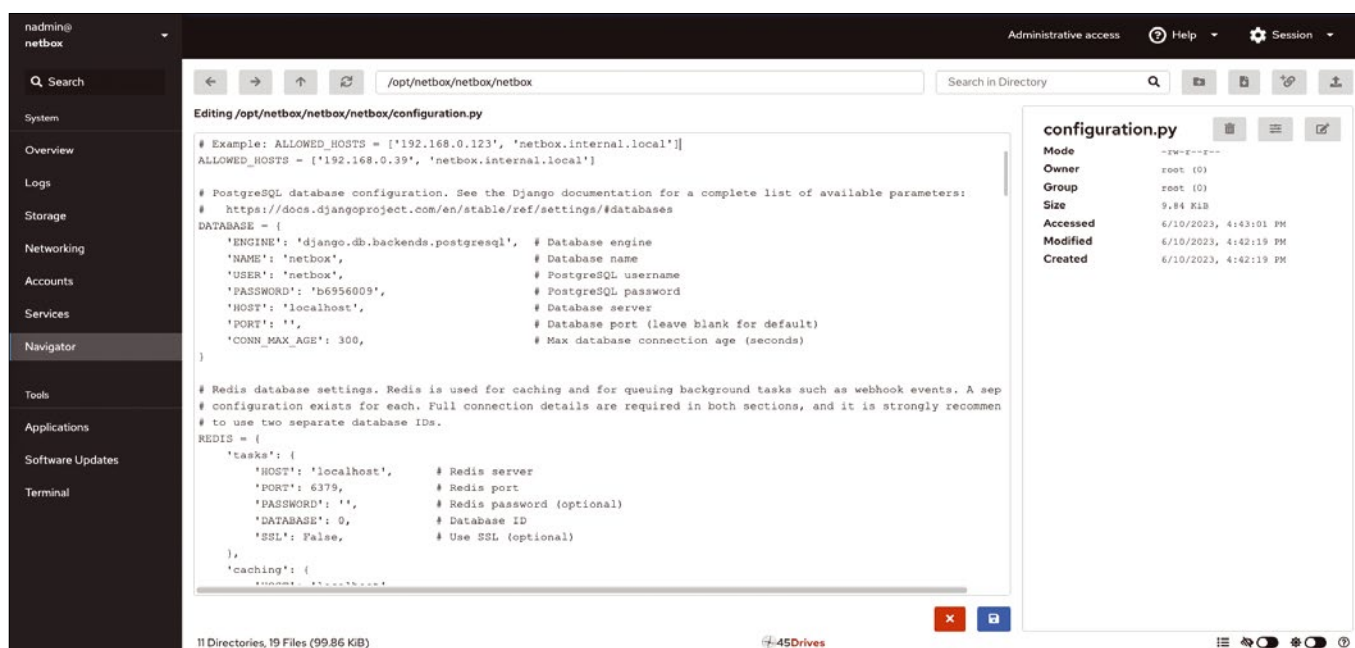


Figure 4: You can edit `configuration.py` with Navigator from 45Drives in Cockpit to set up NetBox if you prefer a graphical editor.

Listing 2: Modifying Redis Parameters

```

01 REDIS = {
02     'tasks': {
03         'HOST': 'localhost', # Redis server
04         'PORT': 6379,       # Redis port
05         'PASSWORD': '',     # Redis password (optional)
06         'DATABASE': 0,      # Database ID
07         'SSL': False,       # Use SSL (optional)
08     },
09     'caching': {
10         'HOST': 'localhost',
11         'PORT': 6379,
12         'PASSWORD': '',
13         'DATABASE': 1,      # Unique ID for second database
14         'SSL': False,
15     }
16 }

```

If you are also installing this on Ubuntu Desktop, use:

```
$ sudo gedit configuration.py
```

For versions of Ubuntu newer than 20.04, use:

```
$ sudo gnome-text-editor configuration.py
```

You will then be able to edit the file directly in a text editor. This might be easier if you are less familiar with using Vim or nano from the terminal.

You will need to modify four keys in the config file (Figure 4). First, you will change the `ALLOWED_HOSTS` parameter:

```
ALLOWED_HOSTS = ['netbox.example.com', '192.0.2.123']
```

If you are unsure about what to enter here, use your server's IP address, which you

can get by typing `$ ip address` into the terminal. You can also type `$ hostname` into the terminal and put your server's hostname here as well, or you can use both if that makes sense. My installation has four entries here, separated by commas. You will enter this in a web browser later to get to the NetBox web GUI.

Warning: Do *not*

type this in the example `ALLOWED_HOSTS` comment directly above the `ALLOWED_HOSTS` parameter. If you get a 400 bad request error, I suggest checking the `ALLOWED_HOSTS` comment to see if you accidentally entered your IP address or hostname there.

Next you need to modify the database parameters as shown in Listing 1.

You then need to modify the Redis parameters as shown in Listing 2.

Finally, you will need to modify the secret key parameter. First, you need to generate your secret key with:

```
$ python3 /opt/netbox/netbox/
generate_secret_key.py
```

Copy and paste the generated key into the `SECRET_KEY` parameter space below the `REDIS` parameter. You will use this key if you decide to set up a second NetBox server for backup, so jot it down for future reference. If you forget to do this, you can

always get back into `configuration.py` to retrieve the secret key if needed later.

Now you need to create and edit the `local_requirements.txt` file with:

```
$ cd /opt/netbox/
$ sudo nano local_requirements.txt
```

If you intend to use a remote database as mentioned in the online documentation, you will want to enter `django-storages` [12] here. What is not clear in the documentation is that you also need to enter any other plugins that you intend to use in the `local_requirements.txt` file in `/opt/netbox`. (For a list of NetBox plugins, see the NetBox Community Plugins GitHub page [13].) In addition, the plugins need to be listed in the previously modified `configuration.py` file in `/opt/netbox/netbox/netbox`.

In `local_requirements.txt`, simply enter each desired plugin (Figure 5), one per line, as follows:

```
django-storages
netbox-topology-views
netbox-inventory
netbox-qrcode
```

In `local_requirements.txt`, note that the plugins use dashes (-). When you edit the `configuration.py` file, you will change some of the dashes to underscores (_). I found this super frustrating and inconsistent because it is not *always* dashes in one place and underscores in the another, but it is often different between the two. (If you build plugins for NetBox and can avoid this, please consider making your plugin names one word to avoid this issue.)

Now that you have edited and saved the

`local_requirements.txt` file to include your desired plugins, you also need to enable them by once again modifying the `configuration.py` file. You can open `configuration.py` with:

```
$ cd /opt/netbox/netbox/netbox/
$ sudo nano configuration.py
```

Look for the `PLUGINS` parameter and, depending on the plugin, you may also need to change `PLUGINS_CONFIG` as well. Based on the plugins I added to my `local_requirements.txt` file, my `PLUGINS` section should look like this:

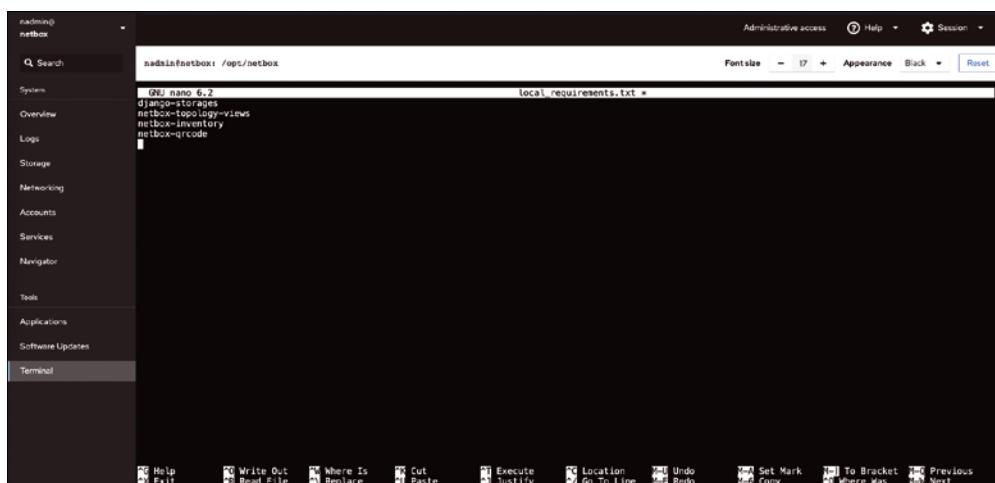


Figure 5: Entering plugins to `local_requirements.txt` with nano.

```
PLUGINS = [
    'netbox_inventory',
    'netbox_qrcode',
    'netbox_topology_views'
]
```

I will use a configuration that is similar to the default suggested plugin configuration as shown in Listing 3.

I tend to switch things up between using nano in the terminal and using Navigator from 45Drives to modify files, but you can also use gedit [14] (or Gnome Text Editor [15]) if using Ubuntu Desktop to modify the `configuration.py` file and to create and populate the `local_requirements.txt` file. I prefer troubleshooting configs with a graphical editor to remembering the quit and save commands in a terminal text editor. It is all the same to NetBox in the end, so use whichever is easiest for you.

I strongly recommend setting the `allow_coordinates_saving` and `always_save_coordinates` to False because they ended up causing me quite a bit of grief. It appears that what happens is that they will set a custom field for each device. The problem with that is that *you* will actually need to create that custom field, and failure to do so will stop you from being able to edit the devices in any way after using the topology view. In short, I would set those parameters to false and read the plugin page carefully. If you get errors when trying to make changes to your devices after using the topology view plugin, create a custom field called *Coordinates* that is data type text and is hidden. Then follow the instructions on the plugin page to do the needed conversion. After that you will again be able to edit devices without getting an error message each time.

Having said that, the topology views plugin looks downright awesome, and I highly recommend it for creating topology maps. Just be aware that this error could occur and that there is a fix, even if it is a bit convoluted. I was able to tweet at the NetBox team, and the

wonderful folks there helped me to find the GitHub issue on the plugin page that I needed to get it all sorted out. (Unfortunately, prior to that, I had given up the night before and tried recreating all of my devices – only to find myself in the same exact situation after looking at the topology view. Luckily for me, it wasn't too many devices and it was a great learning experience.)

With the plugins sorted, you are ready to run `upgrade.sh` to install the plugins:

```
$ sudo /opt/netbox/upgrade.sh
```

Note, the syntax needs to be exact for the `configuration.py` file. I have found that even copying and pasting often results in errors. If you make an error, carefully read the output from the terminal and then go back and edit the `configuration.py` file until you get it right. The terminal output will tell you which line the issue is on and give a best guess as to the source of the problem. Remember, you can always add plugins and their configurations later by putting them into `local_requirements.txt` and `configuration.py` and running the `upgrade.sh` command in the future. I ended up using the configuration above, but that was not without some frustration. Bear in mind that the

Listing 3: Plugin Configuration

```
01 PLUGINS_CONFIG = {
02     "netbox_inventory": {},
03     "netbox-qrcode": {
04         'with_text': True,
05         'text_fields': ['name', 'serial'],
06         'font': 'ArialMT',
07         'text_location': 'up',
08         'qr_version': 1,
09         'qr_error_correction': 0,
10         'qr_box_size': 4,
11         'qr_border': 4,
12         'cable': None,
13         'rack': {
14             'device': {
15                 'qr_box_size': 6,
16                 'custom_text': None,
17             }
18         }
19     },
20     "netbox_topology_views": {
21         'static_image_directory':
22         'netbox_topology_views/img',
23         'allow_coordinates_saving': True,
24         'always_save_coordinates': True
25     }
26 }
```

Listing 4: Failed Restart Message

```
$ sudo systemctl restart netbox netbox-rq
Failed to restart netbox.service: Unit
netbox.service not found.
Failed to restart netbox-rq.service: Unit
netbox-rq.service not found.
```

```
nadmin@netbox: /opt/netbox
ipam.fhrpgroup... No objects found.
ipam.ipaddress... No objects found.
ipam.iprange... No objects found.
ipam.l2vpn... No objects found.
ipam.prefix... No objects found.
ipam.rir... No objects found.
ipam.role... No objects found.
ipam.routetarget... No objects found.
ipam.service... No objects found.
ipam.servicetemplate... No objects found.
ipam.vlan... No objects found.
ipam.vlangroup... No objects found.
ipam.vrf... No objects found.
extras.journalentry... No objects found.
tenancy.contact... No objects found.
tenancy.contactgroup... No objects found.
tenancy.contactrole... No objects found.
tenancy.tenant... No objects found.
tenancy.tenantgroup... No objects found.
virtualization.cluster... No objects found.
virtualization.clustergroup... No objects found.
virtualization.clustertype... No objects found.
virtualization.virtualmachine... No objects found.
virtualization.vminterface... No objects found.
wireless.wirelesslan... No objects found.
wireless.wirelesslangroup... No objects found.
wireless.wirelesslink... No objects found.
netbox_inventory.asset... No objects found.
netbox_inventory.supplier... No objects found.
netbox_inventory.purchase... No objects found.
netbox_inventory.inventoryitemtype... No objects found.
netbox_inventory.inventoryitemgroup... No objects found.
Completed.
Removing expired user sessions (python3 netbox/manage.py clearsessions)...
Clearing the cache (python3 netbox/manage.py clearcache)...
Cache has been cleared.
Upgrade complete! Don't forget to restart the NetBox services:
> sudo systemctl restart netbox netbox-rq
nadmin@netbox: /opt/netbox$
```

Figure 6: This is what you should see at the end of a successful `upgrade.sh` run during the initial installation.

sample configurations in the NetBox documentation are just that and might need to have something added, removed, or otherwise changed in order to work properly.

NetBox now will tell you to restart the NetBox services (Figure 6). If you restart at this time, you will get the failure to restart message shown in Listing 4. You get the failure message because you have not yet created those services.

If you decide for whatever reason that you no longer want a plugin, then you can simply remove it from both files. If you are unsure, or if you would like to temporarily disable a plugin for some reason, then you can do so by removing it from `configuration.py` but leaving it in `local_requirements.txt`. In order to enable or disable a plugin, you will need to restart the NetBox service, which I will explain shortly.

You now need to create the NetBox superuser by entering:

```
$ source /opt/netbox/venv/bin/activate
```

The `(venv)` added to the command prompt is normal. Now, add the superuser:

```
(venv) $ cd /opt/netbox/netbox
(venv) $ python3 manage.py createsuperuser
```

You can use a `deactivate` command to leave the virtual environment and get

Listing 5: Creating Services

```
01 $ sudo cp /opt/netbox/contrib/gunicorn.py /opt/netbox/gunicorn.py
02 $ sudo cp -v /opt/netbox/contrib/*.service /etc/systemd/system/
03 $ sudo systemctl daemon-reload
04 $ sudo systemctl start netbox netbox-rq
05 $ sudo systemctl enable netbox netbox-rq
06 $ systemctl status netbox.service
```

back to a normal terminal window, but I usually found myself closing and opening a new terminal (or rather refreshing the page because I was connected using Cockpit).

With the superuser added, you will want to add the housekeeping script as a cron job to occasionally do some cleanup. To do this, add a link to the script in the `cron.daily` folder as follows:

```
$ sudo ln -s /opt/netbox/contrib/netbox-housekeeping.sh /etc/cron.daily/netbox-housekeeping
```

At this point, you should have a running NetBox instance that you can connect to by entering the IP address or hostname that you defined earlier in `configuration.py`. You will need to have the Uncomplicated Firewall (UFW), which might require running:

```
$ sudo ufw allow 9090
$ sudo ufw allow OpenSSH
$ sudo ufw allow https
$ sudo ufw allow http
```

```
$ sudo ufw enable
$ sudo ufw status
```

I added ports 9090 and 22 because I was using Cockpit and also wanted the ability to connect to this server using SSH, but that is beyond the scope of this article. If you don't know how to SSH into a server or haven't installed and used Cockpit or a similar server admin web GUI such as Webmin, I suggest you do those things first, because it will help to familiarize you with running servers without a GUI from the terminal. You can connect to your machine with SSH or Cockpit even if you are running the desktop version of Ubuntu.

Creating Services

You will now want to copy the default Gunicorn [16] configuration and create the services to turn NetBox on and off using the code in Listing 5.

The code in Listing 5 copies the default Gunicorn config file and then creates the NetBox system service, starts it, enables it to run each time the system

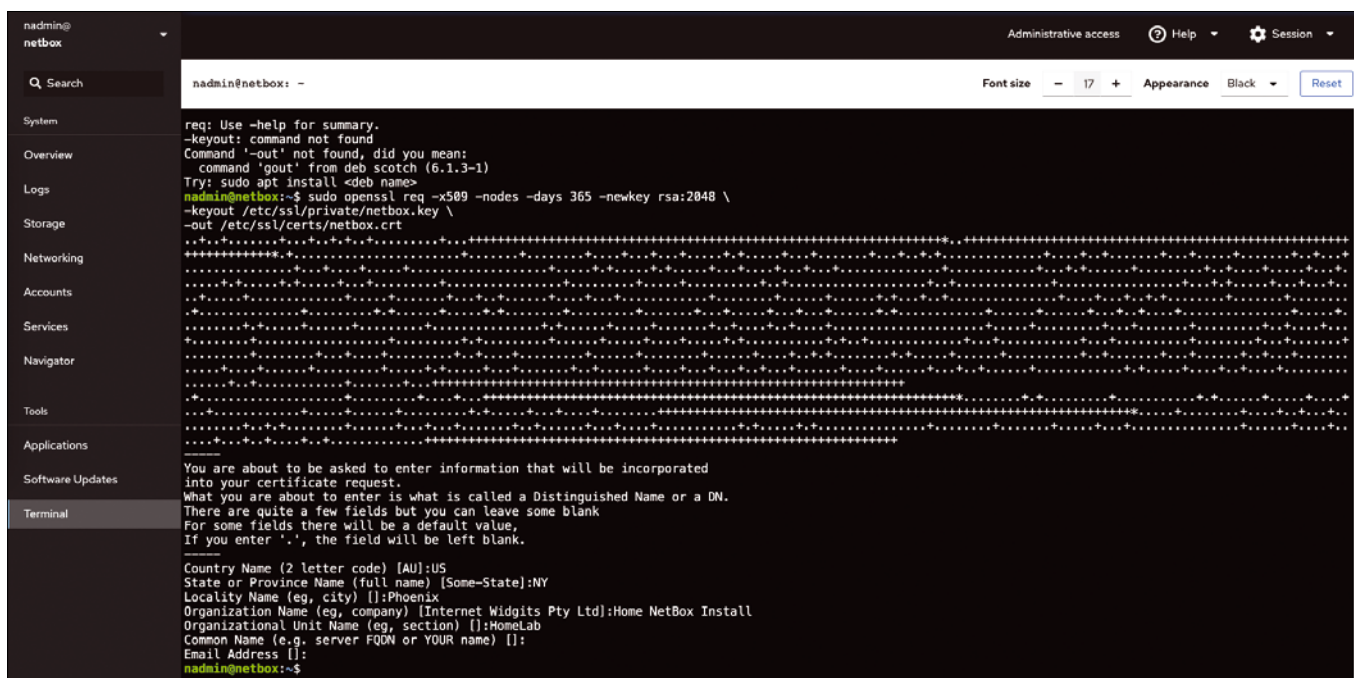


Figure 7: Making a self-signed SSL certificate for HTTPS in NGINX.

starts up, and checks to make sure that it is running properly and enabled to run on startup. After you complete this step, if you run the `upgrade.sh` command and it prompts you to restart the NetBox services, you now will be successful.

Web Server Setup

To set up the web server, you will want to create an SSL certificate to allow HTTPS encrypted traffic (Figure 7):

```
$ sudo openssl req -x509 -nodes \
  -days 365 -newkey rsa:2048 \
  -keyout /etc/ssl/private/netbox.key \
  -out /etc/ssl/certs/netbox.crt
```

This certificate is self-signed, which is fine for our purposes here. For production use, you most likely will not want to use a self-signed certificate.

Next, you will be prompted for some optional details that you might want to include in the certificate. I suggest making a little game out of it to see if anyone actually reads your certificate details!

Now you will install the NGINX web server [17]:

```
$ sudo apt install -y nginx
```

The next step is critical: You must add your hostname and IP address from earlier to the server's configuration file:

```
$ sudo nano \
  /opt/netbox/contrib/nginx.conf
```

Change `netbox.example.com` to the hostname or IP address that you entered

Listing 6: Setting Up the NGINX Server

```
01 $ sudo cp /opt/netbox/contrib/nginx.conf /etc/nginx/sites-available/netbox
02 $ sudo rm /etc/nginx/sites-enabled/default
03 $ sudo ln -s /etc/nginx/sites-available/netbox /etc/nginx/sites-enabled/netbox
04 $ sudo systemctl restart nginx
```

previously in `configuration.py`. Note that you can have more than one entry here, such as the installation's IP address and hostname. Simply separate them with a space. You can use the `$ ip address` and `$ hostname` commands in the terminal to find out this information if you've forgotten.

Now you will copy the NetBox HTTPS config file to the NGINX directory, delete the default enabled site, replace it with a link to this one, and restart the HTTPS service by entering the commands in Listing 6 in order.

If you make any future changes to the config file, you need to recopy that file to the `sites-available` directory using the command in line 1 of Listing 6 and then by restarting NGINX using line 4. Failure to do so will result in your configuration not actually changing.

(There is an optional LDAP service installation section in the online NetBox documentation as well, but that is beyond the scope of this article.)

You should now be up and running (Figure 8). You can access your NetBox installation by going to your web browser and entering the hostname or IP address that you used in the NGINX configuration file and in `configuration.py` in the addressbar. You will log in with the username and password that

you previously created. Upon login, I suggest creating a new user for yourself and whoever else needs access to NetBox.

When you connect to your new NetBox instance using this SSL certificate, you will get a warning about connecting to an insecure server (Figure 9). For my installation that isn't a concern, but it might be for you. If that is the case, you can look into a certificate that is not self-signed. It is probably safe in this case to ignore that warning altogether for the purposes of this article.

Assigning User Rights

You can give users rights to view only or to edit. These rights are based on each individual subsection of each aspect of the network. This is quite granular, which is fantastic but also a bit confusing. I strongly suggest using groups smartly – think first about the best way to set permissions for your organization.

There is also an add-on for single sign-on (SSO), which might make sense for your installation. You may want one person to add all of the new devices and another person to add IP addresses to those devices. In that case, you would give the first person access to change device types and devices, but only to view IP addresses, and vice versa.

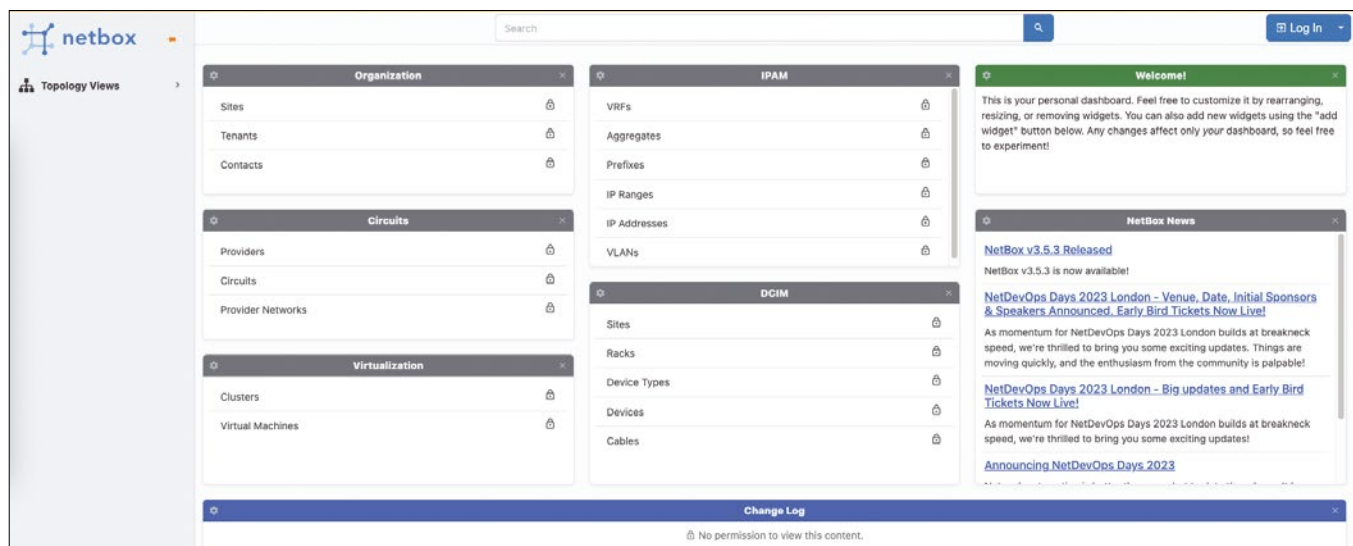


Figure 8: If you made it here, congratulate yourself!

Some users may not need to see a particular section, so you can simply not give them that access. This is also useful if you have users from different sites or locations all connecting to your NetBox instance. In this instance, users may have the ability to only change items within their location or site as needed.

Permissions are a big deal and need to be well thought out. Remember, these are the keys to the kingdom. Even if that isn't really the case for your organization, it is probably a safe approach to NetBox.

Conclusion

When I first started reading about NetBox, I found their aim of being “the premiere network source of truth” kind of cheesy. It's just a web GUI for a network map with some stuff from a database after all. Since installing and using it, however, I'm a true believer. I think every single network should be mapped-out using NetBox. I have installed the community edition at work as well as on my home network. It really is your network's “truth” – or at least it becomes that once you put in the work.

I am still in the process of documenting the network in my organization. It is an enormous task and one that never ends, because updates will need to be made as the network expands, contracts, and changes. My intent is to build NetBox out thoroughly enough that I can give a new employee a login to NetBox so they can find their way around the entire organization, even if they aren't able to access everything that they see in there. I also needed to add maps for each building in the NetBox's location

section, which I could do by simply uploading PNG images of our physical maps. I might not give a new employee the credentials for our building's switches, but they will know where the switches are located, what devices connect to them and through which ports, and the IP addresses those devices have, as well as be able to very quickly and easily find product information. NetBox will give new employees all that they need to know about the devices that they will work on to be able to effectively and efficiently do their jobs. With dozens of printers and hundreds of PCs on our network, the ability to pinpoint one device and its connection to a specific switch by searching for a room or a building, and to have a link to the user's guide and installation manual, drivers download page, specifications, asset tag number and corresponding serial number, and so on, will greatly improve team efficiency in the long run, though it will absolutely take time to implement. None of that time is wasted, however, because anyone who is mapping out the network in NetBox is also learning all of the ins and outs of it simultaneously.

Something incredibly important that I didn't focus on in this article: Knowing what is on your network and where that gear is located is critical to network security and is connected directly to CIS Safeguards numbers 1.1, 1.2, 3.2, 4.6, and many others in the CIS Risk Assessment Methodology (RAM)[18].

Initially, I found a ton of overlap between tools that we were already using and NetBox. However, as I added more devices to the database and put in more details

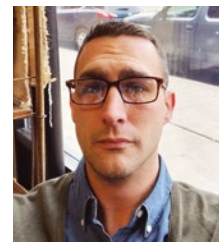
about each device, I realized that I had been looking at it all wrong. While alternatives exist to basically all of NetBox's functions, the disconnect between the individual programs and services creates inefficiency. NetBox offers a fantastic opportunity for long-term improvement in network management. NetBox, if well-implemented, will act as a central knowledge hub for your network, with each of those other programs and services functioning as spokes off of that hub. NetBox doesn't lay over the top of your existing management stack, but instead coalesces with it in a way that empowers your IT team. While NetBox may initially look like a lot of work, you will wonder what you ever did without it once you get it up and running. ■■■

Info

- [1] NetBox: <https://github.com/netbox-community/netbox/wiki>
- [2] NetBox demo: <https://demo.netbox.dev/login?next=/>
- [3] Ubuntu: <https://ubuntu.com/>
- [4] Level1Techs Guide to NetBox: <https://forum.level1techs.com/t/netbox-ipam-dcim-guide/132435>
- [5] PostgreSQL: <https://www.postgresql.org/>
- [6] Redis: <https://redis.io/>
- [7] Python: <https://www.python.org/>
- [8] Cockpit: <https://cockpit-project.org/>
- [9] 45Drives cockpit-navigator: <https://github.com/45Drives/cockpit-navigator>
- [10] nano: <https://www.nano-editor.org/>
- [11] Vim: <https://www.vim.org/>
- [12] Django: <https://www.djangoproject.com/>
- [13] NetBox Community Plugins: <https://github.com/netbox-community/netbox/wiki/Plugins>
- [14] gedit: <https://github.com/GNOME/gedit>
- [15] Gnome Text Editor: <https://gitlab.gnome.org/GNOME/gnome-text-editor>
- [16] Gunicorn: <https://gunicorn.org/>
- [17] NGINX: <https://www.nginx.com/>
- [18] CIS RAM: <https://www.cisecurity.org/insights/white-papers/cis-ram-risk-assessment-method>

Author

Adam Dix, a former teacher and product line manager, is a mechanical engineer and Linux enthusiast working as a LAN technician. You can



check out some of his related work at EdUBudgie Linux (<https://www.edubudgie.com>).

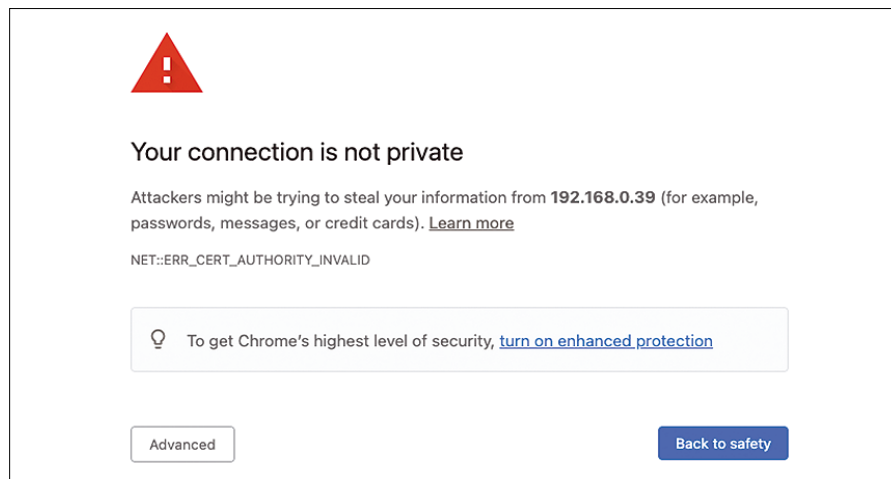


Figure 9: When you first connect to your NetBox installation you may get a privacy warning if you use a self-signed SSL certificate.

Looking for your place in open source?



Set up job alerts and get
started today!

OpenSource
JOB HUB



opensourcejobhub.com/jobs

Algorithm-driven team building

Team Spirit

Instead of the coach determining the team lineup, an algorithm selects the players based on their strengths for Mike Schilli's amateur soccer team. *By Mike Schilli*

At the weekly pickup soccer game in my adopted hometown of San Francisco, with 16 players signed up for an eight-on-eight match, there is always the question of who is going to compete against whom on match day. Preparing the team sheet is usually something taken care of by experienced players, but it is time-consuming and it leads to endless discussions about whether the eight players on one side are simply too skilled for the other eight players, leading to a lopsided game that is no fun to play. To resolve this, I've developed an algorithm to generate the team lineup. This allows me to take an interesting excursion into the world of combinatorics and explore the development of suitable algorithms in this month's column.

How many ways are there to assign 16 players to two teams of eight players each? If one team is fixed, the opponent's team is automatically derived

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 he will gladly answer any questions.



from the remaining participants; this means that the result is written in combinatorics style as “8 over 16,” which you compute as

$$(16*15*...*10*9) / (8*7*...*2*1)$$

with a result of 12,870. Any home computer can churn through that many combinations effortlessly. Even if your task was to assign 22 players to two teams of 11 each, it's in the same league with 705,432 possibilities.

Brute Force

In small event spaces like this, it is definitely an option to simply try out all the combinations, call an evaluation function on each pass, store the best scores, and output the top result at the end. Using this brute force method for a larger event space would simply not work, because combinations can easily spiral out of control exponentially. It would make more sense to first choose a non-perfect but reasonable solution as a starting point and then change it incrementally to see if the results improve or get worse. Simulated annealing is the technical term for this process [1].

Before moving on to find better solutions, you first need a metrics function to grade the quality of each team. This metric can then be used to compare different constellations and select the best one. Like the mantra of all management

courses, “You can't manage what you can't measure.” However, before you can determine the quality of a team, an expert has to grade the strength of the individual players on a few criteria, and then an algorithm can assign the teammates to teams and compare their cumulative strengths.

Listing 1: teams.yaml

```
01 players:
02   - name: joe
03     skills:
04       offense: 3
05   - name: tony
06     skills:
07       defense: 2
08       goalie: 3
09   - name: fred
10     skills:
11       defense: 2
12   - name: mike
13     skills:
14       offense: 9
15   - name: manuel
16     skills:
17       goalie: 9
18   - name: mark
19     skills:
20       defense: 9
```

Lead Image © bowrie15, 123RF.com



Collecting Points

The individual player ratings are shown in the YAML file in Listing 1. For the purpose of illustration, this model is limited to a total of six players. Each entry in `players` identifies a player by name and specifies their rating in three categories with values from 0 to 9: offense, defense, and goalie (i.e., the willingness to play as a goalkeeper). If a player does not have a rating in a category, the algorithm simply assumes a value of 0.

For example, Listing 1 rates player `tony` as a 3 in terms of goalkeeping skills, while player `manuel` has a goalie value of 9 (any matches with the names of

existing goalkeepers are pure coincidence). Because every team needs a goalkeeper and nobody else has goalkeeper qualities, a good algorithm should assign `tony` and `manuel` to play on different teams. On top of this, the algorithm needs to ensure that both teams have roughly equal offensive and defensive players to keep the match exciting and open.

Listing 2 reads the YAML data from the file and then stores it in a data structure for the team-building Go binary later. Listing 3 shows the main program that defines what happens after you call it from the command line. As a kind of warm-up, Listing 4

shows a simple algorithm that randomly assigns players to two teams. You can see the output from the program in Figure 1. The teams are quite

```
$ ./random
Team A
1.   joe [off:3 def:0 goa:0]
2.  fred [off:0 def:2 goa:0]
3.  mark [off:0 def:9 goa:0]

Team B
1.  mike [off:9 def:0 goa:0]
2. manuel [off:0 def:0 goa:9]
3.   tony [off:0 def:2 goa:3]
```

Figure 1: Randomly created teams often have significant differences in performance.

lopsided and the algorithm obviously far from perfect, but stay tuned, improvements are coming.

Strict Types

For the Go program to be able to understand the YAML data with the player skills, it first needs to define similar data structures because of Go's strict type checking. Listing 2 reads in the entire YAML structure from Listing 1 which consists of a dictionary under the

Listing 2: data.go

```
01 package main
02
03 import (
04     "gopkg.in/yaml.v2"
05     "io/ioutil"
06 )
07
08 type Skills struct {
09     Goalie int `yaml:goalie`
10     Offense int `yaml:offense`
11     Defense int `yaml:defense`
12 }
13
14 type Player struct {
15     Name string `yaml:name`
16     Skills Skills `yaml:skills`
17 }
18
19 type Config struct {
20     Players []Player `yaml:players`
21 }
22
23 func yamlParse() ([]Player, error) {
24     config := Config{}
25
26     text, err := ioutil.ReadFile("teams-real.yaml")
27     if err != nil {
28         return config.Players, err
29     }
30
31     err = yaml.Unmarshal([]byte(text), &config)
32     if err != nil {
33         return config.Players, err
34     }
35
36     return config.Players, nil
37 }
```

Listing 3: teams.go

```
01 package main
02
03 import (
04     "fmt"
05 )
06
07 func main() {
08     players, err := yamlParse()
09     if err != nil {
10         panic(err)
11     }
12
13     teams := mkTeams(players)
14     if err != nil {
15         panic(err)
16     }
17
18     teamNames := []string{"A", "B"}
19     for tIdx, team := range teams {
20         fmt.Printf("Team %s\n", teamNames[tIdx])
21         for pIdx, player := range team {
22             fmt.Printf("%d. %8s [off:%d def:%d goa:%d]\n",
23                 pIdx+1, player.Name, player.Skills.Offense,
24                 player.Skills.Defense, player.Skills.Goalie)
25         }
26         fmt.Print("\n")
27     }
28 }
```

players key, containing an array of player structures. Each key contains a name and defines the player's ability under the skills key with the skill name and numerical ratings from 0 to 9.

Based on this, line 19 of the Go code in Listing 2 defines the Config structure with a Players field, whose value is an array slice with Player type elements. This type, defined in line 14, in turn, contains the Name field for the player's name and a Skills field of the same type. The skills type is defined starting in line 8 and specifies the player's strength in values of the int type in the Goalie, Offense, and Defense fields.

Because Go can handle YAML data natively, the code following the field names and types can provide useful hints in backticks (e.g., if the field name used in YAML differs from the one used in the type). In our case, the field names in the Go data structure are uppercase (the YAML library wants it that way), but the YAML fields in the configuration are lowercase. For example, line 9, Goalie int

`yaml:goalie`, defines the Goalie field as being of the int type while its counterpart in the YAML data is named goalie.

That's all the YAML evaluator yaml.Unmarshal() needs in line 31. It works through all the elements of the configuration's YAML array in one go, populates the program's internal Go structures with them, and also takes care of allocating the required memory resources – a miracle of modern computer science. For example, if the code wants to access the offensive skills of the third player from the top later, it can access the

associated integer value as config.Players[2].Offense.

No Plan

Listing 3 defines the main() program, which first calls yamlParse() to read the

Listing 4: random.go

```
01 package main
02
03 import (
04     "math/rand"
05     "time"
06 )
07
08 func mkTeams(players []Player) [][]Player {
09     rand.Seed(time.Now().UnixNano())
10
11     rand.Shuffle(len(players), func(i, j int) {
12         players[i], players[j] = players[j], players[i]
13     })
14
15     if len(players) < 2 || len(players)%2 != 0 {
16         panic("Needs even number of players")
17     }
18
19     return [][]Player{
20         players[0 : len(players)/2],
21         players[len(players)/2:],
22     }
23 }
```

Listing 5: Compiling

```
$ go mod init teams
$ go mod tidy
$ go build random.go teams.go data.go
```

Listing 6: diff.go

```
01 package main
02
03 func diff(teamA []Player, teamB []Player) int {
04     d := Skills{}
05
06     for idx, player := range teamA {
07         s1 := player.Skills
08         s2 := teamB[idx].Skills
09
10         d.Offense += s1.Offense - s2.Offense
11         d.Defense += s1.Defense - s2.Defense
12         d.Goalie += s1.Goalie - s2.Goalie
13     }
14
15     return 2 * abs(d.Offense) +
16         abs(d.Defense) +
17         10 * abs(d.Goalie)
18 }
19
20 func abs(x int) int {
21     if x < 0 {
22         return -x
23     }
24     return x
25 }
```

Listing 7: brute.go

```
01 package main
02
03 func mkTeams(players []Player) [][]Player {
04     bestTeams := make([][]Player, 2)
05     bestDiff := -1
06
07     perm(players, func(t1 []Player, t2 []Player) {
08         d := diff(t1, t2)
09         if bestDiff < 0 || d < bestDiff {
10             bestDiff = d
11             t1c := make([]Player, len(t1))
12             copy(t1c, t1)
13             t2c := make([]Player, len(t2))
14             copy(t2c, t2)
15             bestTeams = [][]Player{t1c, t2c}
16         }
17     })
18
19     return bestTeams
20 }
```

YAML data from Listing 2. Line 13 then calls the `mkTeams()` function defined further down, to line up the two teams. Its return value is a `teams` array slice containing two elements which represent, you guessed it, teams. These elements in turn each consist of array slices of players of the `Player` type. The `for` loop starting in line 19 iterates over both teams; the inner `for` loop starting in line 21 then iterates over the individual players. `Printf()` prints out the name of each player in a team and a summary of the player's ratings according to the three predefined criteria.

Now, for the warm-up, Listing 4 shows a simple algorithm to randomize the teams. The implementation of `mkTeams()` in Listing 4 first initializes a random generator from the standard `math/rand` package using a seed with the current time. This is to ensure that calls to the virtual coach return different teams each time. The `rand.Shuffle()` function then shuffles the elements of the array slice with the 16 players starting in line 11. The return statement starting in line 19 returns the two halves of the slice to the caller. That completes the team lineup.

Before the actual `build` command to generate the Go binary, the compiler first needs to retrieve the YAML package used in Listing 2 from GitHub and compile it (Listing 5, first two lines). Then, the call from the last line of Listing 5 ties everything together to form the random

binary, whose call on the command line is shown in Figure 1 along with the generated output.

Systematic Approach

Unfortunately, this algorithm comes up short. As Figure 1 shows, the random number generator assigned the only two players with goalkeeping skills to the same team, even though each team needs at least one goalkeeper. The offensive and defensive capabilities are also unbalanced. For this reason, the next stage in the algorithm design needs to assign the players to two equal teams based on their individual strengths.

As I mentioned earlier, due to the small event space, it is easy to work through all of the options, determining and comparing the suitability of the individual solutions. For this purpose, Listing 6 provides a suitability function `diff()` that rates the fairness of the assignments in the two teams, `teamA` and `teamB`. If the integer result is close to zero, the match is balanced; the larger its value, the more unfair the match is likely to be.

To determine the score, the function adds the absolute differences of the teams

in terms of offense, defense, and goalkeeping capability. It weights the offensive skills with a factor of 2 and applies a factor of 10 to the goalkeeper; after all, if you don't have a goalkeeper, you've already lost the match.

Quirky Go

You'd think that a programming language would inherently provide a function for determining the absolute value of an integer, but Go's quirky developers decided against it. The `math` package does provide an `Abs()` function for floats, but using it with integers requires tons of conversion operations. This is why, starting in line 20, Listing 6 implements an `abs()` function for turning negative values positive.

This gives Listing 7 the ability to serve up a new version of the `mkTeams()` function as an alternative to the random product in Listing 4. It calls `perm()` to work through all the possibilities, evaluating the resulting groupings with `diff()`, and storing the best constellation

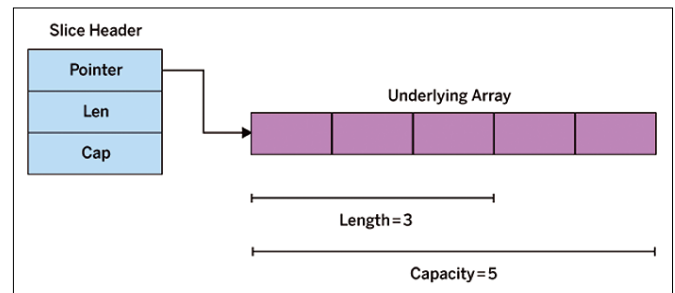


Figure 2: An array slice in Go only references a real array.

Listing 8: perm.go

```
01 package main
02
03 func perm(players []Player, f func([]Player, []Player)) {
04     _perm(players, 0, []Player{}, []Player{}, f)
05 }
06
07 func _perm(players []Player, idx int, subset1 []Player, subset2 []Player, f func([]Player, []Player)) {
08     if len(subset1) == len(players)/2 && len(subset2) == len(players)/2 {
09         f(subset1, subset2)
10         return
11     }
12
13     if idx == len(players) {
14         return
15     }
16
17     _perm(players, idx+1, append(subset1, players[idx]), subset2, f)
18     _perm(players, idx+1, subset1, append(subset2, players[idx]), f)
19 }
```

(i.e., the one with the lowest value) in `bestTeams` and `bestDiff`.

When storing the best results, though, it is important to avoid a hidden stumbling block: If you thought you could just keep the values of the `t1` and `t2` variables with the teams in a safe place, think again. Because the Go runtime modifies the teams referenced by `t1` and `t2` in the underlying array, you end up with garbled lineups. An array slice in Go is always just a pointer to a real array (Figure 2). If you only save the pointer, you will inevitably see arbitrary changes to the underlying array.

Lines 11 to 14 avoid this trap by duplicating the teams, once found, using Go's built-in `copy()` function. It is also

important to make sure that the target array slice (the first argument for `copy()`) is the same length as the original that you want to copy. If the length of the former is zero, `copy()` will simply copy zero bytes, leaving the user pulling their hair out in frustration.

So how does Listing 7 find all possible team constellations? To do this, Listing 7 calls the `perm()` function from Listing 8 in line 7; the function expects an array slice of available players and a callback function that it calls exactly once for each combination it finds. With each individual datapoint, as shown in Listing 7 starting in line 7, the program can then unleash actions on the two teams, such as rating them and keeping the best results found so far.

Help from ChatGPT

Because I was lazy, I didn't wrack my brain to come up with the permutation algorithm in Listing 8 myself, but relied on the assistance of the all-singing-all-dancing electronic brain on ChatGPT. I simply asked ChatGPT in plain English for a method that could split an array of *N* players into all possible combinations of two teams. It came back with a solution within 10 seconds (Figure 3) – a lifesaver for overworked programmers!

All that remained to be done was to adapt the code to the `Player` array element data structure used for the teams, and Listing 8 was ready. This definitely puts the fun back in programming! What you need to know, however, is that the solutions presented by ChatGPT are often not 100-percent correct. In fact, the first result it gave me had an off-by-one bug. However, as soon as I pointed it out, the AI apologized and



Figure 3: ChatGPT wrote some of the Go code for this column.

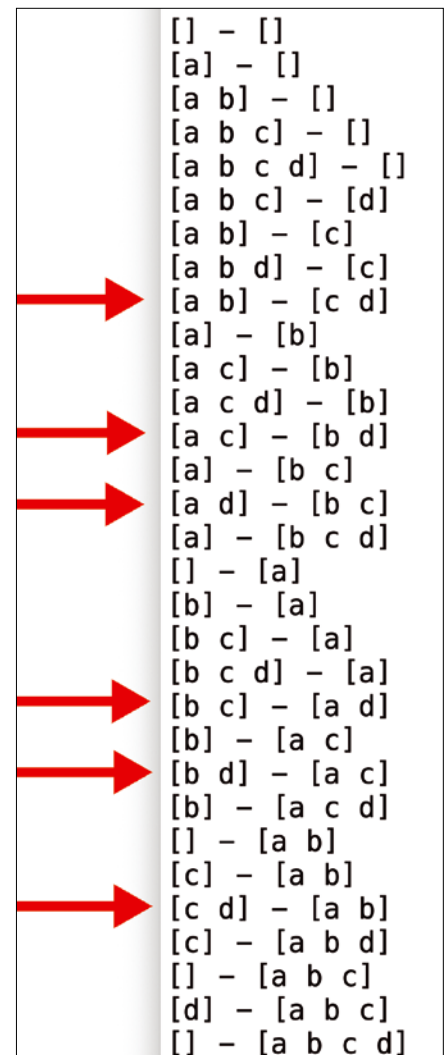


Figure 4: Valid two-by-two lineups for four players.

promptly came up with a better solution. Like Magic!

The permutation algorithm designed by ChatGPT is based on recursion. It calls the `perm()` function twice with two initially empty team slices, `subset1` and `subset2`. To do this, the exported `perm()` function starts the `_perm()` function, masked by an underscore, in line 4 of Listing 8, populating it with the two subset parameters and the index of the element currently being processed. With `idx` initially set to 0, the function remembers its current position in the array slice of available players and then calls itself twice with two new tryout teams. In one call, it appends the current element referenced by `idx` to `subset1`, while it appends it to `subset2` in the following call (lines 17 and 18).

This recursively creates all possible constellations and reliably divides the array elements into two subgroups. However, the function does not pay attention to whether

the teams actually contain all the players required to create a team. In fact, it even leaves the subsets empty in parts (Figure 4).

That's not a deal-breaker. The code in Listing 8 starting in line 8 checks whether the groups generated here have the required number of players and ignores any solutions that do not meet this criterion. Line 9 only calls the callback function `f` with the two team candidates for numerically matching constellations; these are marked by the red arrows in Figure 4. The caller then uses this to evaluate the constellation to home in on the best result in the end.

This procedure, which is implemented in a quite compact way, generates all possible encounters. In Figure 5, I've reduced the problem to six players for the purpose of illustration. However, it is still not perfect, because it also evaluates symmetrical matches (Team A against Team B and vice versa) as new. Again, the event space is small, so this is not too much of an issue.

The brute binary generated by the `build` command

```
go build brute.go teams.go diff.go
perm.go data.go
```

launches itself at the `teams.yaml` YAML file, working its way through all the possible team constellations and outputting the best values based on the evaluation function. Figures 6 and 7 show two

examples of usable setups, one for the task posed at the beginning with a pool of six players and one for a game with two teams of eight.

In both solutions, the algorithm found distributions where each team has a goalie and the player skill values for offense and defense are roughly balanced. This will keep the match exciting until the last minute, assuming that all of the players put in a 100-percent performance, but that's a different story. ■■■

Info

- [1] Michalewicz, Zbigniew, and David B. Fogel. *How to Solve It: Modern Heuristics*. Springer, 2004, <https://link.springer.com/book/10.1007/978-3-662-07807-5>

[a	b	c	d	e	f]
a	b	c	-	d	e f
a	b	d	-	c	e f
a	b	e	-	c	d f
a	b	f	-	c	d e
a	c	d	-	b	e f
a	c	e	-	b	d f
a	c	f	-	b	d e
a	d	e	-	b	c f
a	d	f	-	b	c e
a	e	f	-	b	c d
b	c	d	-	a	e f
b	c	e	-	a	d f
b	c	f	-	a	d e
b	d	e	-	a	c f
b	d	f	-	a	c e
b	e	f	-	a	c d
c	d	e	-	a	b f
c	d	f	-	a	b e
c	e	f	-	a	b d
d	e	f	-	a	b c

Figure 5: All permutations of six players in teams of three.

```
$ ./brute
Team A
1.    joe [goa: 0  off: 3  def: 0 ]
2.    tony [goa: 3  off: 0  def: 2 ]
3.    fred [goa: 0  off: 0  def: 2 ]

Team B
1.    mike [goa: 0  off: 9  def: 0 ]
2.    manuel [goa: 9  off: 0  def: 0 ]
3.    mark  [goa: 0  off: 0  def: 9 ]

Diff: 81.00
```

Figure 6: A distribution determined by the brute force method.

```
$ ./brute
Team A
1.    zack [goa: 5  off: 3  def: 2 ]
2.    tan  [goa: 0  off: 2  def: 4 ]
3.    ryan [goa: 0  off: 8  def: 8 ]
4.    mike [goa: 0  off: 5  def: 1 ]
5.    simon [goa: 0  off: 7  def: 5 ]
6.    john [goa: 0  off: 9  def: 7 ]
7.    leo  [goa: 0  off: 5  def: 8 ]
8.    jack [goa: 0  off: 6  def: 6 ]

Team B
1.    chris [goa: 6  off: 3  def: 9 ]
2.    don  [goa: 0  off: 1  def: 5 ]
3.    alex [goa: 0  off: 8  def: 8 ]
4.    kevin [goa: 0  off: 4  def: 2 ]
5.    srini [goa: 0  off: 7  def: 6 ]
6.    max  [goa: 0  off: 7  def: 6 ]
7.    scott [goa: 0  off: 6  def: 6 ]
8.    andy [goa: 0  off: 6  def: 4 ]
```

Figure 7: The algorithm also does a good job with two teams of eight players each.



MakerSpace

Scoreboards and Video Routing in Python Scorekeeper

We look at a broadcast video system network that uses Python code to control a video router and check out another program that creates a scoreboard. *By Scott Sumner*

Each year my church hosts a basketball league, and several years ago we wanted to upgrade to digital scoreboards because our existing classic board was showing its age. The first version of this new scoreboard software was written in Python [1] and used the GTK toolkit to create the public display. A web page designed for an iPad allowed courtside control.

The gym is a shared space that also hosts a meeting room, party hall, and general-purpose room. To support these roles, two NUC small-form-factor computers drive the displays (100-inch LCD TVs).

Windows was necessary to support all of the display software so that PowerPoint and ProPresenter would run natively.

In general, a video system has a hub-and-spoke-style network wherein each device has a dedicated home that runs to a video switcher (Figure 1). The video switcher, as its name implies, accepts all of the video inputs and allows the operator to pick which one should be displayed, overlaid, or otherwise presented to the final output (Figure 2).

Displays are slightly more complicated because they need to select the switched video program from the video switcher or the scoreboards at any given time, which is accomplished by a matrix router. See the “Matrix Routers” box for more details.

Think of the router as a grid with sources (video signals – see the “Video Signals and Conversions” box) coming in on the left and going out along the bottom. Each column (an output or destination) is allowed one connection.

Matrix Routers

In the video world, a router is a really helpful device. It accepts some number of video signals in and can send them to any of its destinations. This example uses a small four-input by four-output version, but other versions are available with up to thousands of I/O points.

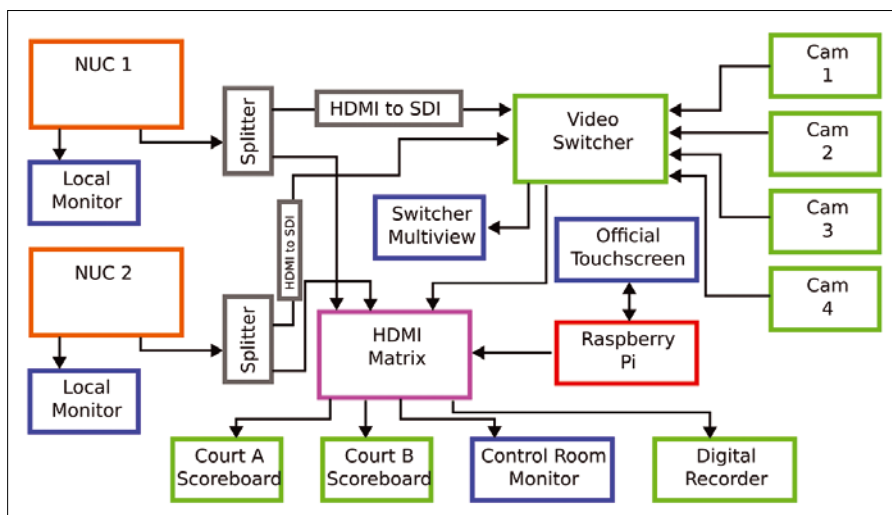


Figure 1: Video system signal flow. Displays shown in blue can be seen in the control room photo (Figure 2).



Figure 2: The video control room is a converted classroom. From left to right, the first two monitors in the background are the main displays of the NUCs running the scoreboard software. The text window is the output of the Python program, and the scoreboard itself is on the secondary monitor output. The third monitor is routable from the video matrix to select what is shown. In the foreground are the matrix control touchscreen and the camera controller. The tall monitor is the switcher multiview. The NUCs and the matrix are visible at the very bottom of the rack under the table.

Inputs can connect to as many outputs as needed. To say it another way, all of the outputs can display a single input, but each output can only display one thing at a time. When a selection is made, the selected input and output are connected as if they were wired together directly. These connections are easily changed throughout the day as the needs of the basketball game dictate.

Video Signals and Conversions

Many different video formats can carry live video down a wire. You're probably familiar with HDMI because it has been widely adopted since its introduction. The video system discussed here primarily uses a serial digital interface (SDI), which is a digital standard for broadcast video. Hardware converters between these formats also exist, so changing between them is as easy as connecting a cable of each type. The cost for this conversion is a slight delay in the video signal that is normally not noticeable; however, if you have stacked several conversions, the delays can start to add up and become visible. This scenario generally presents itself as audio out of sync with its associated video.

The Scoreboard

In addition to team points, a scoreboard also offers other game information, such as time remaining, period of play, time outs, or player numbers. Figure 3 shows the general flow of the program. At program startup, the graphics initialize and the initial state of the scoreboard renders. Concurrently, the web server starts and the operator brings this page up on a courtside computer (Figure 4).

As the basketball game progresses, the operator uses the web interface to update the game score and other statistics.

Each action on the web interface starts as a JavaScript function that runs in the browser, communicating with the CherryPy [2] server through background Ajax calls. When CherryPy receives these commands, it updates the graphics screen to reflect the change on the scoreboard itself and then sends the updated values back to the browser so that the operator's interface is also updated.

Listing 1 shows some of the code behind the scoreboard. (You can download the complete code online [3].) To begin, you must import the external libraries the program needs: The `pygame`, graphics library [4] draws the scoreboard; `_thread` allows multiple branches of the program to run at the same time, so you can run the scoreboard display and the web control interface at the same time; `pymysql` is the MySQL database library for Python; and `cherrypy` sets up the CherryPy framework.

The Web Class

The web class (lines 6-339) connects to a MySQL database that will be used to retrieve team names, determine which team plays which, and record game history. When the class is instantiated, the `__init__` function is called automatically to set up a few things needed later. Line 8 creates a class variable `self.scoreboard` and saves the reference to the scoreboard class that's passed in as an argument.

The `reconnect` function is really only one call split up across multiple lines for easier readability (lines 12-18). `pymysql.connect` creates `self.db`, the local reference to the database. Each line supplies the appropriate credential.

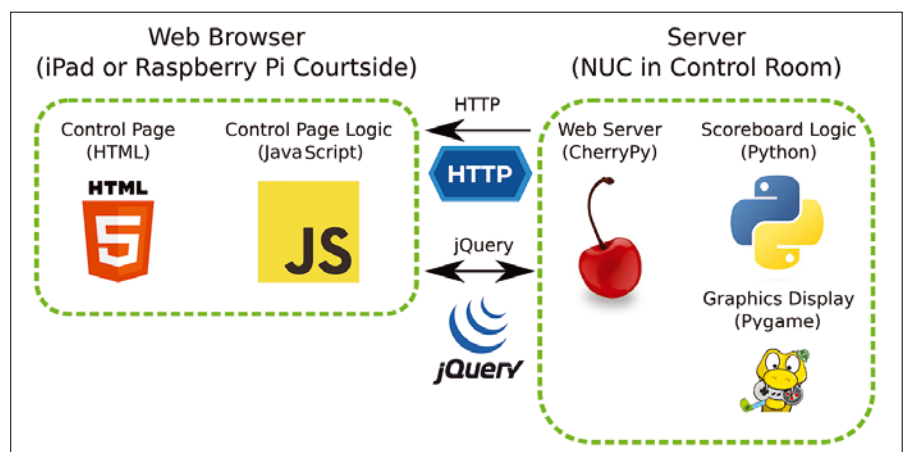


Figure 3: The servers, clients, displays, and their associated libraries communicate and interact with each other.

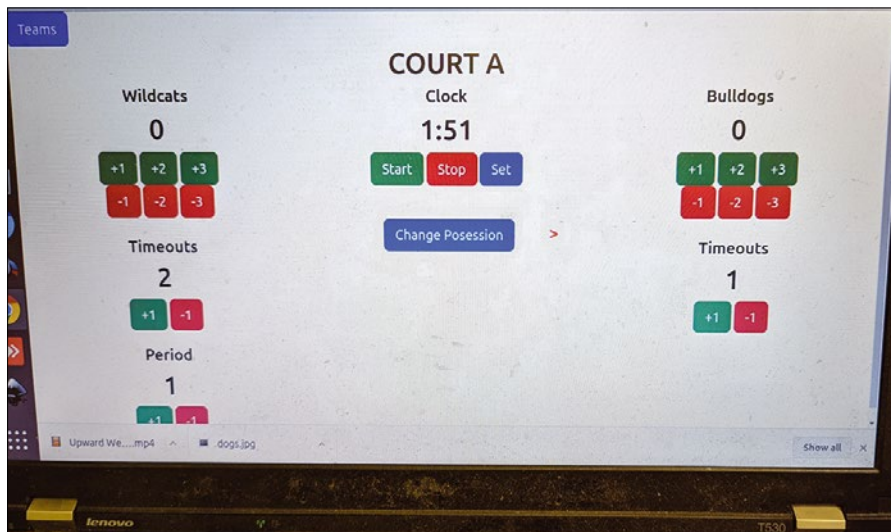


Figure 4: The scorekeeper's scoreboard interface.

Listing 1: scoreboard.py

```

001 import pygame
002 import _thread
003 import cherryypy
004 import pymysql
005
006 class web:
007     def __init__( self , scoreboard ):
008         self.scoreboard = scoreboard
009         self.reconnect()
010         self.dbGameID = None
011
012     def reconnect ( self ):
013         self.db = pymysql.connect(
014             host='DB_HOSTNAME',
015             user='DB_USER',
016             password = "DB_PASSWORD",
017             db='DB_NAME',
018         )
019
020     def gameEvent ( self , event , value ):
021         if self.dbGameID == None: return
022
023         self.db.ping ( True )
024         period = self.scoreboard.period.value
025         time = self.scoreboard.clk.seconds
026
027         sql = "INSERT INTO `updates` ( `gameID` , `period` ,
028             `gameTime` , `event` , `value` ) VALUES ( " + str
029             ( self.dbGameID ) + " , " + str ( period ) + " ,
030             " + str ( time ) + " , " + event + " , " + str
031             ( value ) + " );"
032
033         cursor = self.db.cursor ( pymysql.cursors.DictCursor
034             )
035         cursor.execute ( sql )
036         self.db.commit()
037
038     @cherryypy.expose
039     def index ( self ):
040
041         html = ""
042         ...
043         """ .format (
044             self.scoreboard.court.value + " " + self.
045             scoreboard.court.label,
046             self.scoreboard.scoreA.label,
047             self.scoreboard.scoreB.label,
048             self.scoreboard.scoreA.value,
049             self.scoreboard.scoreB.value,
050             self.scoreboard.clk.clockString,
051             self.scoreboard.timeoutA.value,
052             self.scoreboard.timeoutB.value,
053             self.scoreboard.possession.leftString,
054             self.scoreboard.possession.rightString,
055             self.scoreboard.period.value
056         )
057         return html
058
059     @cherryypy.expose
060     def teams ( self ):
061         cursor = self.db.cursor ( pymysql.cursors.DictCursor )
062         sql = "SELECT t1.teamName AS t1name , t2.teamName AS
063             t2name , games.* FROM `games` LEFT JOIN `teams` t1
064             ON t1.ID=games.team1 LEFT JOIN `teams` t2 ON t2.
065             ID=games.team2";
066         cursor.execute ( sql )
067
068         html = "<select id='gameSelect'>"
069         for game in cursor.fetchall():
070             html += "<option value='" + str ( game [ 'ID' ] ) +
071                 "'>" + str ( game [ 'date' ] ) + " | " + str
072                 ( game [ 'time' ] ) + " | " + game [ 't1name' ]
073                 + " | " + game [ 't2name' ] + "</option>"
074         html += "</select>"
075
076         return html
077
078     @cherryypy.expose

```

The `gameEvent` function (lines 20-31) logs any change to the scoreboard. In a future version, this function will allow for recovering the scoreboard after a crash and exporting a game report. To begin, the code checks whether `self.dbGameID` (which was initialized to `None` in line 10) is set. If not, then you don't have a database ID, so you just return.

The `self.db.ping` convenience function, with an argument of `True`, checks whether the database connection is live and, if not, reconnects automatically in the next line. The two lines that follow get the game period and time in seconds. These two values pinpoint a unique time within the game.

Listing 1: scoreboard.py (continued)

```

270 def processGameSelect ( self , gameID ):
271     cursor = self.db.cursor ( pymysql.cursors.DictCursor )
272     sql = "SELECT t1.teamName AS t1name , t2.teamName AS
          t2name , games.* FROM `games` LEFT JOIN `teams` t1
          ON t1.ID=games.team1 LEFT JOIN `teams` t2 ON t2.
          ID=games.team2 WHERE games.ID = " + str ( gameID ) ;
273     cursor.execute ( sql )
274     game = cursor.fetchone()
275
276     self.dbGameID = gameID
277
278     self.teamName ( "1" , game [ "t1name" ] )
279     self.teamName ( "2" , game [ "t2name" ] )
280
281     @cherryypy.expose
282     def teamName ( self , team , name ):
283         if team == "1": self.scoreboard.scoreA.label = name
284         if team == "2": self.scoreboard.scoreB.label = name
285
286     @cherryypy.expose
287     def score ( self , team1 , team2 ):
288         self.scoreboard.scoreA.value += int ( team1 )
289         self.scoreboard.scoreB.value += int ( team2 )
290
291         if team1 != "0": self.gameEvent ( "T1SCORE" , team1 )
292         if team2 != "0": self.gameEvent ( "T2SCORE" , team2 )
293
294         self.scoreboard.render()
295         return str ( self.scoreboard.scoreA.value ) + ":" +
          str ( self.scoreboard.scoreB.value );
296
297     @cherryypy.expose
298     def timeouts ( self , team1 , team2 ):
299         self.scoreboard.timeoutA.value += int ( team1 )
300         self.scoreboard.timeoutB.value += int ( team2 )
301
302         if team1 != "0": self.gameEvent ( "T1TIMEOUT" , team1 )
303         if team2 != "0": self.gameEvent ( "T2TIMEOUT" , team2 )
304
305         return str ( self.scoreboard.timeoutA.value ) + ":" +
          str ( self.scoreboard.timeoutB.value )
306
307     @cherryypy.expose
308     def period ( self , period ):
309         self.gameEvent ( "PERIOD" , period )
310
311         self.scoreboard.period.value += int ( period )
312         return str ( self.scoreboard.period.value )
313
314     @cherryypy.expose
315     def getClock ( self ):
316         return self.scoreboard.clk.clockString
317
318     @cherryypy.expose
319     def clockRun ( self ):
320         self.scoreboard.clk.running = True
321
322     @cherryypy.expose
323     def clockStop ( self ):
324         self.scoreboard.clk.running = False
325         self.gameEvent ( "CLOCKSTOP" , 0 )
326
327     @cherryypy.expose
328     def clockSet ( self , seconds ):
329         self.scoreboard.clk.seconds = int ( seconds ) + 1
330         self.scoreboard.clk.tick ( manual = True )
331         return self.scoreboard.clk.clockString
332
333     @cherryypy.expose
334     def possession ( self ):
335         self.scoreboard.possession.toggle()
336         if self.scoreboard.possession.leftString == "<":
337             self.gameEvent ( "POSSESSION" , 1 )
338         if self.scoreboard.possession.rightString == ">":
339             self.gameEvent ( "POSSESSION" , 2 )
340
341         return self.scoreboard.possession.leftString + ":" +
          self.scoreboard.possession.rightString
342
343     class clock:
344     def __init__ ( self , screen ):
345         self.screen = screen
346         self.width = self.screen.get_width()
347         self.height = 250
348         self.seconds = 6 * 60
349         self.clockSurf = pygame.surface.Surface ( (
350             self.width , self.height ) )
351         self.clockFont = pygame.font.Font ( "open24.ttf" ,
352             self.height )
353         self.running = False
354
355         mins = int ( float ( self.seconds ) / 60.0 )
356         secs = self.seconds % 60
357
358         self.clockString = "{0}:{1:02d}".format ( mins ,
359             secs )
360
361     def tick ( self , manual = False ):
362         if self.seconds > 0 and ( self.running == True or
363             manual == True ):
364             self.seconds -= 1
365
366         mins = int ( float ( self.seconds ) / 60.0 )
367         secs = self.seconds % 60
368
369         self.clockString = "{0}:{1:02d}".format ( mins , secs )
370
371     def render ( self ):
372         timeSurf = self.clockFont.render ( self.clockString ,
373             True , ( 0 , 255 , 0 ) )
374
375         x = self.width / 2 - int ( timeSurf.get_width() / 2 )
376         self.clockSurf.fill ( ( 0 , 0 , 0 ) )
377         self.clockSurf.blit ( timeSurf , ( x , -25 ) )
378         return self.clockSurf

```

Listing 1: scoreboard.py (continued)

```

372
373 class possession:
374     def __init__ ( self ):
375         self.width = 300
376         self.height = 100
377         self.font = None
378         self.direction = "<"
379         self.posSurf = pygame.surface.Surface ( ( self.width
380             , self.height ) )
381         self.leftString = ""
382         self.rightString = ""
383         self.makeSurfaces()
384     def makeSurfaces ( self ):
385         self.leftSurf = self.font.render ( "<" , True ,
386             ( 255 , 0 , 0 ) )
387         self.rightSurf = self.font.render ( ">" , True ,
388             ( 255 , 0 , 0 ) )
389     def toggle ( self ):
390         if self.direction == "<":
391             self.direction = ">"
392             self.leftString = ""
393             self.rightString = ">"
394         else:
395             self.direction = "<"
396             self.leftString = "<"
397             self.rightString = ""
398     def render ( self ):
399         self.posSurf.fill ( ( 0 , 0 , 0 ) )
400         label = self.font.render ( "POSSESSION" , True ,
401             ( 255 , 0 , 0 ) )
402         x = self.width / 2 - int ( label.get_width() / 2 )
403         self.posSurf.blit ( label , ( x , 0 ) )
404         if self.direction == "<": self.posSurf.blit ( self.
405             leftSurf , ( 0 , 0 ) )
406         else: self.posSurf.blit ( self.rightSurf , ( self.
407             width - self.rightSurf.get_width() , 0 ) )
408         return self.posSurf
409 class label:
410     def __init__ ( self ):
411         self.width = 250
412         self.height = 350
413         self.label = ""
414         self.value = ""
415         self.labelFont = None
416         self.valueFont = None
417         self.labelColor = ( 200 , 200 , 200 )
418         self.valueColor = ( 200 , 200 , 200 )
419         self.countSurf = pygame.surface.Surface ( ( self.
420             width , self.height ) )
421     def render ( self ):
422         self.countSurf.fill ( ( 0 , 0 , 0 ) )
423         valSurf = self.valueFont.render ( str ( self.value )
424             , True , self.valueColor )
425         x = self.width / 2 - int ( valSurf.get_width() / 2 )
426         self.countSurf.blit ( valSurf , ( x , 0 ) )
427         labelSurf = self.labelFont.render ( str ( self.label
428             ) , True , self.labelColor )
429         x = self.width / 2 - int ( labelSurf.get_width() / 2 )
430         self.countSurf.blit ( labelSurf , ( x , valSurf.get_
431             height() - 10 ) )
432     return self.countSurf
433 class counter:
434     def __init__ ( self ):
435         self.width = 250
436         self.height = 350
437         self.label = ""
438         self.value = 0
439         self.labelFont = None
440         self.valueFont = None
441         self.labelColor = ( 200 , 200 , 200 )
442         self.valueColor = ( 200 , 200 , 200 )
443         self.countSurf = pygame.surface.Surface ( ( self.
444             width , self.height ) )
445     def render ( self ):
446         self.countSurf.fill ( ( 0 , 0 , 0 ) )
447         valSurf = self.valueFont.render ( str ( self.value )
448             , True , self.valueColor )
449         x = self.width / 2 - int ( valSurf.get_width() / 2 )
450         self.countSurf.blit ( valSurf , ( x , 0 ) )
451         labelSurf = self.labelFont.render ( str ( self.label
452             ) , True , self.labelColor )
453         x = self.width / 2 - int ( labelSurf.get_width() / 2 )
454         self.countSurf.blit ( labelSurf , ( x , valSurf.get_
455             height() - 10 ) )
456     return self.countSurf
457 class board:
458     def __init__ ( self ):
459         pygame.display.init()
460         self.screen = pygame.display.set_mode ( ( 1280 , 720 ) ,
461             display = 1 , flags = pygame.FULLSCREEN )
462         pygame.font.init()
463         self.scoreFont = pygame.font.Font ( "font.ttf" , 200 )
464         self.labelFont = pygame.font.Font ( "font.ttf" , 50 )
465         self.timeoutFont = pygame.font.Font ( "font.ttf" , 50 )
466         self.posFont = pygame.font.Font ( "font2.otf" , 50 )
467         self.logo = pygame.image.load ( "upwardBlack.jpg"
468             ).convert()
469         self.clk = clock ( self.screen )
470     ...

```

Line 27 builds a SQL statement. The `INSERT INTO` command names the table followed by a list of columns separated by commas. The `VALUES` keyword assigns the values for each column, provided in the same order. Line 29 creates a database cursor that interacts with an SQL statement. In this case (line 30), it just executes the SQL line created in line 27, but it has many more capabilities, especially when retrieving records. Line 31 calls `commit` to confirm that you want to write the data.

The `cherrypy` decorator (lines 33-339) creates a small web server for controlling the scoreboard. Each Python function becomes a web address that returns its designated content to the browser that has called it.

The `index` function acts just like `index.html` in a traditional web server. In the absence of another address, it is the default item returned. Most of this function is a very large multiline string (not shown here, grab the full code online [3]) enclosed by triple quotes (""") that instructs Python to ignore any newlines or other special characters until it encounters another triple quote. The enclosed string is the HTML5 [5] and JavaScript of the control web page. After the massive text string, the Python `format` command inserts all of the current variables into the web page (lines 240-251).

The `@cherrypy.expose` decorator tells CherryPy that it should allow this function to be reachable through its web server. Without this decorator the function remains private from anyone on the web, which allows the program to be structured with additional functions as needed, with only those specifically designed to be web-accessible published. You'll see this decorator before each function in the `web` class, so it's only described once here.

The `teams` function (lines 257-267) is somewhat of a misnomer in that it generates a game selector. For the purposes of this database, a game is a pair of teams at a specific date and time on a specific court. This function retrieves all of the games and returns an HTML `select` widget.

The function begins by creating an SQL cursor and statement and then executing it (as in lines 29-31). However, now it's retrieving records, so the cursor has the results of the query, which is

data instead of just a message that the query succeeded.

After starting the HTML `select` widget on line 262, the program loops over the SQL results. As the name implies, `cursor.fetchall` gives all of the results as an iterator. `game` will contain one row from the database for each time through the loop. Each pass creates an HTML `option` and lists the game date, time, and teams.

The `processGameSelect` function (lines 270-279) is called when the scoreboard operator selects a game from the dropdown just generated. It receives the `gameID` argument and uses that to get the game details from the database. Line 272 is an almost identical SQL statement to line 259, but with a `WHERE` clause added to the end with the `gameID`.

After doing the SQL dance one more time, line 274 calls `cursor.fetchone`. The last time, *all* of the records were retrieved, but this time only one is needed, which is saved into `game`. `gameID` is saved into a class variable, and `self.teamName` sets the names retrieved from the database.

The `teamName` function (lines 281-284) accepts a team number and name as arguments. Each of two `if` statements determines which team is being named and then saves the new name to a variable in the `scoreboard` class.

Housekeeping

The `score`, `timeouts`, and `period` functions (lines 286-312) accept the `team1` and `team2` arguments (`period` only accepts a period number) which are the amounts to change each value. Negative values are allowed so that values can be corrected or reset for the next game or period. Note also that the variables are preceded with `int`. All arguments from CherryPy come in as strings, so they have to be converted to integers before they can be used algebraically.

Finally, the two newly adjusted values are returned, separated by a colon, to go back to the JavaScript function and be split into the two scores so that the control screen is updated properly.

The JavaScript versions of these functions are generated in the `index` function (not included in the listing). The `score` function, for example, creates a JavaScript object and adds `obj.team1` and `obj.team2`:

```
function score ( team1 , team2 )
{
  obj = new Object();
  obj.team1 = team1
  obj.team2 = team2
```

These are the score deltas (amounts to change) for each team. jQuery [6] (represented by the `$`) then posts the object to the address `score` (the Python CherryPy function described above):

```
$.post (
  "score" , obj , function ( data ) {
    scoreParts = data.split ( ":" );
    $ ( "#team1score" ).html (
      scoreParts [ 0 ] );
    $ ( "#team2score" ).html (
      scoreParts [ 1 ] );
  } );
```

`obj` has the values to send that were set up earlier. When `post` finishes and receives a response, `function (data)` is called, where `data` is the returned string with the newly updated scores (or timeouts or period). First, the data is split into its two parts with `data.split`, and then the score on the control page is updated with jQuery.

Time and Possession

The clock is a unique case because it is the only game variable that updates independently. Everything else is a response to something that happens on the court.

In the clock function (lines 314-331), `getClock` returns a string that represents the current game time, called by a recurring interval in a JavaScript function on the operator's web page to update the operator's clock. A timer event in the graphics thread both updates the internal clock variable and redraws the clock on the scoreboard display.

`updateClock` is only found in the JavaScript and does the job of keeping the control page clock up to date. It calls `getClock` by jQuery to get the current clock time then updates the HTML `div` with the current value:

```
function updateClock()
{
  $.post ( "getClock" ,
    function ( data ) {
      $ ( "#clock" ).html ( data );
    } );
}
```

Both the `clockRun` and `clockStop` functions change the state of `self.scoreboard.clk.running`. If that variable is `True`, the clock is updated. If it's `False`, the clock is not updated or is, for all practical applications, paused. `clockStop` also calls `self.gameEvent` to add an entry that the clock was stopped. Because the clock is not team-specific, the second argument is `0`. The JavaScript version of these functions (lines 80-92) are not shown. Once the Python function is called by jQuery, a JavaScript interval is either set or cleared to update the control page.

For the `clockSet` function, the variable `self.scoreboard.clk.seconds` keeps track of the game time remaining in seconds, which is set to one second more than requested. Next, `self.scoreboard.clk.tick` automatically subtracts one second. Passing `manual = True` forces a redraw of the clock even though it is not running. Finally, `self.scoreboard.clk.clockString`, which is a text representation of the time remaining in minutes and seconds, returns.

The JavaScript version can be found on lines 101-116 (not shown). It asks the operator what the clock should be set to and will contain either the requested time or `null` if the dialog box was canceled.

If the value is valid, the request is split into minutes and seconds and then converted to seconds. After a new object is created and a `seconds` parameter added, it is posted by jQuery back to the Python function. The control screen then updates with the new clock value and the update interval clears so clock requests are not made until it starts running again.

The `possession[sic]` function (333-339) calls the `toggle` method of `self.scoreboard.possession`, which flips which team currently has the ball (indicated by the `<` and `>` symbols as arrows).

Lines 336 and 337 check to see which team has possession and add a `gameEvent` to reflect the change. Finally, two strings are returned separated by a colon; each string is either a space or one of the `<` or `>` symbols. The JavaScript version is on lines 71-78 (not shown). After posting a possession change by jQuery the return string is split into two parts, and the left and right divs are updated on the control screen.

The Clock Class

The `clock` class (lines 341-371) manages the game clock for the main graphics (scoreboard) display. The `__init__` function sets up a few things: `self.screen` is a reference to the screen object passed when creating the class; `self.width` stores the screen width by calling `get_width()` on the screen surface; `self.height` is set explicitly to define how tall you want the clock to be displayed; and `self.seconds` is the time on the clock, initialized to `6 * 60` seconds to reflect six-minute periods. Line 347 creates a Pygame surface onto which the clock is drawn, and line 348 sets up the font. Finally `self.running` tracks whether the clock is running or paused.

Lines 351-354 calculate the clock values and convert to `int`; `seconds` is the mod (remainder) of `self.seconds / 60`. Finally, `self.clockString` is assembled from the calculated values.

The main loop calls the `tick` function (lines 356-363) once a second. If time is left on the clock and either `self.running` or `manual` is set to `True`, `self.seconds` decrements by one, and the same calculations as before are repeated to update `self.clockString`. The `manual` parameter used when setting the clock forces the clock to be regenerated immediately rather than waiting for the next automatic call to `tick`.

The `render` function (lines 365-371) draws the clock. `self.clockFont.render` takes `self.clockString` and a color tuple as arguments. The `True` argument says to anti-alias the text as it is drawn. The result is `timeSurf`, a Pygame surface with the current game time drawn onto it.

After the center point of `timeSurf` is calculated, `self.clockSurf` is cleared by filling it with black and blitting (copying) `timeSurf` onto it. Then the surface is returned. This copying might seem like an extra step, but here's what's happening: When `self.clockFont.render` generates the text, the surface is the exact size of the text it generated. For it to fill the space at the dimensions specified in `__init__`, you have to copy it onto the larger surface.

The Possession Class

The `possession` class (lines 373-406) indicates who has the ball at any given time. `__init__ (self)` sets up the graphical display, with the initial state

specified by `self.direction` (who currently has the ball). A surface is created and initialized with `self.leftString` and `self.rightString`.

The convenience function `makeSurfaces` creates `self.leftSurf` and `self.rightSurf` possession arrows rendered with `<` and `>` symbols.

When the `toggle` function (lines 388-396) is called, possession has changed from one team to the other. The `if` checks to see which team currently has possession and then updates all of the variables to their opposite states. Note that `self.leftString` and `self.rightString` either have an arrow or an empty string.

When everything is ready to be drawn in the `render` function, `self.posSurf` is cleared by filling it with black (line 399) before generating the label text, calculating the center point, and blitting it onto the final surface.

Lines 404 and 405 make sure the arrow is blitted on the appropriate side of the surface. If `self.direction` is `<`, the arrow is drawn on the left side of the surface; otherwise, it's drawn on the right before the final surface is returned.

Labels and Counters

The `label` class (lines 408-430) represents an arbitrarily labeled value on the scoreboard. The `label` and its `value` can have separate fonts and colors. The Pygame surface is created on line 418.

The `render` function starts by clearing the surface (`fill`) with black then rendering the `value`, calculating its center point, and drawing it. Lines 426-428 do the same thing for the label text before returning on line 430.

Lines 432-454 describe the counter class. Counters are identical to labels, except their `self.value` is expected to be an integer.

The Board Class

Everything comes together in the `board` class (lines 456-538). All of the modules that have been defined to this point are initialized here, starting with `pygame.display.init`, which starts the Pygame engine. The `pygame.display.set_mode` creates the drawing surface, setting the window size. `display = 1` makes the Pygame window default to the second monitor, and `flags = pygame.FULLSCREEN` is self-descriptive.

Because this program uses Pygame's font capabilities, these are initialized on line 460, after which all of the fonts are loaded. In Pygame a font is the face itself and an associated size, so the same font file may be loaded multiple times with different sizes.

Line 466 loads `self.logo`, which is the logo displayed along the bottom of the scoreboard. The `convert` at the end of the line makes the image's internal format match Pygame's initialized display format so the logo blit onto the screen is faster.

Line 468 creates an instance of the `clock` class, with a reference to `self.screen` passed in so that it can draw the updated clock directly.

The final lines of the program (not shown) initialize five different instances of the counter class for the home and visitor's scores, number of timeouts for each team, and the game period. In each case, `valueFont` and `labelFont` are set to one of the font/size pairs initialized earlier. A `label` indicates which court each scoreboard is tracking.

A Typical Game Day

When the crew arrives on game day, the first task is to set up the scoreboard operator's console (two Raspberry Pis, one per scoreboard). Once that's done, robotically controlled cameras are placed around the court and everything is tested. The youngest teams play first in a half-court model. Halfway through the day, play switches to full-court games (Figure 5).

At halftime of each game, video messages play onto the screens, and the scoreboard returns to normal operation (Figure 6). At the halfway point of the day one scoreboard system is routed to both score displays because a single court is now in use. This switching is accomplished with a matrix video router.

Controlling a Video Matrix

Although the basic operation of a matrix router is fairly simple to understand, it's always easier to have meaningful names and labels attached instead of remembering sets of numbers. Adding some shortcuts to common configurations makes things even easier, which is the purpose of the matrix control program (not shown, but available online [3]).

The Raspberry Pi communicates with the matrix over a USB serial connection set up in Python:

```
class matrixSerial:
    def __init__( self ):
        self.port = serial.Serial ( 2
            "/dev/ttyUSB0" , 9600 )

    def send ( self , cmd ):
        self.port.write ( cmd )
```

The `__init__` line creates `self.port`, which is an instance of the `Serial` object with the name of the serial device and the requested baud rate. The defaults are

fine for everything else. The `send` function takes in `cmd` and transmits it with a call to `self.port.write`, thus sending data to the matrix.

The Button Class

The main function of the program is to present easy options to the user on a touchscreen; a button class sets up the actions to button presses. When the class is instantiated, it receives `screen`, which is a reference to the Pygame screen object, and `font`, which is a Pygame font object to be used to label the buttons. Each of these is stored in class properties for use by other methods.

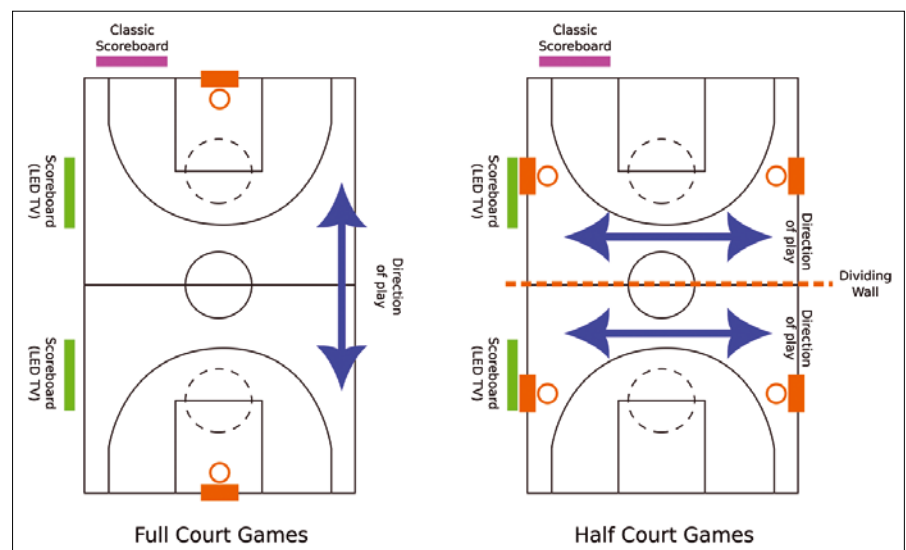


Figure 5: Depending on the age group playing, the game may be on full or half courts. Each scoreboard must be routed to follow the current court configuration.



Figure 6: The scoreboard display as seen on the large, audience-facing monitors.

When the create function is called, it receives a number of variables to get started, including the x and y screen coordinates where the button should be drawn, the width and height of the button, and the label text displayed on the button. `self.rect` stores the Pygame rect object, which has functions to check whether coordinates are within its bounds, along with several other useful utilities, and `self.matrixCommands` and `self.osCommands` are the actions taken when the button is pressed.

The `addOsCommand` function takes its associated string argument and passes it to the operating system when the button is pressed:

```
def addOsCommand ( self , cmd ):
    self.osCommands.append ( cmd )
```

The `addMatrixCommand` function is the same, except it transmits to the matrix.

The `render` function is responsible for drawing the button on the screen. As you have seen before, it creates a surface and fills it – with green in this case. If its label is not empty, the program renders the text, calculates how to center the text, and adds the text to the button `blit`. The surface is then drawn to the screen with a reference saved to its Pygame `rect`.

The Matrix Class

The `matrix` class draws and manages the interface on the touchscreen. As usual, an `__init__` function initializes the different components of the interface, but this example has a lot of them. The graphics system starts and creates a window to draw on. Although the window has a caption, it is generally not

visible because the window is in fullscreen mode.

As usual, the fonts and serial communications are initialized, although `pygame.font.SysFont` has an empty string as its first argument, thus asking Pygame for the default font:

```
buttonFont = pygame.font.SysFont ( "", 32 )
self.matrixPort = matrixSerial()
```

The next line creates `self.buttons` as a list, which is where each of the buttons added to the interface are stored. Most of the rest of the `init` function creates each of the buttons and defines their functionality. They all follow the general format:

```
btn = button (self.screen, buttonFont)
btn.create ( <x> , <y> , <width> , <height> , "<label"> )
btn.addMatrixCommand ( chr ( 0x05 ) + chr ( 0x55 ) + chr ( 0x19 ) + chr ( 0x00 ) )
btn.addMatrixCommand ( chr ( 0x05 ) + chr ( 0x55 ) + chr ( 0x19 ) + chr ( 0x11 ) )
self.buttons.append ( btn )
```

The first line creates a button and passes in `self.screen` and `buttonFont`. The `create` makes the button, and `addMatrixCommand` adds commands for the video matrix. The command string comes from the matrix documentation; it tells you to send the first three hex values to start



Figure 7: The switcher multiview shows all of the cameras and other sources available. They can be put “on air” by clicking on them.

Table 1: Buttons and Configurations

Line Nos.	Button	Action
57-61	Separate Scoreboards	Each display shows its own scoreboard computer.
63-67	Single Scoreboard	Both screens display the first scoreboard computer.
69-73	Both Program	Both screens display the video switcher program output.
75-78	Court A Program	Switch only the court A screen to the video switcher output.
80-83	Court B Program	Switch only the court B screen to the video switcher output.
85-88	Court A Scoreboard A	Switch only the court A screen to its scoreboard computer.
90-93	Court B Scoreboard B	Switch only the court B screen to its scoreboard computer.
95-98	CR Gym 1 Action	View Scoreboard (NUC) 1 on the control room monitor.
100-103	CR Gym 2 Action	View Scoreboard (NUC) 2 on the control room monitor.
105-108	CR Program Action	View the video switcher program output on the control room monitor.
110-113	Record	Start recording onto an SD card or flash drive.
115-118	Stop	Stop recording.

the command. The last byte contains the routing as 0 indexed addresses. The high-order digit defines the output or destination, and the low-order digit defines the input or source. Here, 0x00 says “connect the first output to the first input.” The next line connects the second output to the second input. Table 1 shows the configuration for each button.

One output of the matrix is connected to a monitor in the control room so that the sources can be easily monitored. The three CR buttons allow any of the matrix sources to be viewed on the control room monitor (Figure 7).

All of the buttons except the last two use `addMatrixCommand`. The Record and Stop buttons use `addOsCommand` instead and then use `wget` to trick the digital recorder into thinking the buttons on its web GUI are being pressed to start and stop the recording onto an SD card or flash drive (Listing 2).

The `render` function iterates over `self.buttons`, calling each `render` method and creating all of the buttons onscreen, as well as all the Pygame `rect` objects to see whether they’ve been clicked. Once everything has been drawn to the buffer, the function calls `pygame.display.flip` to make it visible in the drawing window.

The `loop` function makes everything interactive by watching for buttons

presses, sending commands when they are, and keeping up with everything that’s happening.

The `looping` value is first set to `True`, before the function enters the `while` loop with `looping` as its argument. If `looping` is set at any point to `False`, the loop exits.

The function then waits for Pygame events and checks to see if the event is a button click. To see if any of the buttons have been pressed, the function loops over `self.buttons` again and gets the coordinates of the mouse click:

```
for btn in self.buttons:
    if btn.rect.collidepoint (
        event.pos ) == True:
        for cmd in btn.matrixCommands:
            self.matrixPort.send (
                cmd + chr ( 0x77 ) )
```

If `True`, a button has been pressed. To make the button do its thing, the program loops over `btn.matrixCommands` and sends each string to the matrix. To give the matrix time to process each command, the function waits half a second before moving on.

The lines that follow do the same thing for any operating system commands (`btn.osCommands`) but pass them to `os.system` instead. The call to `matrix` on the last line of the program sets everything in motion.

Conclusion

I hope this article has given you some insight into a different type of network. When you think about the idea that similar control rooms like this exist all over the world and they each feed their own set of displays or a broadcast video system network, you can see the similarities and differences between the Internet and IP routing. You’ve also seen how these scoreboards are generated and controlled to support a local basketball league. Now that all the games and the season are done, it’s time to hit the showers. ■■■

Info

- [1] Python: www.python.org
- [2] CherryPy: <https://docs.cherrypy.dev/>
- [3] Code for this article: <https://linuxnewmedia.thegood.cloud/s/5Rzx9tQW2FJ6N3Z>
- [4] Pygame: <https://pygame.org>
- [5] HTML5: <https://www.w3docs.com/learn-html/html5-elements-reference.html>
- [6] jQuery: <https://www.jquery.com>

Author

Scott Sumner has worked in the museum and nonprofit industry for most of his professional career. He enjoys exploring technology solutions with Arduinos, Raspberry Pis, microcontrollers, and Linux systems.

Listing 2: wget Trick

```
btn.addOsCommand ( r'wget --header="Accept:*/*" --header="Accept-Encoding: gzip, deflate" --header="Accept-Language:
en-US,en;q=0.9" --header="Connection: keep-alive" --header="Cookie: serenity-session=72952074"
--user-agent="Mozilla/5.0 (X11; Linux x86_64; rv:60.0) Gecko/20100101 Firefox/60.0" "http://172.16.2.20/
cgi-bin/system.cgi?command=recording&action=start" )
```

Linux Magazine Subscription

Print and digital options
12 issues per year



► SUBSCRIBE
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



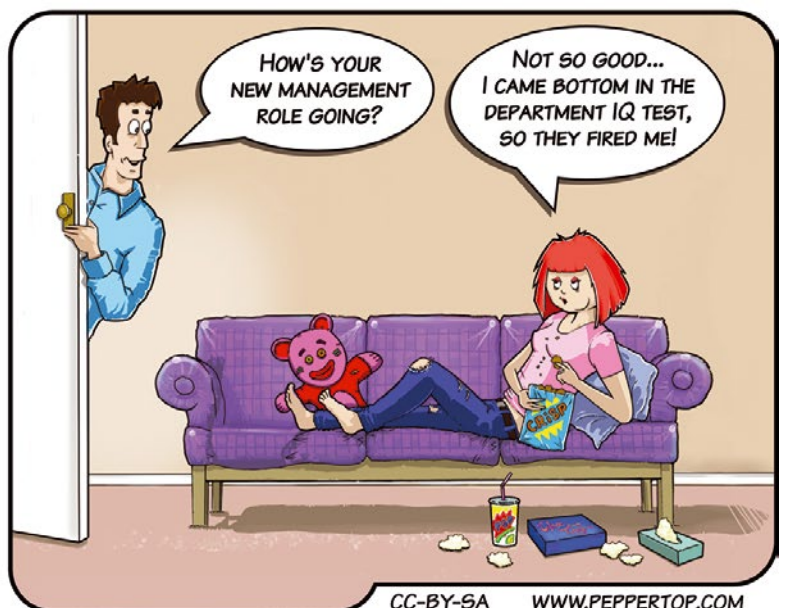
The free tools of the Linux environment bring the power of the professional down to everyday users. Experts pay hundreds of dollars for sophisticated photo editing tools, but in Linux, the answer is only a download away. RawTherapee is a tool for processing raw photo data. Before a photo gets to be a JPEG image, it is in a raw form that consists of data dumped from the camera’s sensors. The camera itself “interprets” that data to make a JPEG or other image form. But what if you want to interpret it differently? A raw image processor starts with the image data in its most fundamental form, letting you transform the image in ways that aren’t possible with an ordinary photo editor. In this month’s Linux Voice, we introduce you to the RawTherapee image processor. Also inside: We take a walk with fdupes, a simple tool that helps you find duplicate files and directories.



Image © Alexandr Moroz, 123RF.com

LINUXVOICE ▶

Doghhouse – AI	77
<i>Jon “maddog” Hall</i>	
Earlier technologies have persisted despite government regulations – so will AI.	
Fdupes	78
<i>Ferdinand Thommes</i>	
The command-line tool fdupes helps you find duplicate folders and directories.	
FOSSPicks	82
<i>Graham Morrison</i>	
This month Graham looks at wallabag, Read It Later, killport, F3D, Tenacity, Cataclysm: Dark Days Ahead, botany, and more!	
Tutorial – RawTherapee Workshop	88
<i>Anna Simon</i>	
The current RawTherapee version finally adds selective image editing, among other long-desired features, to help it compete with king of the hill, darktable.	



CC-BY-SA WWW.PEPPERTOP.COM

Hone your skills with special editions!

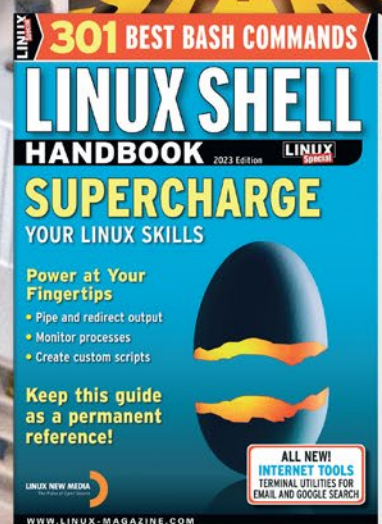
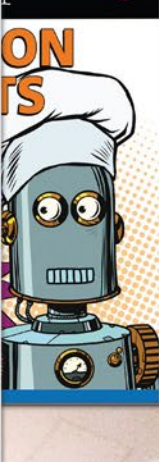
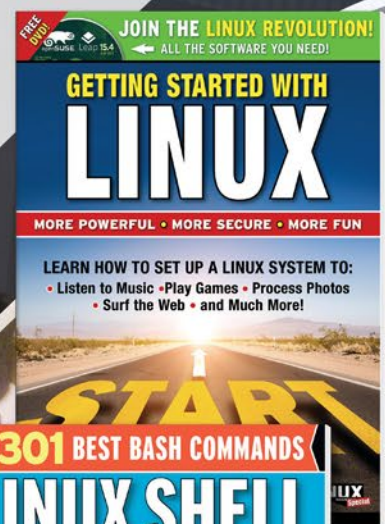
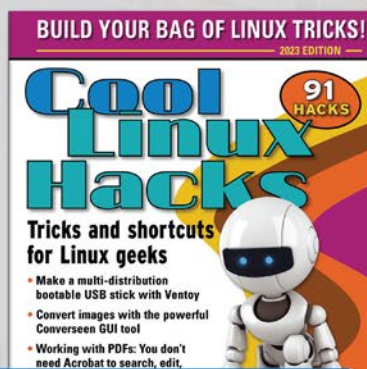
Get to know Shell, LibreOffice, Linux, and more from our Special Edition library.



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



MADDOG'S DOGHOUSE

Earlier technologies have persisted despite government regulations – so will AI. BY JON “MADDOG” HALL

Not Intelligent

I have been writing articles for Linux magazines for over two decades. Some articles were very technical, some were business-oriented, and some had to do with the philosophy of free and open source versus closed source in producing systems.

However, recent movements by various governments to “limit AI” have caused me to write an opinion article this month describing why this limiting will not work.

I look back in time to other “limiting” exercises, such as embargoes. Do we really think that countries that are under embargo cannot get the hardware and software they need to do whatever they want to do with AI, either by smuggling it out of some country or relying on one of their “partners” to break the embargo?

I remember the limitations on encryption back in the day. The United States has such arcane rules about shipping encryption out of the country that even if you bought a box of encryption software from Canada, never unwrapped it, and tried to send it back, it was cause for the Feds to come after you. We tried to tell the government that all of the best cryptologists were leaving the USA for places where they could sell their skills, but no use – the government would not listen.

Now it is artificial intelligence (AI). The mighty and the wealthy are coming together to tell the software community that the government should license and control the use of AI as if no one had seen even one movie about AI. As if you need a computer the size of a room in order to do AI. As if these titans of industry could somehow put locks on every mind who has ever thought about AI.

First there is the issue that most people cannot even define “AI.” They think computers will *never* become intelligent enough to do *their* job. That is their first mistake. If you are talking *real* AI, then the computer (or robot) will not only be able to do your job, but it will do the job faster and better than you, with no time off to sleep or eat – 24 hours a day, 365 days a year. And the computer/robot will not only be able to do your job, but it will build computer/robots like it or better than itself to do your job in parallel.

I have seen AI go from a concept in science fiction to where it is today. For many years, we underestimated the amount of

CPU, memory, data, and other factors needed to duplicate human thought and consciousness. Most people still underestimate those factors, possibly by several orders of magnitude, yet through a combination of massive data farms, huge amounts of processing power, and methods of programming, we can automate a lot of the filtering that can bring us closer to what appears to be intelligence.

I am one of those people who think the human mind is a sophisticated electro-chemical machine. Neurons (about 86 billion, +/- 8 billion) supply “storage” and are connected by synaptic connections (7,000 synaptic connections per neuron to other neurons) – sparks of energy that fire off the 5 to 50 “messages” that each neuron sends per second to other neurons in ways that we still are exploring. In an adult, these synaptic connections measure in the trillions.

There is a certain amount of “programming” built into us through eons of evolution, but a baby is still in some ways a blank slate and is influenced and taught how to think – to grow those synaptic connections.

Yes, we are amazed at what ChatGPT can do, but remember that many of these examples have been shown to be far from “intelligent.” They come from datasets which may not be complete, and which have been shown to be inaccurate in ChatGPT’s answers. These examples often need context which is outside the dataset being used by ChatGPT, even as huge as it is.

All of that having been said, someday AI will be “intelligent,” and no laws will stop its exploitation or misuse.

The governments and companies that try to limit its development will only force other governments and companies who want to “get ahead on AI” to go underground. We will have an AI race just as we had a space race and a nuclear arms race years ago. Except you do not need to shoot off rockets or ship uranium to develop AI.

We are no longer in the realm of computer science where you need billions of dollars and government support to build a system. Almost any individual could fund the development of an AI system, just as they wrote their own encryption code years ago. Each developer will justify their development outside of the law.

The only way to win the war of AI is to engage in it openly. Then you may be ready when AI comes to your doorstep. ■■■



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®.

Detect duplicates with fdupes

Double Trouble

The command-line `fdupes` tool helps you find duplicate folders and directories.

BY FERDINAND THOMMES

Hard disks have the unpleasant tendency of filling up faster than expected. It is not always immediately obvious why. Keeping things tidy should not be underestimated in this context. Untidy, poorly organized hard disks tend to fill up faster than well-organized ones. Because life is a mixture of order and chaos, most users probably face this problem.

The unexpectedly high utilization level of hard disks is often caused by duplicate files. The typical candidates are photos, music, or videos, which can quickly occupy several gigabytes of space and are often difficult to find. There are several graphical applications on Linux to help you detect and remove duplicates like this, and there are several more for the command line.

GUI or CLI?

Well-known tools with a graphical interface for a cleanup include FSInt and dupeGuru. In this article, I will look at `fdupes` for the command line [1], first released in 2000. Most distributions include the tool, which weighs in at just over 100KB, in the archives; you can install using your distribution's choice of package manager. Listing 1 shows a guide for Debian, Fedora, and Arch Linux.

The current 2.2.1 version from September 2022 has not made its way into all repositories [2]. If you want to compile `fdupes` from the source code, you can use the tarball from GitHub. After unpacking, just follow the familiar three-step process of `./configure`, `make`, and `make install`. As of `fdupes 2.0`, there are two dependencies that you may also need to resolve yourself, depending on the distribution. To do this, follow the instructions in the `INSTALL` file from the unpacked archive.

After the install, you can use the tool immediately without any configuration. It identifies duplicate files in the specified directories in several steps. The file name is not important for detection as a duplicate. Instead, two files must first be the same size; given this, `fdupes` compares their MD5 checksums.

Listing 1: Installing fdupes

```
##### Debian and derivatives:
$ sudo apt install fdupes
### Fedora:
$ sudo dnf install fdupes
##### Arch Linux and derivatives
$ sudo pacman -S fdupes
```

Finally, the software performs a byte-by-byte comparison, to make sure that it is definitely the same file.

`Fdupes` has numerous options that let you control the search and the subsequent deduplication. Initially, you will want to familiarize yourself with the tool by running the `fdupes --help` command. This will help you identify the options that suit your use case.

Test Run

For the test, I created an `fdupes` directory in the `Documents` directory and then created 10 text files whose content read `fdupes finds and removes duplicates`. Listing 2 shows you how to do this quickly.

A following `ls -l` confirms that the files were created. The easiest way to search for duplicates in the new directory is to use the `fdupes ~/Documents/fdupes` command (Figure 1). By separating the paths with spaces, you can specify multiple directories at the same time. To search recursively in directories, you need to use the `-r` option, as in `fdupes -r ~/documents` (Figure 2). In this case, the tool finds my 10 text files along with some other duplicates. Use the `-r` option to specify the path of subdirectories you want to include.

The `-s (--size)` options shows you the size of the hits. You can use `-t` or `--time` to find out when a file was last modified. `-G` or `--minsize=SIZE` and `-L` or `--maxsize=SIZE` lets you further narrow down the selection.

Be Careful When Removing

But finding is only the first part of the task; after all, we want to delete duplicates to clean up the hard disk. This is where the `--delete` option comes in. When using `-d`, always make sure that your path specification is correct – files deleted with `fdupes` cannot be recovered. The command

```
fdupes -d ~/documents/fdupes
```

Listing 2: Create Multiple Text Files at the Same Time

```
mkdir /home/"$USER"/Documents/fdupes\
&& cd /home/"$USER"/Documents/fdupes\
&& for i in {1..10}; do echo\
    "fdupes finds and removes duplicates."\
    > fdupes${i}.txt ; done
```

first lists the files in a numbered list (Figure 3). Note that the number at the beginning of the line will not necessarily match the number in the file name. If you now enter numbers separated by commas, they are tagged with a plus sign and remain intact, while the software removes all of the duplicates with a minus sign.

If you make a mistake, the `rg` command cancels your previous entries. Pressing Delete applies your entries. If you want to remove all duplicates except the first one displayed, use the command

```
fdupes -r -d -N /path
```

You do not need to press Delete here – the `-N` (no-prompt) option works without any confirmation.

Another selection option after calling `fdupes` with the `-d` option relies on the `se1` parameter. You can select all files with a specific term in the path by typing `se1 <term>`. To select all files whose path starts with the term, use `se1b <term>`. Use `se1e <term>` to select files whose path ends with the term. To select all files whose path corresponds exactly to the term, use the `se1m <term>` command. After that, you can decide which of the candidates you want to keep. Further options are described by the `help` command, which displays the matching `fdupes` man page sections.

Hard and Soft Links

Beyond this, you can tell `fdupes` to reduce the space required by duplicates by converting the duplicates into hard links. A hard link connects a name directly to a specific file in the system. Several hard links can refer to the same file (i.e., several directory entries or file names can exist for the same file). Unlike a soft link, where the pointer changes as soon as you rename the file, a hard link continues to point to the underlying file afterwards.

Suppose you manage a computer used by multiple users, and they store the same files in their home directories (e.g., YouTube videos or audio files). In this case, you cannot simply delete the files. Instead, the `-H` option lets you convert the duplicates to hard links. If a user deletes their copy, the other copies are still kept. However, this option is not universally suitable – if a user changes the metadata of a song or photo, for example, the data will also change in all linked objects. The same applies to editing text files. Because of this, `-H` is more suitable for read-only files. Another switch to be used with caution is `-s` (`--symbolink`), which converts duplicates of a file into symbolic links (soft links). In some cases, users save the symlink and accidentally delete the actual file.

Why the Command Line?

Why choose a command-line solution when there are graphical alternatives that could actually give

```
dd@ubudesk220411ts: ~$ fdupes ~/Documents/fdupes/
/home/dd/Documents/fdupes/fdupes5.txt
/home/dd/Documents/fdupes/fdupes10.txt
/home/dd/Documents/fdupes/fdupes6.txt
/home/dd/Documents/fdupes/fdupes3.txt
/home/dd/Documents/fdupes/fdupes7.txt
/home/dd/Documents/fdupes/fdupes4.txt
/home/dd/Documents/fdupes/fdupes8.txt
/home/dd/Documents/fdupes/fdupes2.txt
/home/dd/Documents/fdupes/fdupes9.txt
/home/dd/Documents/fdupes/fdupes1.txt

dd@ubudesk220411ts: ~$
```

Figure 1: The simplest method of finding duplicates does not need any call parameters to specify the directory.

```
dd@ubudesk220411ts: ~$ fdupes -r ~/Documents/
/home/dd/Documents/fdupes/fdupes5.txt
/home/dd/Documents/fdupes/fdupes10.txt
/home/dd/Documents/fdupes/fdupes6.txt
/home/dd/Documents/fdupes/fdupes3.txt
/home/dd/Documents/fdupes/fdupes7.txt
/home/dd/Documents/fdupes/fdupes4.txt
/home/dd/Documents/fdupes/fdupes8.txt
/home/dd/Documents/fdupes/fdupes2.txt
/home/dd/Documents/fdupes/fdupes9.txt
/home/dd/Documents/fdupes/fdupes1.txt

dd@ubudesk220411ts: ~$
```

Figure 2: Use the `-r` parameter to dig down further in a directory tree search.

```
dd@ubudesk220411ts: ~$ fdupes -d
Set 1 of 1:

1 [ ] /home/dd/Documents/fdupes/fdupes5.txt
2 [ ] /home/dd/Documents/fdupes/fdupes10.txt
3 [ ] /home/dd/Documents/fdupes/fdupes6.txt
4 [ ] /home/dd/Documents/fdupes/fdupes3.txt
5 [ ] /home/dd/Documents/fdupes/fdupes7.txt
6 [ ] /home/dd/Documents/fdupes/fdupes4.txt
7 [ ] /home/dd/Documents/fdupes/fdupes8.txt
8 [ ] /home/dd/Documents/fdupes/fdupes2.txt
9 [ ] /home/dd/Documents/fdupes/fdupes9.txt
10 [ ] /home/dd/Documents/fdupes/fdupes1.txt

( Preserve files [1 - 10, all, help] ):
Ready Set 1 of 1
```

Figure 3: The `-d` parameter is used to delete any duplicates found and lists them in a numbered list.

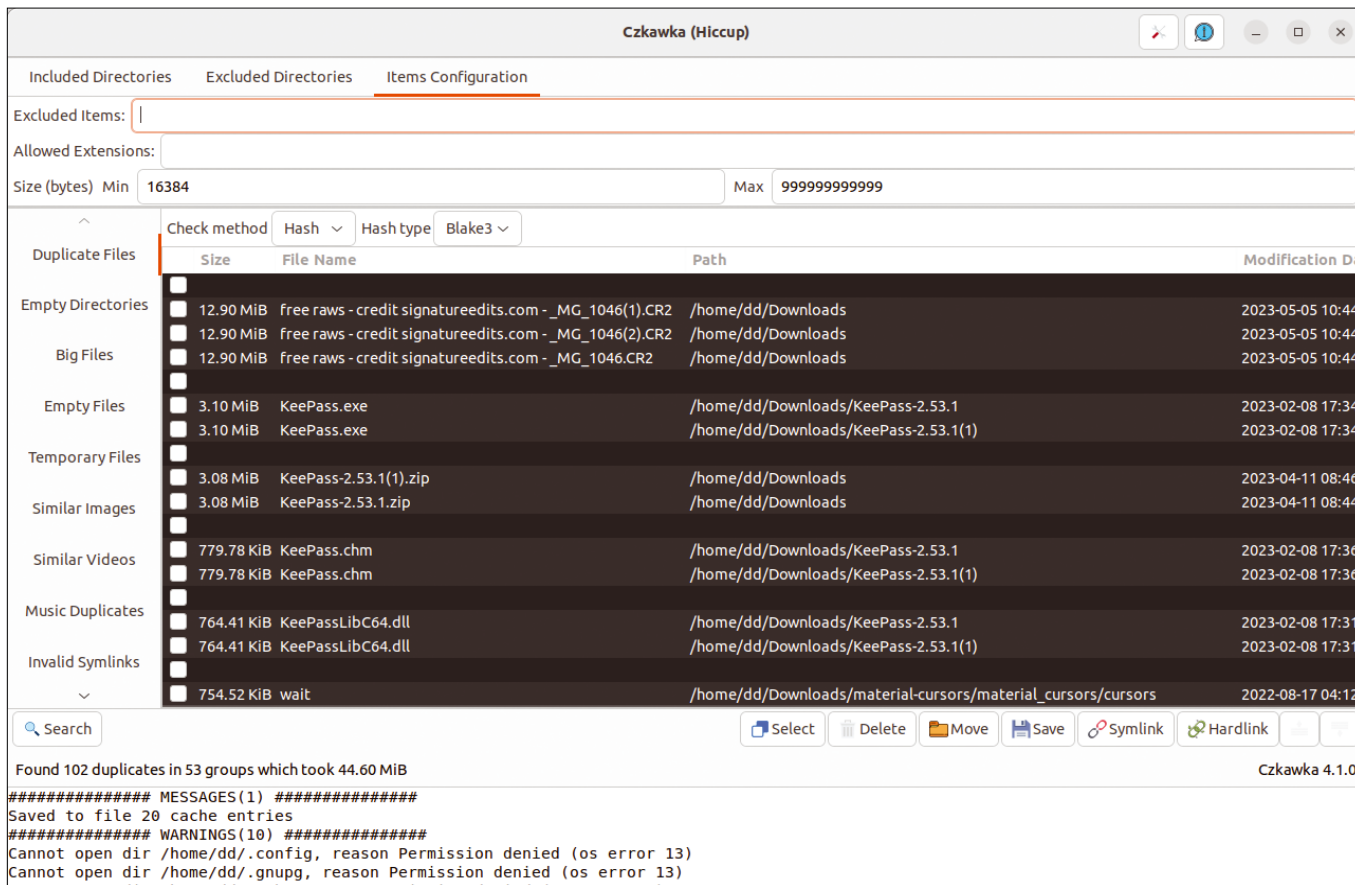


Figure 4: Czkawka is an implementation of the formerly widely used FSlint written in Rust. It is faster and uses less working memory than the predecessor.

you a better overview? On the one hand, GUI applications cannot be deployed on most servers. On the other hand, the terminal variant is far faster, especially when searching in larger directories.

The formerly widely used GUI application FSlint, written in Python, has now disappeared from the repositories of Ubuntu, Debian, and other distributions, for example. Instead, there is Czkawka (Figure 4), a far faster fork in the Rust programming language. DupeGuru (Figure 5) is another alternative. Jdupes is recommended as a replacement for fdupes. However, bear in mind that the commands differ in part between the two tools.

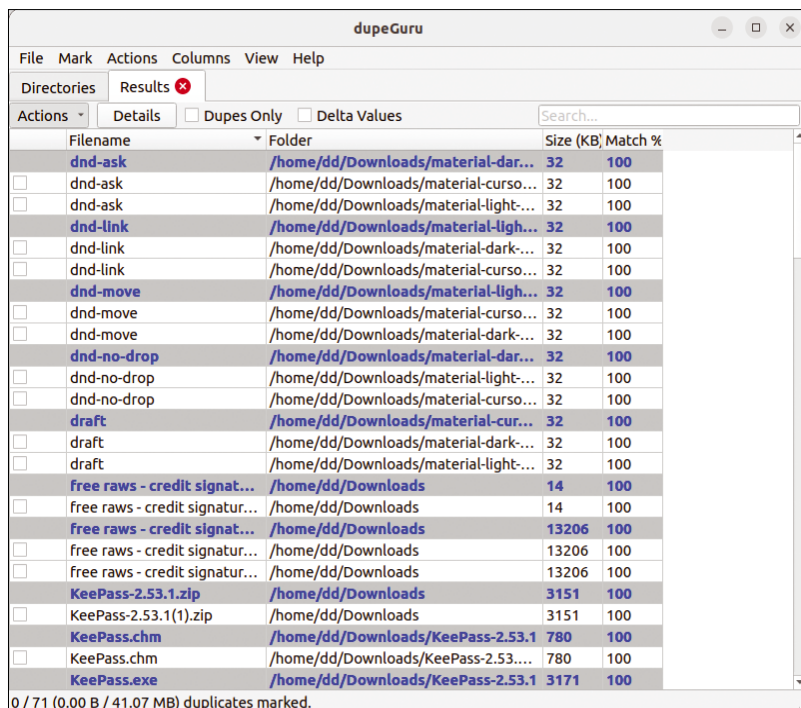


Figure 5: DupeGuru is another alternative if you want to find duplicates in a GUI. The results can be narrowed down using filters or regular expressions.

Conclusions

Fdupes lets you find, remove, or convert duplicates. However, there are pitfalls inherent to the typical use cases that can lead to data loss. In addition to the potential problems with symlinks and hard links, as mentioned earlier, there are system files of which duplicate versions must exist, but which are still identified as duplicates. Even empty directories or files cannot simply be deleted without thinking. The idea is to only remove those files and directories whose content and importance you can judge. ■■■

Info

- [1] fdupes: <https://github.com/adrianlopezroche/fdupes/releases>
- [2] Versions of fdupes: <https://repology.org/project/fdupes/versions>

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



@adminmagazine



@adminmag



ADMIN magazine



@adminmagazine

FOSSPicks

Sparkling gems and new releases from the world of Free and Open Source Software



Graham has this month spent far too much time (and money) using an Arduino to control the temperature, pressure, and flow of coffee being brewed by his espresso machine. **BY GRAHAM MORRISON**

Link manager

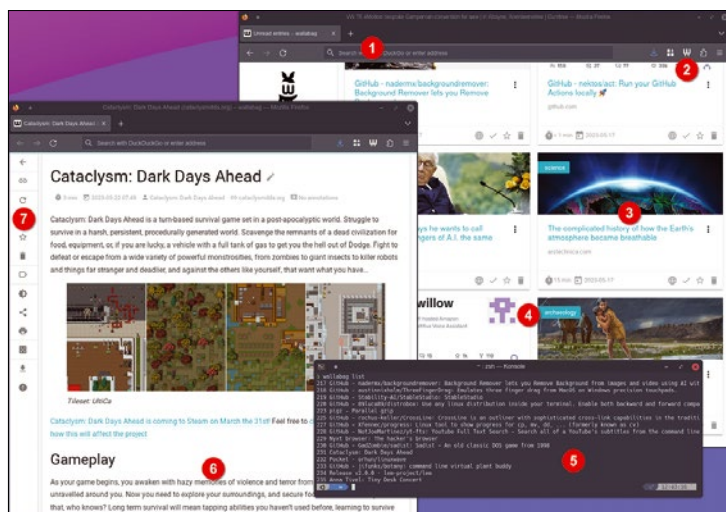
wallabag

For as long as there's been web browsing there have been ways to save and manage links for later reading. Of course, most of us use bookmarks, but it's also common to sync bookmarks across devices, or to use a third-party service to save a bookmark and cache its content for improved readability or for offline access. One of the best options is Pocket, a commercial subscription service with a generous free tier that was

called Read It Later, before being acquired by the Mozilla Corporation in 2017. Thanks to this commercial background and its continued patronage under Mozilla, Pocket has mature platform support, with apps for Android, iPad, iOS, macOS, and Windows, and first-party integration with Firefox. This makes Pocket convenient and accessible from wherever and whenever you need it. It's even central to the creation of these very pages because the author extensively uses it to cache open source software discoveries from whichever machine or device they were discovered on.

Despite Mozilla's position as custodian of Firefox and the open web, and its promises to the contrary, Pocket's server code remains proprietary, and the service recently dropped one of its most useful features. This was its ability to add links to your Pocket account by sending an email to a special address. This was a great solution if you didn't have an integrated client handy and was the only option if you used a web browser without Pocket support, such as the brilliant qutebrowser. Which is where we come to true open source alternatives, and wallabag is the best of them. Wallabag is a PHP web application and server with accompanying apps and browser integrations. You can self-host the server or pay a small fee (about \$10) to use the project's hosted service. Setting this up yourself is relatively convoluted, but can be done with a cheap VPS or home server. You'll need a domain name, NGINX, MySQL/MariaDB, Let's Encrypt, and some configuration tinkering – nothing the average Linux nerd or sys admin can't handle before breakfast, and it's often worth doing it this way for privacy and peace of mind. You can then generate client keys from the web client for any of the accompanying apps, which will securely and privately add links to your account whenever you summon them.

A clever tagging system can be used to either categorize links manually or with automatic pattern matching, such as listing certain sites for "science" or "synthesizers." These make browsing your new and archived links much easier to manage, regardless of whether you use the web portal, or Android and iOS apps, to read the content you save. Each client will extract the contents of your links, allowing you to read an article within wallabag's own distraction-free reading environment, or you can use integrated RSS generation to turn your lists of links into an RSS feed for use with a native Linux newsreader, such as Read It Later, or even to get Calibre to automatically send articles to your e-ink device. There's even a command-line client for adding, removing, and tagging articles, and it perfectly replaces Pocket's old email functionality, letting you add wallabag support to any browser or scripted solution. Wallabag excels like this because it's open source, and developers can make it work anywhere from Emacs to Safari on macOS, and there's never been a better time to give it a go.



1. Self hosted: Even a Raspberry Pi can run the server, or use a cheap VPS, but you will need your own domain name. **2. Browser integration:** Most browsers, including Firefox, can integrate a wallabag add-on for seamlessly adding sites and tags to your finds. **3. Thumbnail or list views:** The thumbnail preview view includes a synopsis of any saved web page. **4. Tags:** Your own tags can be added manually or, more importantly, automatically according to certain rules. **5. Terminal client:** A command-line tool can be used to add, remove, and list your links. **6. Article view:** Wallabag will extract the content of an article to make it more readable and accessible. **7. Import:** Add your links database from many other services, including Pocket, Readability, and various bookmark managers.

Project Website
<https://www.wallabag.it>

wallabag client

Read It Later

As great as wallabag is at making web-based articles easy to read and consume, the standard web portal and page rendering is never going to be able to compete with a native application. This is something that other caching services offer, especially on other operating systems and devices, and it's something that this Read It Later app now does for Linux. Read It Later is a beautifully designed Gnome desktop application for accessing your wallabag library of content, adding new links, and reading stories. It does this with a wonderfully minimal interface so that the browsing and reading experience can be as distraction-free as possible. The article view can be toggled between unread, favorite, and archived stories, and these are

shown in a detailed list alongside a thumbnail from the article.

Unlike the web interface which adds an icon-only view and further filtering options, Read It Later is intended to catch-up on unread articles, which are sorted according to the time they should take to read. These options are taken from the web interface, but the reading experience itself is much better than either the web interface or the Android application. Selecting an article will render its contents within the frame of the desktop application, reformatted to look and render perfectly. All distractions and advertising are removed, and the text is beautifully rendered and presented within Gnome's minimal window frame and titlebar. As with other modern GTK4 applications, light and dark theming will follow your



Read It Later offers a minimal and distraction-free reading environment for whatever articles you collect with wallabag.

desktop configuration, and the few options offered are accessible from the embedded menu in the titlebar. You can mark a story as a favorite or add a link to a new story. There's no other manual control over rendering style, and the only management options you have are to either favorite or delete a post which is automatically marked as read after a web-configured period of time. It sounds simple, but it's the best way to read the links you hoard, and a refreshing change from the distraction of reading online.

Project Website

<https://gitlab.gnome.org/World/read-it-later>

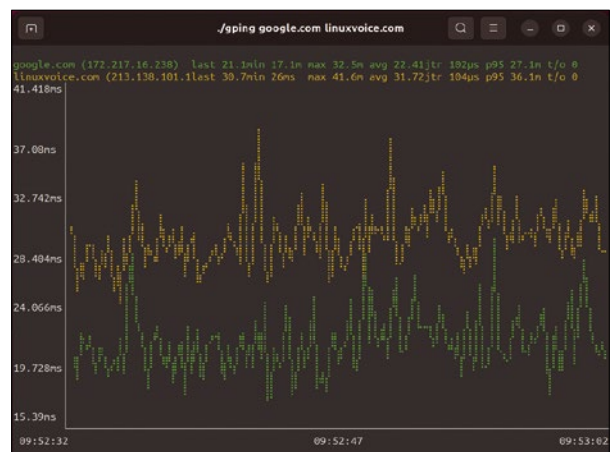
Visual ping

gping

If there's one command that Linux users everywhere have relied on for decades, it's ping. Written in the early 1980s, before almost anyone had Internet access and networks were built with coaxial cables, it's a command that's followed us from simple home networks and dial-up modem connections, through full-fiber symmetric broadband, and into the multi-provider cloud-synergized scaling. Ping has survived because it performs one simple task perfectly, letting you always know whether some other machine on a network is accessible and responding to your requests to acknowledge itself. The name "ping" famously refers to the pulse sound of submarine sonar when the sound echos off some

remote object. Linux ping does the same for remote servers, telling you a remote machine is accessible and how long it takes for a packet to trek from your machine to the remote machine and back. Ping's output is nothing more than a line of text for each packet showing the destination IP and the time the trip took. This leaves a lot of room for improvement, especially if you want to track responses over time, and that's exactly what the super-simple and super-useful gping does.

Gping is still a command-line ping but with a graph showing response times for one or more servers. At its simplest, it's a drop-in replacement for ping where, instead of the individual line of output for each response, there's a point drawn on a graph. You can change the interval, the amount of buffer, and the horizontal and vertical margins, but the graph will also start to scroll



Gping can resolve both IPv4 and IPv6 target addresses and use color to better differentiate between response times.

when it hits the edge, allowing you to see quickly any variation in the remote host's accessibility from your current machine. Best of all, adding more than one target to the command will graph response times for each separately on the same graph, so you can better check network performance, or keep an eye on your own servers, and it looks awesome when you're trying to impress people with your command-line skills.

Project Website

<https://github.com/orf/gping>

Screen controller

DDCcontrol

We looked at a small command-line utility called `ddcutil` for controlling monitors a little while ago. It is an excellent tool for controlling various monitor settings exposed over a screen's I2C bus using the DDC/CI standard. The only problem was that `ddcutil` could be a little too limited, especially when it came to exploring a screen's more experimental or bespoke features and other parameters exposed over I2C from your monitor. `DDCcontrol` is a more comprehensive utility with the same goals. It works at a hardware level, rather than through user space, and can access manufacture-specific parameters not accessible with `ddcutil`. Like `ddcutil`, however, it does all this over the I2C bus, and this means there's always a risk you could mess with

something you can't revert, so it's worth being cautious and to only experiment on hardware you're prepared to lose.

With that out of the way, `DDCcontrol` runs from the command line and can first conveniently scan your I2C bus for any screens. This is especially helpful because getting the wrong device can destroy more than just your monitor. As with `dd`, sending potentially destructive commands to other devices on your system will cause problems bigger than the wrong screen brightness, and with the correct device now added to the command, you can safely probe your screen for capabilities. This risk is mitigated in `DDCcontrol` by using a database of known screens and their features, allowing you to safely access parameters

```

graham@graham-neon: ~
> ddccontrol dev:/dev/i2c-8
ddccontrol version 0.6.0
Copyright 2004-2005 Oleg I. Vdovikin (oleg@cs.msu.su)
Copyright 2004-2006 Nicolas Boichat (nicolas@boichat.ch)
This program comes with ABSOLUTELY NO WARRANTY.
You may redistribute copies of this program under the terms of the GNU General Public License.

Reading EDID and initializing DDC/CI at bus dev:/dev/i2c-8...
I/O warning : failed to load external entity "/usr/share/ddccontrol-db/monitor/GSM76E4.xml"
Document not parsed successfully.

EDID readings:
  Plug and Play ID: GSM76E4 [LG Standard LCD]
  Input type: Analog

***** WARNING *****
There is no support for your monitor in the database, but ddccontrol is
using a generic profile for your monitor's manufacturer. Some controls
may not be supported, or may not work as expected.
Please update ddccontrol-db, or, if you are already using the latest
version, please send the output of the following command to
ddccontrol-users@lists.sourceforge.net:

LANG=LC_ALL= ddccontrol -p -c -d

Thank you.
***** WARNING *****

> LG Standard LCD
> Color Settings
  > Brightness and Contrast
    > id=brightness, name=Brightness, address=0x10, delay=1ms, type=0
      supported, value=0, maximum=100
    > id=contrast, name=Contrast, address=0x12, delay=1ms, type=0
      supported, value=0, maximum=100
  > Color maximum level
    > id=red, name=Red maximum level, address=0x16, delay=1ms, type=0
      supported, value=50, maximum=100
    > id=green, name=Green maximum level, address=0x18, delay=1ms, type=0
      supported, value=50, maximum=100
    > id=blue, name=Blue maximum level, address=0x1a, delay=1ms, type=0
      supported, value=50, maximum=100

```

If the command-line options in `DDCcontrol` are too much, there's an excellent GUI that provides the same options in a much more pleasant graphical window.

outside of the MCCS specification. They're typically options that might otherwise require a menu deep-dive or the activation of some engineering mode, such as refresh rates and color gamuts. It's still useful for switching inputs, but you might also find you can split the display and change the dark levels of your panel.

Project Website

<https://github.com/ddccontrol/ddccontrol>

Process manager

killport

It's ugly, it leaves you feeling guilty, and it's something you really shouldn't do. But it is sometimes necessary to kill a running process. This usually happens only after trying everything else. You've asked politely using the application's Quit option. You pressed colon and typed `q`. Even mashing Escape or Ctrl+C hasn't worked, while the process ID in `top` persists in sucking your compute and memory resources. This is where `kill`, or even `kill -9`, is your friend. Like a cold-blooded assassin, it will destroy a process and purge your system of its memory. But what happens when you don't know which process to kill, or

don't know which process is responsible for the process in `stasis`. Like any good assassin, you then need to do a little reconnaissance.

The `top` command and its ilk are great at displaying which processes are stalled and which are using CPU and memory resources, and you can use commands such as `lsof` to list which processes are accessing which files and devices. But processes accessing the network or running as a server are harder to track down. These processes will use a port for communication, usually waiting on data to arrive or processing data as it passes through, and killing processes such as these is precisely why the aptly named `killport` was developed. At its simplest, you run `killport` with a single argument for the port number. For

```

graham@graham-neon: ~/build/killport/target/debug
./killport -h
A command-line tool to easily kill processes running on a specified port.

Usage: killport [OPTIONS] <ports>...

Arguments:
  <ports>...  The list of port numbers to kill processes on

Options:
  -s, --signal <SIG>  SIG is a signal name [default: sigterm] [possible values: sigkill, sigterm]
  -v, --verbose...    More output per occurrence
  -q, --quiet...     Less output per occurrence
  -h, --help         Print help
  -V, --version     Print version

```

`killport` helps you to stop services running on specific ports without having to learn the arcane ways of `netstat`.

instance, `killport 22` would kill the SSH daemon if you're running it with the default configuration. Similarly, you can include more than one port to request the massacre of an entire suite of processes, and you can specify the signal if you want to inflict extra torture before a brute force attack. It's simple but works brilliantly and is a much better option than trying to work out which processes are attached to which services through `netstat`.

Project Website

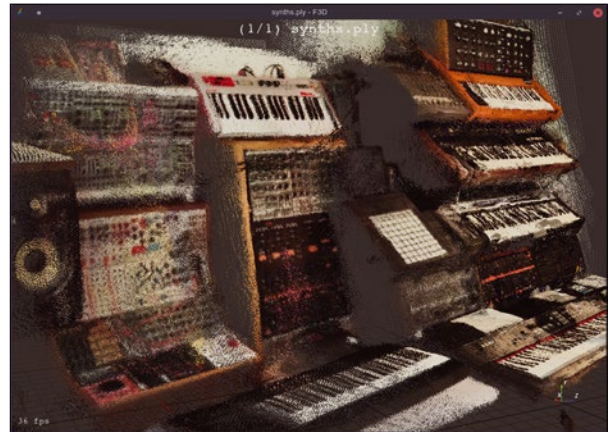
<https://github.com/jkfran/killport>

3D viewer**F3D**

Ever since computers could draw dots and connect lines between them, we've had 3D viewers. Even in the 1960s, computers could drive oscilloscopes and cathode ray tubes to show three-dimensional objects mapped into a two-dimensional plane. It's an obsession that stuck with computers through the space-station approach in the 1968 movie *2001: A Space Odyssey*, in the video game *Maze* (aka *Maze War*) in the 1970s, and through the plethora of wireframe and filled polygons that filled out televisions in the 1980s before accelerated graphics made it trivial from the 1990s onwards. Now you can view things in 3D anywhere, from your phone, within a web browser, and in real 3D with a VR headset. But

you're still stuck if you need a quick preview of a 3D object from the command line. With a few minutes to spare on a desktop, you could quickly drop a file into Blender, create a quick scene, and render this, but the brilliant F3D does a better job with the output and does this without any complicated UI interaction right from the command line.

F3D will load a whole host of 3D files, from VTK files from 1993 to scientific datasets and PTS point clouds generated by modern LIDAR with millions of points. Whatever the content, it's beautifully rendered into a separate desktop window where you also have considerable control over the output. Mouse and shift will rotate and slide the scene around in three dimensions,



F3D not only loads static 3D objects, but it can also load and display glTF 3D animations.

while hotkeys can enable anti-aliasing, tone mapping, color cycling, point sprite rendering, HDR and environment mapping, opacity, and volumetric rendering. These options can also be provided on the command line, and the ray tracing option requires special build options to have been enabled, but they all look wonderful, and you often find yourself downloading all kinds of 3D files just so you can explore them so conveniently.

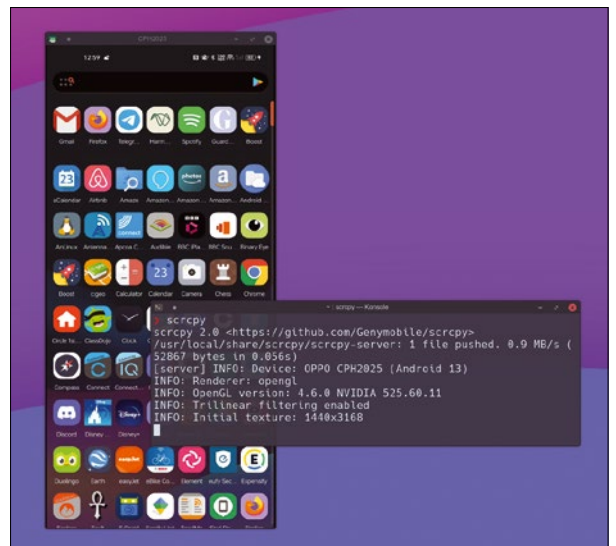
Project Website

<https://f3d.app>

Remote Android**scrcpy**

There was a time when accessing a remote desktop was an essential part of running Linux. It was a time when, unless you wanted to wrestle with Webmin, the best way to solve problems remotely was to simply use the desktop of the troubled system. This helped with Linux adoption because Linux is so good at it. Even before the Internet became all-conquering, graphical applications could be tunneled across a network through the X11 protocol itself, enabling you to run remote graphical applications locally. A little later, VNC solved the same problem by compressing an entire desktop into a set of JPEGs that could be reconstituted anywhere with a network connection. It often suffered from poor quality, but it got the job done.

Considering Android is Linux, it's surprising that the platform is not equally festooned with remote-viewing options. There are very few for unrooted devices that let you see exactly what you would see looking at the screen in your hand. But this is what the terribly named (for dyslexics) scrcpy does, and it does this without root and without requiring anything to be installed on your Android device. The only requirement is that ADB debugging is enabled on your Android device, and that you trust the required USB connection you make between your phone and your Linux machine. It's then a matter of running the scrcpy command, if you can type it, and waiting for the connection to start. Within a few moments you'll be presented



Stream an exact replica of your Android device's screen to your Linux desktop, with glorious acceleration.

with a 1:1 frame-by-frame perfect capture of what you see on your Android screen. It's amazing how useful this can be, from streaming the screen across a network, to replying to messages while you're in VR.

Project Website

<https://github.com/Genymobile/scrcpy>

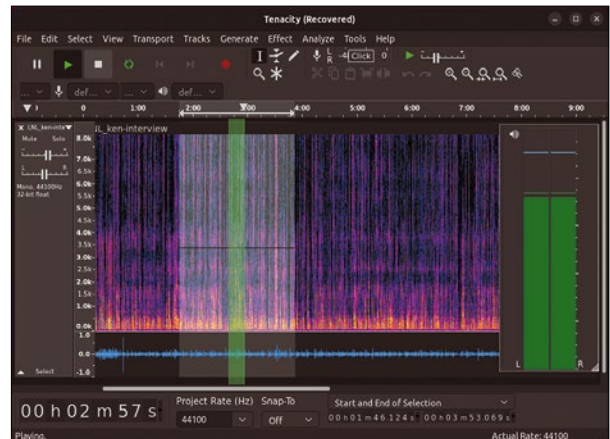
Audio editor

Tenacity

Writing audio software is a difficult and thankless task. Not only is the signal-processing code a challenge to program, and the Linux audio stack difficult to interact with, creating a user interface to help users ignore this complexity is almost impossible. It's why the applications that find a successful approach change very little over time. Modern Cubase still has a lot in common with 1989 Cubase, Ableton hasn't changed since its release in 2001, and on Linux, Ardour remains as austere as it did in 2005, albeit with a host of modern features and incredibly powerful levels of control. But it's Audacity that's perhaps changed the least while remaining one of the most popular audio editors on Windows, macOS, and Linux. Audacity is used by everyone, from ordinary desktop users converting FLACs to MP3s, through amateur musicians mastering their audio for distribution on Bandcamp, to professional podcasters. It remains a rock-solid application that can record and edit glitch-free for hours.

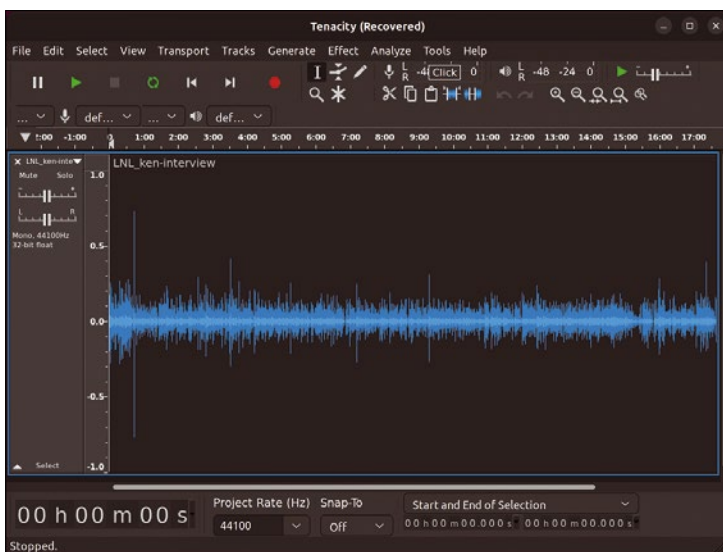
Despite this commitment to consistency, Audacity development has been considered too slow by many, especially when the project added, and reneged, on adding opt-out telemetry after it was taken over by the Muse Group (the same organization behind the brilliant MuseScore). This led to two separate forks of the project, both committed to bringing Audacity into the modern age without any telemetry, and under the auspices of their respective open source communities. Initiatives such as these are difficult to pull off and rarely succeed in supplanting an original, but both projects have survived and have recently merged under the name of the most successful, Tenacity.

Tenacity promises to be a fresh, modern, more convenient, and more practical audio editor than Audacity. The latest release is a great showcase of what progress has already been made. The main differences can be seen in the user interface, which has been rethemed. Icons, text rendering, and theme colors all look vastly improved and are vastly superior to the



Competition is a good thing, and hopefully Tenacity can keep up with recent Audacity developments to include real-time audio effects.

originals on a dark-themed desktop. This was a feature that Audacity seemed reluctant to add, and its own dark mode can be hit-and-miss when it comes to integrating with the VU meters, for example, or the waveform rendering. It helps to make Tenacity more pleasurable to use, while also helping it feel more modern. It's also built without Audacity's telemetry, requiring neither the build flag to be set nor the runtime setting to be unset, ensuring your data isn't going anywhere. Everything else is the best of what Audacity included when the fork occurred at version 3.0.2. This means you get decades of developer and user experience and a rock-solid audio editor. Audacity is still a great project, of course, and has itself improved significantly by finally adding real-time effects processing, albeit at an early stage, and it will be interesting to see whether Tenacity can merge these additions into its own code, now significantly diverged from Audacity's, or whether the team will be able to maintain enough momentum to create a truly independent project with its own unique approach to audio editing. There's no doubt that there's huge demand for an open source audio editor like this, and there's potentially significant rewards for the project that succeeds.



Tenacity is a fork of Audacity that's already overhauled its aesthetic and hopes to take things further with community involvement.

Project Website
<https://github.com/tenacityteam/tenacity>

Survival roguelike

Cataclysm: Dark Days Ahead

This is a complicated adventure game with a steep learning curve. It runs from the command line, and when you create a character and drop into the game world, everything initially looks a lot like the original NetHack. It's built using text characters with ncurses, with a central character for your avatar, a "fog of war" that lifts according to your line of sight as you explore, and a selection of stats and attributes on the right. Like NetHack, the ASCII text can be replaced with graphical tiles which can make the game feel a lot more modern but will also pull the gameplay out of the terminal and onto your desktop. Either way, both versions of the game

play identically, and they're both equally difficult. Cataclysm describes itself as a "turn-based survival game set in a post-apocalyptic world," and this is the world you need to explore from the outset while attending to your attributes to keep you alive. Turn-based means there's a move cost in every action you perform, and starting difficulty is dependent on which starting scenario you choose, with "Evacuee" giving you enough resources to help initially. This also depends on which character profession you choose, because different professions come equipped with different default inventories. There are many professions to choose between, from Ballroom Dancer to Punk Rock



Cataclysm: Dark Days Ahead is an open source roguelike adventure that's now even available on Steam.

Dude, and everything seems to affect everything else. It takes many hours to begin to understand the effects of the choices you make both before the game starts, and within the gaming environment itself, but some excellent onboarding guides and online videos can really help. The end result is a roguelike with elements of NetHack, a scenario like Fallout, and an emphasis on survival and combat. Like NetHack, it rewards the time you invest in learning it with a deep gaming experience, albeit with a much harder learning curve.

Project Website

<https://cataclysmdda.org>

Plant simulator

botany

We're used to the idea that many AAA games are becoming more complicated and more demanding, often taking dozens or even hundreds of hours to conquer. But there's a subclass of games that attempt to do the opposite. It's one of the reasons why Stardew Valley was so successful, as many players used its farming and growing aspects to wind down and relax without having to plough their real gardens. The wonderful botany takes this even further in both implementation and in the demands it puts on you mentally. It does this first by being a simple Python 3 command-line application you can grab and run from its repository without requiring

installation or dependencies for configuration. Secondly, it requires very little attention or thought. Like a real plant, it demands simply that you take a look occasionally and don't abandon it.

On first launch, the game starts with you being given a seed. You're not told any of this directly, but you can read about it within the included instructions. It then becomes your task to water the plant, check on its growth and remember to come back every 24 hours to do the same. There are more than 20 different species of plants, all drawn as ASCII art, and given random attributes when you first run the game, but just like with real plants, it takes days for anything to appear from that initial seed. If you forget to water your ASCII, it will perish, and your flower can mutate at any stage of its development. But if it survives, you can harvest the growth



Botany lets you grow ASCII plants on the command line and let your friends visit to help water and watch over your creation.

and use this to seed a new generation with a 20-percent growth bonus. Even more impressively, you can visit other people growing plants from their command line by entering their name, and they can visit you. Visitors can view and even water your plant, helping to keep everything healthy if you go on holiday, for instance, which is a lot easier to organize over a network than someone visiting your IRL garden.

Project Website

<https://github.com/jifunks/botany>

Edit landscape photos with RawTherapee 5.9

Perfect Balance

The current RawTherapee version finally adds selective image editing, among other long-desired features, to help it compete with king of the hill, darktable.

BY ANNA SIMON

When developing landscape photos, the problem often arises that individual areas of the image, especially very bright or overly dark areas, have a color cast or colors that look too intense or too pale. Changing contrast or saturation is more likely to intensify the phenomenon. Good masking and selection functions are must-haves for landscape photographers.

Both free and commercial RAW developers now offer extensive tools for selective image processing. But of all the open source programs, only darktable has offered these features thus far. Although darktable is basically an excellent piece of software, there are drawbacks for some users. It takes a bit of getting used to, and you have to thoroughly understand what is going on to achieve good results. In addition, darktable only runs really smoothly if you have a fast graphics card.

The release of RawTherapee 5.9 [1] in November 2022 after several years of development added the ability to modify specific areas of an image in an intuitive and effective way [2]. Developer Jacques Desmis was inspired by the ingenious selection technology of the popular Nik plugins when programming the function. The good thing is: The program is a RAW developer with similar performance to darktable [3], but avoids the above-mentioned drawbacks [4].

There is actually only one RAW developer backing RawTherapee, unlike most similar programs. It does not have a genuine management module, but only a kind of file browser or image viewer. Launch the application and use the browser to navigate to the directory where the photos are stored. Double-clicking on a thumbnail opens the photo in the editor. You will find the editing tools on the right.

Preliminary Work

Start by adjusting some of the default settings. Click the button for this and you are taken to a separate window with several tabs. First, you need to take care of two options: the color of the user interface and color management, or

automatic detection of the screen profile. RawTherapee's standard user interface is too dark to be able to adjust the intensity of the colors and the image brightness correctly. You can define a lighter color quite a way down in *General | Appearance*. The *TooWaGrey – Average Surround* design is recommended, because it comes quite close to a neutral gray.

Then set the program to automatically detect and use your screen profile. To do this, you first need to install a profile in the color settings of the operating system. In RawTherapee Settings, click *Color Management*, in the *Monitor* group, check *Use operating system's main monitor color profile*, and restart the software.

In the default configuration, the tool activates sophisticated, minimally invasive automatic image optimization for RAW files. To get the most out of a photo, you just need to move a few sliders in many cases. As soon as you open a RAW file, RawTherapee automatically applies three processing steps to the image. The *Auto tone curve* in the *Exposure* module does most of the work. The program creates the curve by comparing the actual RAW file with the JPEG embedded in the file. It tries to imitate today's typically quite useful in-camera image optimization.

The application enables *Input Sharpening* (right at the end of the *Raw* tab). In contrast to conventional sharpening, the highly efficient sharpening method does not take place at the end of the processing, but at the outset, directly after the image has been descreened in the linear RGB color space. The tool also turns on automatic correction of chromatic anomalies.

Lighten and Darken

The sample photo shows a mountain landscape near the community of Lunz am See in the Mostviertel region of Lower Austria. You have to brighten many landscape images to a greater or lesser extent before doing anything else. The camera underexposes them by default to preserve the structure in the brightest areas of the image to the extent possible (Figure 1).

To do this, move the *Exposure compensation* slider in the *Exposure* module a little to the right. After several attempts, a value of 0.8 proved to be perfect in our tests. Then darken the sky using the gray gradient filter. The tool makes the upper part of the image darker, inserting a smooth transition from light to dark. You can enable the filter using the button in the module's titlebar. Set the *Intensity* of the effect to about 1.5. Using the sliders lower down, you can adjust other filter properties, such as the position and size, if necessary.

Contrast and Saturation

Next increase the saturation and local contrast. Move the *Saturation* slider below *Exposure* to the right approximately to a value of 67 and enable *Detail* under *Local Contrast*. This applies an effect that lies somewhere between sharpening and a (global) contrast increase. If you strengthen the effect, this generally increases the clarity of the image.

Internally, the filter uses a blurred mask – which explains the *Radius* slider that determines the amount of blur. If you scale this value up, the contrast increases. If you reduce it, it affects the finer details, and the image appears sharper. Meaningful values for this controller usually range between 100 and 150.

The *Intensity* specifies the extent to which the contrast increases. In addition, you can use *Dark areas* and *Light areas* to determine whether the effect will be stronger or weaker in some areas compared with others. You need to be careful with this effect: If the values you choose are too high, the image will look too stark. You need a higher value such as 0.75 for the intensity.

Also watch out for halos – unnatural-looking areas that are too light or too dark, which often appear near contrast edges. You can see a halo in the clouds in the sample image: They border directly on the ridge in the left half of the picture. You need to reduce the intensity for the bright areas by setting the slider to a value of 0.5.

L*a*b* Fine Tuning

The color of the grass in the foreground is a bit too intense now, while the mountains in the background – and dark colors in general – appear too pale. You can correct these details using the *Lab* module in *Exposure*.

As the name of the tool suggests, it does not operate in the RGB color space (i.e., with the three color channels of red, green, and blue), but in the L*a*b* color space. In this space, the photo is made up of three channels: L* (lightness), a*, and b*. L* only stores brightness information. The a* channel stores the intensity and tone of the colors green and red, while the b* channel contains information on blue and yellow. For certain editing steps, especially when



Figure 1: Without automatic image optimization, the original image will appear severely underexposed and low in contrast; this ensures that the structures in the brightest areas are preserved.

precisely adjusting the hue and saturation, you will achieve better results in L*a*b* than with tools that work in the RGB space.

The tool has six (initially hidden) curve types that let you manipulate the luminance, hue, and saturation (or chromaticity) of specific colors or tonal ranges. The curve designations *CC*, *CH*, *CL*, *HH*, *LC*, and *LH* are abbreviations (see Table 1).

The top three curves exist in many other RAW developers and photo editors, too. For example, show *CH* by clicking on the triangle and then selecting *Min/Max Control Points* from the menu. You will now see a square divided horizontally in the middle by a spline. At the bottom edge, there is a rainbow color gradient. There are also six verticals in red, yellow, green, cyan, blue, and violet. If you click on the point where the yellow vertical line intersects the horizontal spline and drag it up or down, the yellow tones in the preview become stronger or paler. You can use the *LH* and *HH* curves in a similar way to manipulate the brightness and hue of individual hues (Figure 2).

Table 1: Curve Designations

Abbreviation	Meaning
<i>CC</i>	Chromaticity according to chromaticity
<i>CH</i>	Chromaticity according to hue
<i>CL</i>	Chromaticity according to luminance
<i>HH</i>	Hue according to hue
<i>LC</i>	Luminance according to chromaticity
<i>LH</i>	Luminance according to hue



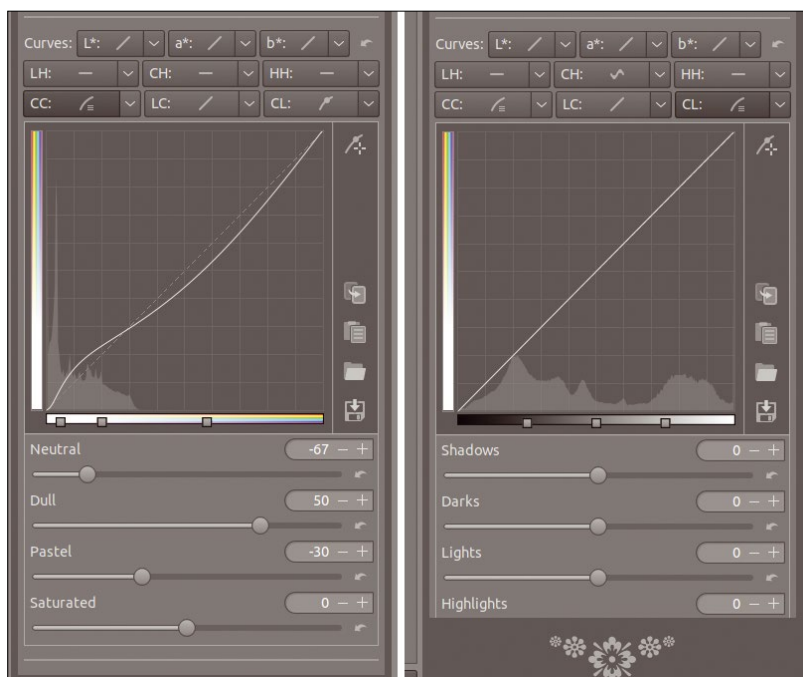
Figure 2: After some basic brightness, contrast, and saturation correction, the grass looks too intense, while the mountains are too pale. This can be fixed with the help of the L*a*b* curves.

The lower curves are a slightly more unusual. In the example, I will mainly be focusing on *CC* and *CL*. Select the *Parametric* or *Standard* curve type. In *CC*, four sliders appear under the parametric curve, dividing the colors into four groups. *Neutral* lets you further wash out or intensify near-white, gray, or black colors. In the case of the sample photo, this mainly means the clouds.

It won't hurt to reduce the color here. Because I have globally intensified the colors with the saturation slider in the exposure module, they have acquired a slight color cast. Set the value to -67. The next slider lets you change matte colors, which is useful for the forest and mountains. Choose, say, 50 here. Finally, I'll reduce the intensity of the pastel colors, which mainly means the grass (Figure 3).

In the next step, select the *Standard* curve type for *CL*. Click the top button in the vertical

Figure 3: Use the *CC*, *LC*, and *CL* curves to independently manipulate saturation and brightness for saturated, matte, and pastel colors and shadows, depths, and highlights, respectively.



toolbar to the right of the curve diagram. When you move the mouse into the preview window, RawTherapee shows you – in the curve graphic – the area of the curve that holds the color you are currently pointing to. You can increase the intensity of dark colors by clicking a point near the bottom of the curve and dragging it upward. Initially, even light colors have become more intense, because the upper half of the curve has automatically shifted upward. You can correct this by moving a point in the upper half slightly downward. The curve should now have an inverted S shape.

Detailed Adjustments

Let's highlight the structure of the clouds in *Local* (hand symbol). Enable the module in the titlebar and click *Add* in the *Settings* field. In the preview window, a red-bordered oval with a red circle in the center appears; move this to the clouds. In *Spot method* in the *Tools* panel, select *Full image*. After doing so, the application will only show you the circle. This selects all the colors in the image that are similar to the color below the image.

In *Add tool to current spot*, select the *Local Contrast and Wavelets* effect. Further down you will find the matching settings. Although they are similar to the local contrast settings in the *Details* tab, they also include a *Scope* slider. You can use it to control how similar the selected colors must be to those in the center of the selection. High values mean that RawTherapee also includes less-similar colors. This is why you only want to set this value to about 25; this applies the effect to the sky only. Then set the value for *Radius* to 100 and to 0.67 for the *Overall Strength*.

Now I'll darken the blue mountains in the background a bit. To do this, create another selection as described above, but keep the *Normal*

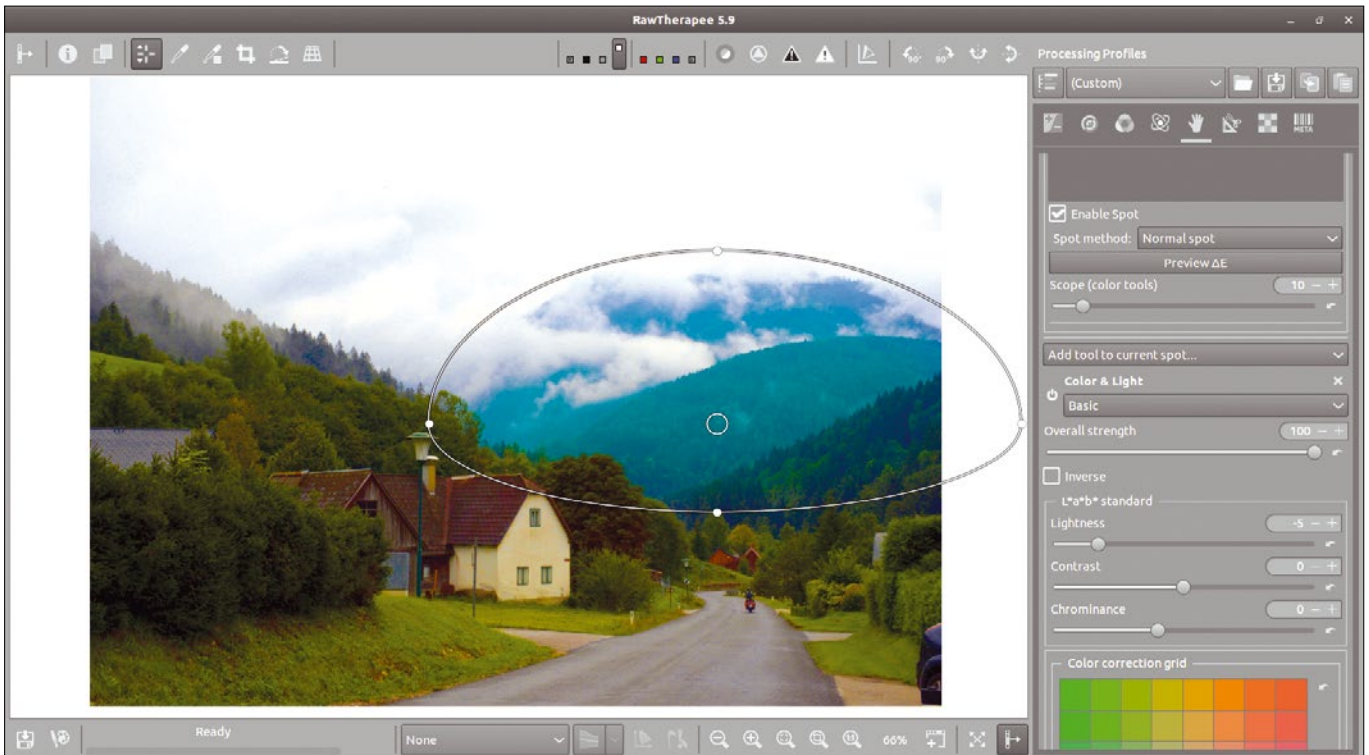


Figure 4: If you reduce the range of the selection with the Range slider, RawTherapee changes only the blue of the mountains, but not the clouds that are also selected.

Spot mode. Move the center of the spot to a blue spot in the area of the mountains. Then enlarge the spot by dragging the four points on the oval outward across the entire mountain. Reduce the number of colors included using *Range (Color Tools)*.

For effects that do not come with their own range sliders, you can control the range with the global slider. These include color tools such as *Color and Light*. If you restrict the area, the application changes only the blue of the mountains and ignores the clouds, which are also selected (Figure 4). Add the *Color and Light* tool to this second selection and set the *Brightness* there to -5.

To finish up, there are only a few small tasks left to complete. Straighten the image, reduce the noise, and remove the car fragment on the right edge of the image using the spot removal tool. You will find the rotation tool in *Transform | Lens Corrections*. Basically, you will want to go for the *Select Guideline* function instead of setting the angle. In *Select Guideline*, trace a vertical or horizontal line in the preview. In the sample image, for example, the lamp is a good choice for this.

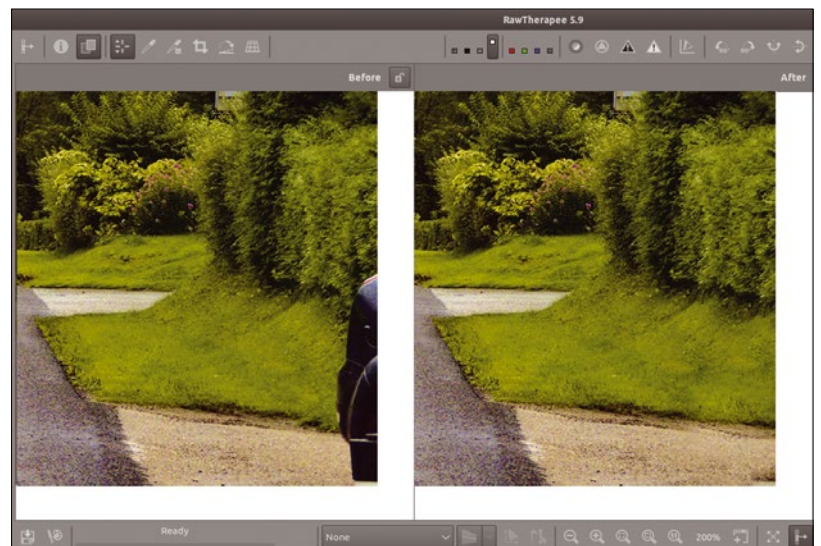
Remove Distracting Objects

To remove distracting objects from an image, enable the spot removal tool in *Details*. As soon as you click on the pencil icon, the mouse cursor turns into a small cross, and you can set a point to remove the car fragment. Hold down Ctrl and

click on the offending element. You will then see two small concentric circles at this point. Drag the inner circle outward to make it larger.

Actually, there are two brushes on top of each other now: The place covering the target area is at the bottom, and the one for the source area is at the top. Click on the circles and move the upper brush to the place you want to blend the car with – in this case, an area of the image to the left of the car (Figure 5). Except when removing very small spots, you will typically need to set several points to remove the object completely. Finally, use the pencil icon again to hide the brush marks or the retouched areas in the preview.

Figure 5: The new spot removal tool lets users remove distracting objects, provided they are not too large.



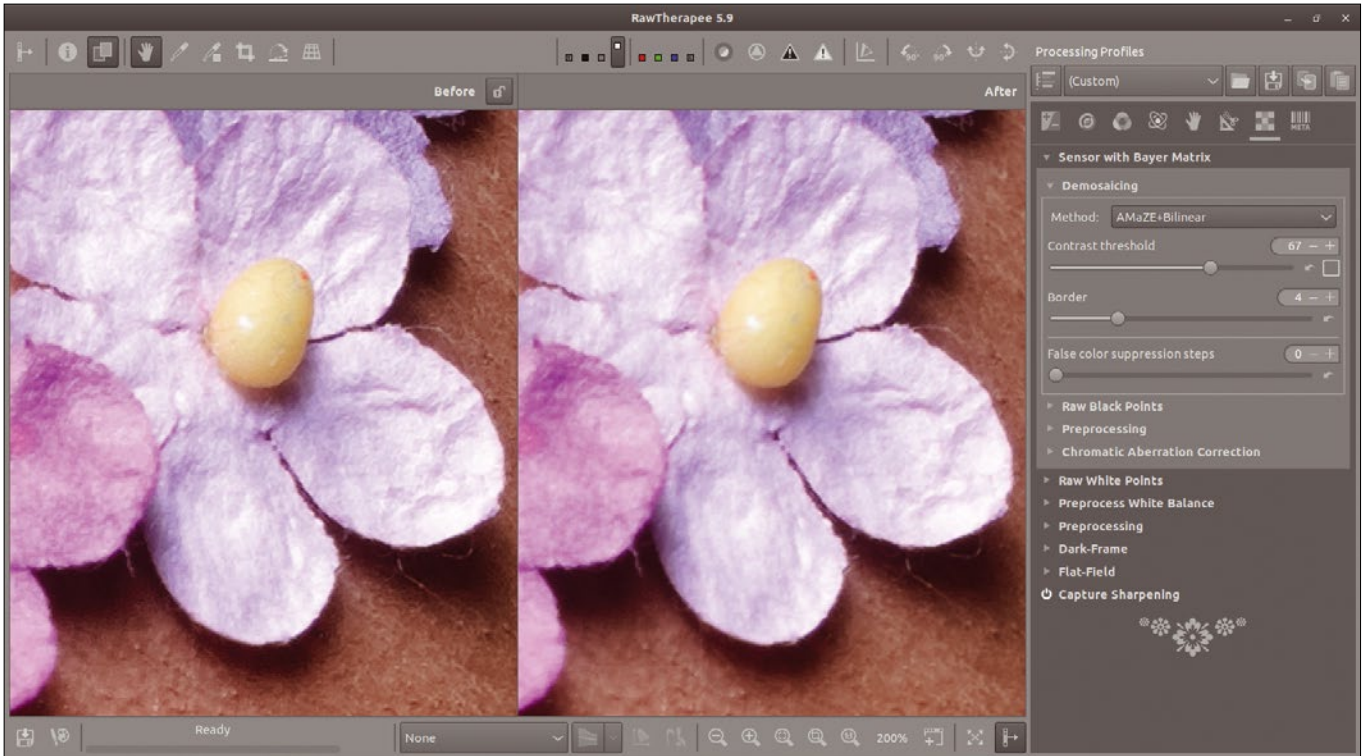


Figure 6: The new *AMaZE+Bilinear* demosaicing algorithm produces more granular noise, which the eye finds more pleasing.

Minimize the Noise

Zoom into the image and check the image noise and sharpness in each image area by moving the view. To do this, first click on the magnifying glass on the right side of the toolbar. Although the landscape photo has seen some intensive touch-up work, the image noise is relatively low. But the image can still be optimized even further. Before

you enable noise reduction, experiment a bit with *Color Interpolation* under *Raw*. This is mainly about the *Method*: Although *AMaZE* is the default value here, you should always try the *AMaZE+Bilinear* algorithm, too.

RawTherapee divides the image into high and low texture areas using an edge detection filter. For zones that have a rich structure, it uses the

Figure 7: Chromatic aberrations are frequently very stubborn; you may have to modify the automatic correction results manually.

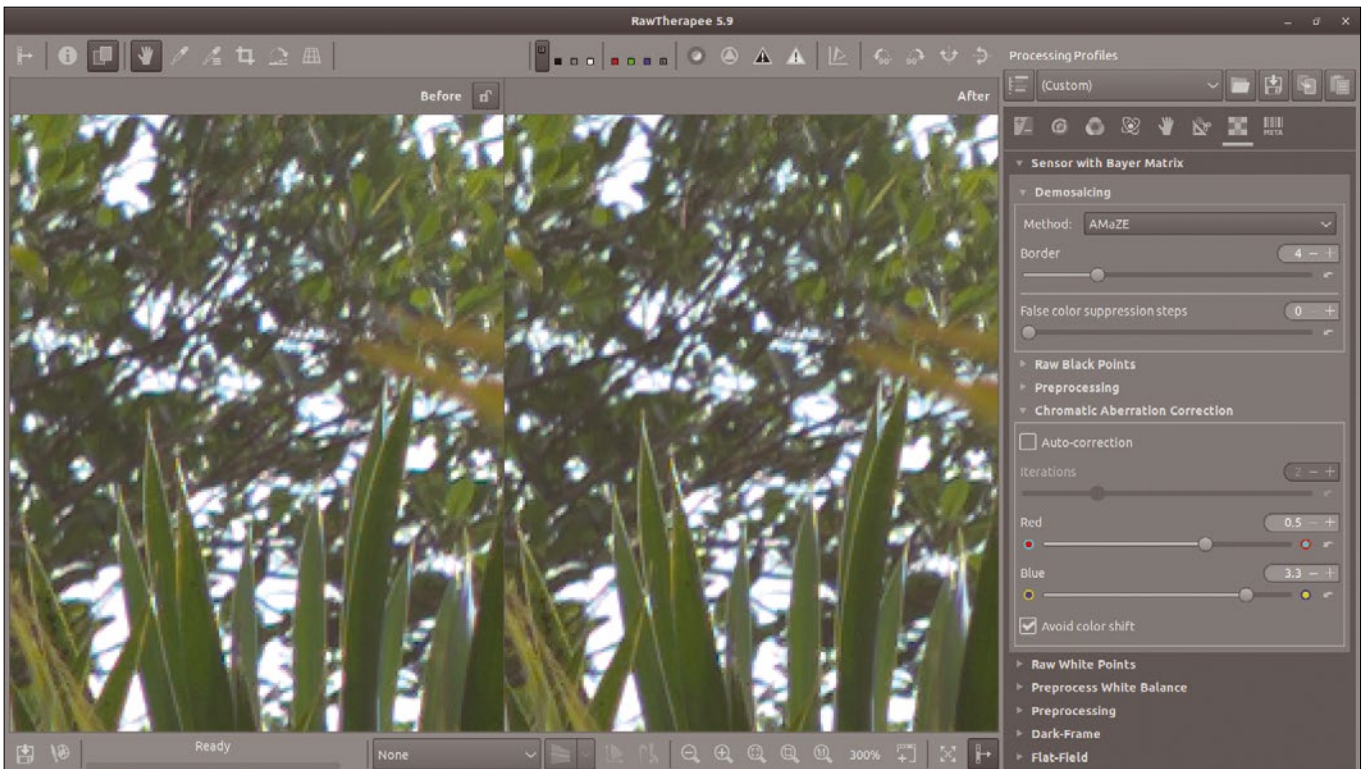




Figure 8: In most cases, distortion and vignetting can be corrected using the Lensfun lens-correction profiles.

sharper *AMaZE* descreening method, and for more monotonous areas, the softer *Bilinear* algorithm to nip any noise in the bud, so to speak. The *AMaZE+Bilinear* method will not necessarily attenuate the noise. The results look finer grained and more pleasing to the human eye (Figure 6). If you still find this too intense, enable *Noise Reduction* under *Details*. This removes the color noise, which is fine for newer cameras up to about ISO 1600.

Don't be too sensitive about image noise – this is a mistake most inexperienced photographers make. On the one hand, noise often doesn't even matter with newer cameras. Even images up to ISO 6400 are basically no longer a problem today. And low noise is not one of the most important characteristics of good photos. In fact, a certain amount of noise often even looks aesthetically pleasing, or at least more natural than a smoothed-out photo. Don't be afraid of high ISO values and don't denoise too aggressively.

Correcting Lens Aberrations

Correcting lens aberrations such as chromatic aberrations, vignetting, and distortion plays an important role in landscape photography. I will demonstrate this step with the help of two more sample photos, because there are virtually no errors of this type in the Mostviertel landscape.

Chromatic aberrations (a.k.a. purple fringing) are color fringes that often appear at contrasting edges, such as trees or branches in front of a bright sky. Automatic correction is already enabled in RawTherapee. However, sometimes the color fringes persist quite stubbornly, and you have to adjust the default

configuration. You will find the settings for this in *Raw* in the last group *Chromatic Aberrations* of the first module *Sensor with Bayer Matrix*.

Zoom into the image and move the view to the image area where the chromatic aberrations occur (Figure 7). Uncheck the *Auto-correction* box and then change the positions of the sliders for *Red* and *Blue* below it. Which slider you need to move in which direction depends on the color of the aberration. Blue and violet color fringes are very common.

For blue chromatic aberrations, move the *Blue* slider to the right until the color fringe disappears. If you see a yellow or yellow-green color fringe near the removed blue color fringe, you have adjusted the slider too far. For violet aberrations, you also need to move the *Red* slider a little to the right. Finally, check all image areas in the 100-percent view. What often happens, when chromatic aberrations are removed, is that new ones are created in other areas.

Distortion happens all the time. Pincushion distortion often occurs, especially with wide-angle lenses, and can be quite annoying for images with trees or a straight horizon. You will find the settings for this in *Transform* at the bottom of the *Lens Corrections* module. The *Automatic* option is selected in the *Lens Correction Profile* group. But this will not work with many cameras because lens selection fails. You need to select the right lens in the list next to *Lens*. After that, both the distortion and the vignetting should disappear.

What happens if you can't find the correction profile for your lens in the Lensfun database? In this case, first try the automatic distortion

correction below *Lens Correction Profile*. This works as long as the RAW file has a JPEG embedded in it, where the camera has automatically corrected the distortion. Otherwise you have to experiment with the slider below *Automatic*.

As mentioned, the Lensfun lens-correction profile also automatically removes vignetting (that is, circular shadowing), especially in the corners of the image. If you want or need to correct this manually, use the *Vignetting Correction* tool in *Transform*. This tool lets you add a vignette by dragging the *Intensity* slider to the left. In general, vignetting is not as disturbing as chromatic aberrations or distortion; quite often, a slight vignette even has an aesthetic effect (Figure 8).

Finally, export the photo. To do this, click on the floppy-disk icon bottom left. The Save dialog box appears. When you get there, select the *Target folder*, the *File format* (JPEG, PNG, or TIFF), the *Bit depth* (8- or 16-bit), and the *Compression*, and press *OK* to confirm.

Conclusions

RawTherapee is great for developing landscape photos, not least thanks to the new local

adjustments. The application gives you options for finding a perfect color balance across the image. As a result, you no longer need to resort to other tools for the final editing steps.

In fact, the software is, in our opinion, one of the best RAW developers currently available. RawTherapee is fun to work with, because it runs at an acceptable speed even on older hardware and is relatively easy to use. In addition, the results can definitely compete with the results of far better-known programs. If the developers can find a way to implement 10-bit screen output, RawTherapee is likely to continue to thrill its users for many years to come. ■■■

Info

- [1] RawTherapee: <https://www.rawtherapee.com>
- [2] RawPedia (official RawTherapee documentation): http://rawpedia.rawtherapee.com/Main_Page
- [3] "RawTherapee Basics, The New Local Adjustments – pt 1": <https://www.youtube.com/watch?v=dE0gofVBJr0>
- [4] "RawTherapee Local Adjustments – pt 2": <https://www.youtube.com/watch?v=vvmBfgFAXXM>



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

Linux Update: bit.ly/Linux-Update

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.



#272/July 2023

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.

On the DVD: xubuntu 23.04 Desktop and Fedora 38 Workstation



#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

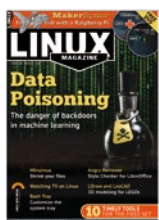


#269/April 2023

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

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.



RustConf 2023

Date: September 12-15, 2023

Location:

Albuquerque, New Mexico & Online

Website: <https://rustconf.com/>

Don't miss the largest annual gathering of the Rust-language community September 12-15 in Albuquerque, New Mexico. This event includes talks and trainings from top Rust developers, digital and in-person discussions, and much more! Learn more and register today at rustconf.com.

EuroBSDCon 2023

Date: September 14-17, 2023

Location: Coimbra, Portugal

Website: <https://2023.eurobsdcon.org/>

EuroBSDCon is the international annual technical conference gathering users and developers working on and with 4.BSD (Berkeley Software Distribution) based operating systems family and related projects. Don't miss the opportunity to learn about the latest news from the BSD world, witness contemporary deployment case studies, and meet other users and companies using BSD oriented technologies.

SDC '23

Date: September 18-21, 2023

Location: Fremont, California

Website: <https://storagedeveloper.org/>

We are excited to announce that SDC'23 will be a face-to-face event once again this year. Attending SDC is a cost-effective way to acquire training in a host of different areas through tutorials, sessions, Keynote speakers, and participation in the co-located plugfests. Take advantage of this opportunity to learn from the experts all in one place.

Events

GUATEC	July 26-27	Riga, Latvia	https://events.gnome.org/event/101/
WeAreDevelopers World Congress	July 27-28	Berlin, Germany	https://www.wearedevelopers.com/world-congress/
RustConf 2023	Sep 12-15	Albuquerque, New Mexico	https://rustconf.com/
stackconf 2023	Sep 13-14	Berlin, Germany	https://stackconf.eu/
EuroBSDCon 2023	Sep 14-17	Coimbra, Portugal	https://2023.eurobsdcon.org/
Storage Developer Conference (SDC'23)	Sep 18-21	Fremont, California	https://storagedeveloper.org/
FOSS Security Campus	Sep 26-29	Berlin, Germany	https://foss-security-campus.de/
All Things Open	Oct 15-17	Raleigh, North Carolina	https://www.allthingsopen.org/
DrupalCon Lille 2023	Oct. 17-20	Lille, France	https://events.drupal.org/lille2023
LinuxFest Northwest 2023	Oct 20-22	Bellingham, Washington	https://linuxfestnorthwest.org/
Hybrid Cloud Conference	Oct 26	Virtual Event	https://www.techforge.pub/events/hybrid-cloud-congress-2/
SeaGL 2023	Nov 3-4	Virtual Event	https://seagl.org/
Open Source Monitoring Conference (OSMC)	Nov 7-9	Nuremberg, Germany	https://osmc.de/
SFSCON 2023	Nov 10-11	Bolzano, Italy	https://www.sfscon.it/
SC23	Nov 12-17	Denver, Colorado	https://sc23.supercomputing.org/
Cassandra Summit	Dec 12-13	San Jose, California + Virtual	https://events.linuxfoundation.org/cassandra-summit/

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

Lori White

Cover Image

© aleksanderdn, 123RF.com

AdvertisingBrian 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.

Please indicate in the subject line that your message is an article proposal.

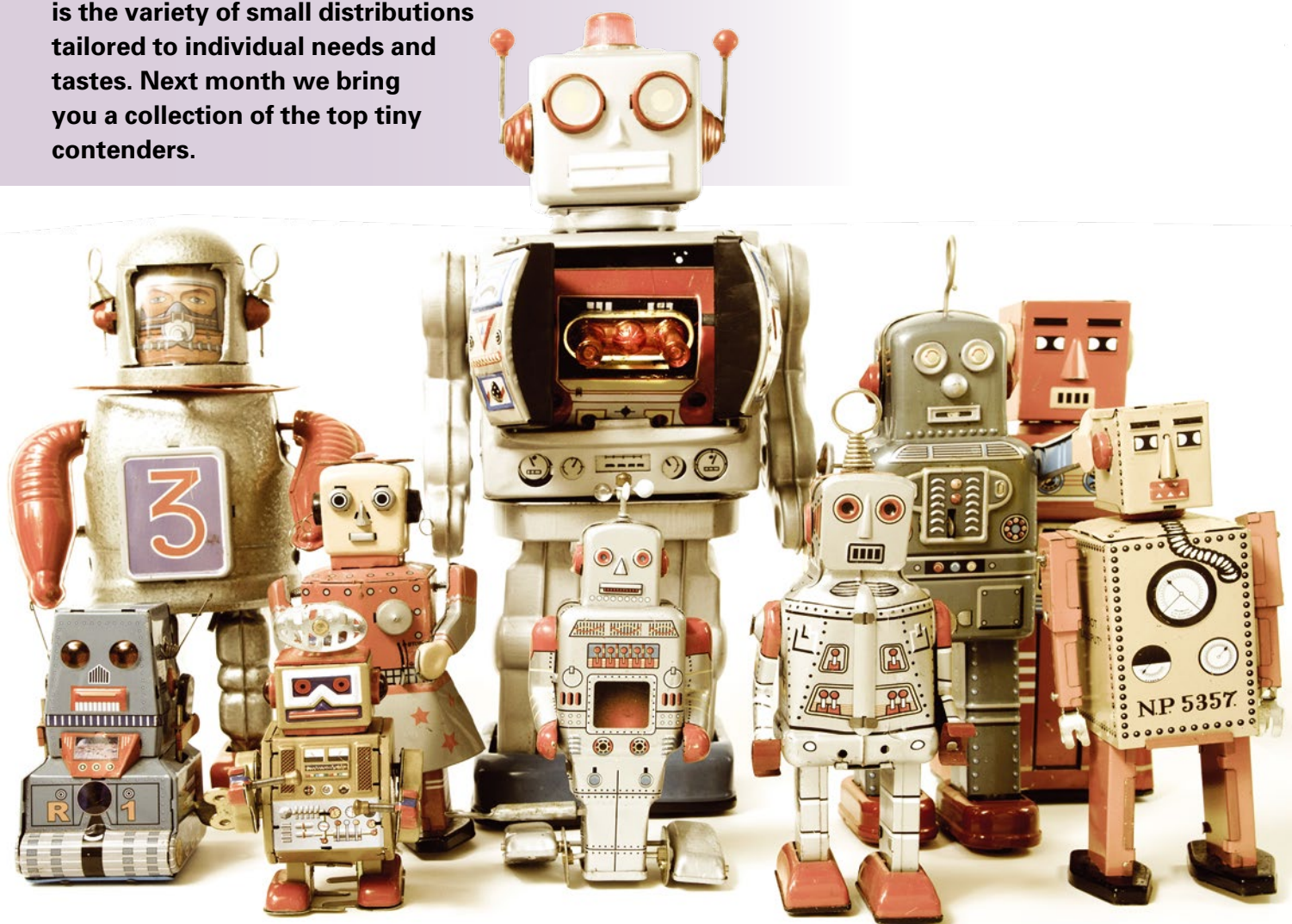
Authors

Zack Brown	12	Graham Morrison	16, 82
Bruce Byfield	6, 36, 40	Hartmut Noack	24
Joe Casad	3, 32	Mike Schilli	58
Mark Crutch	75	Anna Simon	88
Adam Dix	48	Scott Sumner	64
Jon "maddog" Hall	77	Ferdinand Thommes	78
Ken Hess	20	Jack Wallen	8
Vincent Mealing	75	Andrew Williams	42

Available Starting
August 4

Best of Small Distros

One of the great pleasures of Linux is the variety of small distributions tailored to individual needs and tastes. Next month we bring you a collection of the top tiny contenders.



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 © charles taylor, 123RF.com

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



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



VS



Mobility



Battery life



TUXEDO InfinityBook Pro 14 - Gen8

17 mm flat & 1.3 kg light. Premium business notebook in ultra portable magnesium case for maximum mobility.



CPU performance

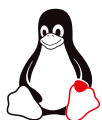


GFX performance



TUXEDO Stellaris 16 - Gen5

Top performance on desktop PC level thanks to GeForce RTX 4090 and Intel Core i9 in a compact form factor.



Linux compatible



Up to 5 Years Guarantee



Immediately ready for use



Made in Germany



German Data Privacy



German Tech Support

TUXEDO

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